NAACL HLT 2018

The 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies

Proceedings of the Conference
Volume 1 (Long Papers)

June 1-June 6, 2018
New Orleans, Louisiana

Diamond Sponsors

Platinum Sponsors

Gold Sponsors

Silver Sponsors

Bronze Sponsors

Order copies of this and other ACL proceedings from:

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

ISBN 978-1-948087-27-8

Message from the General Chair

Welcome to New Orleans and to NAACL HLT 2018 – the biggest NAACL to date. Natural Language Processing and Computational Linguistics is constantly growing and changing with a constant flow of new methods and topics. Every year also sees an even more exciting and diverse research community, with a steadily increasing number researchers, companies both large and small, and a vibrant community of practitioners and students who are excited at the prospect of taking on the newest challenges of the discipline. This year's NAACL HLT conference reflects what an exciting time this is for our field, and highlights the vibrancy and vitality of our community.

I feel extremely lucky to be able to work with a fantastic program committee, especially the two extremely dedicated, creative and resourceful program chairs: Amanda Stent and Heng Ji. Their innovations include a new review form, intended to elicit higher quality reviews, the opportunity for authors to review the reviewers, the Test-of-Time awards, and a program where poster and demo sessions run consistently in parallel to the oral sessions, in order to allow the conference to reflect the ever increasing diversity of research topics and the corresponding volume of accepted papers. I am especially excited about the new Test-of-Time papers award session, and hope to see this new innovation become a regular part of ACL conferences.

We have named the Test-of-Time award in memory of Aravind Joshi, who left us this year, after having a huge lifetime impact on our community. We will always remember him for his gentle conversational style, sharp focus, interest in linguistic, computational and mathematical properties of language, and his lifetime commitment to mentoring women in NLP. I feel extremely lucky to have been one of his Ph.D. students.

This year we also introduced an industrial track, with the aim of featuring papers that focus on scalable, interpretable, reliable and customer facing methods for industrial applications of Natural Language Processing. The idea of having such a track was proposed by Yunyao Li who strongly advocated for it: this proposal was then discussed and approved by the NAACL board. After that, it was all go, with an incredible amount of work to promote and organize it by the industrial track chairs: Jennifer Chu-Carroll, Yunyao Li and Srinivas Bangalore.

The overall program looks amazing and reflects the cooperative way that everyone on the committee worked together. What a team! I am so grateful for getting to be a part of this community of people, and I really appreciate the enthusiasm and attention to detail reflected in their hard work: Amanda Stent and Heng Ji (program chairs); Jennifer Chu-Carroll, Yunyao Li and Srinivas Bangalore (industrial track chairs); Ying Lin (website chair); Marie Meteer and Jason Williams (workshop co-chairs); Mohit Bansal and Rebecca Passonneau (tutorial co-chairs); Yang Liu, Tim Paek, and Manasi Patwardhan (demo co-chairs); Chris Callison-Burch and Beth Hockey (Family-Friendly Program Co-Chairs) Stephanie Lukin and Meg Mitchell (publication co-chairs); Jonathan May (handbook chair); Silvio Ricardo Cordeiro, Shereen Oraby, Umashanthi Pavalanathan, and Kyeongmin Rim (student cochairs) along with Swapna Somasundaran and Sam Bowman (Faculty Advisors) for the student research workshop; Lena Reed (student volunteer coordinator); Kristy Hollingshead, Kristen Johnson, and Parisa Kordjamshidi (local sponsorships and exhibits cochairs); Yonatan Bisk and Wei Xu (publicity and social media chairs); David Yarowsky and Joel Tetreault (treasurers) and Alexis Palmer and Jason Baldridge (the NAACL international Sponsorship Team). Also thanks to Rich at SoftConf for his speedy response to questions and his willingness to help us innovate with our new review form. And thanks to Julia Hockenmaier and the whole NAACL Executive Board for always being willing to consult on any issue.

The program highlights three keynote speakers in the main track: Dilek Hakkani-Tür, Kevin Knight, and Charles Yang. We also have two keynote speakers in the industry track: Mari Ostendorf and Daniel Marcu. These talks promising to be interesting across a range of issues from language acquisition in

children to the commercial possibilities of conversational agents. The industry track will also feature two panels, one on careers in industry (as compared to academia) and the other on ethics in NLP. The program also includes six tutorials featuring topics of current interest and sources of innovation in the field. We have sixteen workshops plus the student research workshop: some of these workshops have become events in themselves with many of them repeated each year. We will also have plenary sessions for the outstanding paper awards and the new Test-Of-Time papers award session.

Any event of this scale can only happen with the the hard work of a wonderful group of people. I especially want to thank the NAACL board for being willing to consult on a range of different issues and Priscilla Rasmussen for taking care of all the millions of details that need to be looked after every single day to make sure the logistical aspects of the conference come together. I want to especially thank Priscilla for her hard work and creativity organizing our social event: we first will go to Mardi Gras World to see the world of wonders created each year for the Mardi Gras. From there we go to the river, to the dockside River City Plaza and River City Ballroom for New Orleans' famous cuisine and libations and dancing to live Zydeco, funk, soul and R&B.

ACL has been working for several years to increase diversity at our conferences and in our community. So, taking inspiration from ACL 2017, we aimed to make NAACL family friendly, by providing childcare at the conference, and encouraging people to bring their families to the social events and breakfasts. Diversity can also be a consequence of the support for students to attend the conference that we receive from the NSF, CRA-CCC and CRA-W: this subsidizes student travel to the student research workshop as well as their registration and ACL memberships. When combined with the support we are able to give to our student volunteers, we aim to make it possible for students from all over the world to come to the conference and be part of our community. We also decided, in consultation with the NAACL board, to provide subsidies to the Widening NLP workshop, which is only being held for the second time at this year's NAACL (last year called the Women in NLP workshop). These subsidies enable participation from students and young researchers from developing countries to attend the conference.

I am grateful to our sponsors for their generous contributions, which add so much to what we can do at the conference. Our Diamond sponsors are Bloomberg, Google, and Toutiao AI Lab (ByteDance). The Platinum sponsor is Amazon. The Gold Sponsors are Ebay, Grammarly, IBM Research, KPMG, Oracle, Poly AI, Tulane University, Capital One and Two Sigma. The Silver sponsors are Nuance and Facebook, and the Bronze sponsors are iMerit and USC/ISI.

Finally, there are many more people who through their hard work and dedication have contributed to make this conference a success: the area chairs, workshop organizers, tutorial presenters, student mentors, and reviewers. And of course you all, the attendees without whom there would be no conference: you are the life and spirit of the conference and the NAACL community. I hope you all have a fun and exciting time at NAACL HLT 2018!

NAACL HLT 2018 General Chair
Marilyn Walker, University of California Santa Cruz

Message from the Program Co-Chairs

We welcome you to New Orleans for the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2018)! We had three primary goals for NAACL HLT 2018: construct a great program; manage the integrity and quality of the publication process; and ensure broad participation.

Construct a great program: NAACL HLT 2018 does have a great program, thanks to all of you! We will have three exciting keynotes, from Charles Yang, Kevin Knight and Dilek Hakkani-Tür. 331 research track papers (205 long, 125 short), accepted following peer review, will be presented[1]. Four of these papers have been identified as outstanding papers, and one will be named best paper. We will also feature a "Test of Time" session with retrospectives (from the authors) on three influential papers from ACL venues. We thank the committees who nominated and voted on these paper awards.

The main program at NAACL HLT 2018 also includes 16 TACL paper presentations, 20 demos, a student research workshop and an industry track. Keynotes from both the research and industry tracks are plenary. In a change from previous years of NAACL HLT, and motivated by EMNLP 2017, poster and oral presentation sessions will be held in parallel during the day. All posters are grouped thematically (including posters from the industry track and student research workshop and demos) and assigned to poster sessions so as to not be against oral presentation sessions with the same theme.

Manage the integrity and quality of the publication process: To manage load, we decided that each area chair should be responsible for no more than 30 submissions and that reviewers should be responsible for reviewing no more than 3 submissions. To help reviewers, we and the ACL program co-chairs constructed a more structured review form, with questions related to the new ACL guidelines on publication and reviewing, as well as to contribution types, experimental methods (thank you, Bonnie Webber!), software and handling of data.

We recruited an excellent group of 72 area chairs; we thank them for their leadership, and for nominating and voting on outstanding papers, outstanding reviewers and test of time papers. 1372 individuals reviewed papers for the conference (as program committee members, ad hoc reviewers or secondary reviewers); all but 49 reviewers had no more than 3 submissions to review overall, and the 49 reviewers who took on a heavier load did so voluntarily. We thank all our reviewers, especially the ad-hoc reviewers who provided last minute reviews and the outstanding reviewers identified by the area chairs.

Submissions were assigned to area chairs and reviewers using a combination of area chair expertise, Toronto Paper Matching System (TPMS) scores and reviewer bids. Both long and short paper submissions received 3 reviews each. Long paper authors had an opportunity to respond to reviews. Accept/reject suggestions were made by area chairs working in small groups of 2-3 and discussing with reviewers as necessary; final decisions were made by the program chairs. Where there was disagreement or discussion, one area chair wrote a short meta review that was shared with the authors.

This year, if the authors of a NAACL HLT 2018 submission and the author of a review for that submission both consented, then we will include the review in a review corpus to be released jointly with the program chairs of ACL, Iryna Gureyvich and Yusuke Miyao. We also asked authors of accepted papers to upload the source code for their papers. Both of these corpora will be released in the coming months.

The health of our field as a science is dependent on a scalable peer review process, which in turn depends on (a) conscientious effort from a broad pool of expert reviewers, and (b) tools, processes and policies that can structure and facilitate reviewing. As a field we are at a breaking point: we are growing rapidly,

[1]We received 1122 research track submissions (664 long, 458 short). 33 were rejected without review and 85 were withdrawn by the authors either before, during or after review.

with corresponding heavy load on experienced reviewers; and we lack good tools to manage the process. Peer review involves several tasks that we, as NLP researchers, ought to be uniquely qualified to address, including expertise sourcing, network analysis and text mining. We have written a proposal with other members of the ACL community about ways the ACL can improve our peer review infrastructure. We have also written a collection of "how to" documents that we will pass on to future conference organizers.

Ensure broad participation: To ensure broad participation, we recruited program committee members using a similar method to that used for NAACL HLT 2016: we invited anyone who had published repeatedly in ACL sponsored venues, who had a PhD or significant experience in the field spanning more than 5 years, and whose email address was up to date in START. We thank Dragomir Radev for giving us a list of names from the ACL anthology.

We also kept a blog where we discussed and attempted to "demystify" each stage of the publication process. This blog can be found at the conference website, `http://naacl2018.org`. We are very grateful to the researchers who wrote guest blog posts, including Justine Cassell, Barbara Plank, Preslav Nakov, Omer Levy, Gemma Boleda, Emily Bender, Nitin Madnani, David Chiang, Kevin Knight, Dan Bikel and Joakim Nivre.

On our blog, we reported on the diversity of our area chair, reviewer and author pools in terms of years of experience, affiliation type and geography, and gender. We will include these details in our report to the NAACL Executive Committee. We hope that future years' chairs will make similar reports.

The excellence of the overall NAACL HLT 2018 program is thanks to all the chairs and organizers. We especially thank the following people: Margaret Mitchell and Stephanie Lukin, the publication chairs; Jonathan May, the handbook chair; Yonatan Bisk and Wei Xu, the publicity and social media chairs; Ying Lin, the tireless website chair; and Marilyn Walker, the NAACL HLT 2018 general chair. We thank the chairs of NAACL HLT 2016 and ACL 2017 for their informative blogs, and the program chairs of NAACL HLT 2016, Owen Rambow and Ani Nenkova, for their advice. We thank the program co-chairs of ACL 2018, Iryna Gurevych and Yusuke Miyao, who have been very collaborative on matters related to reviewing. We thank Shuly Winter, who helped fix a serious START bug. We thank Julia Hockenmaier and the NAACL Executive Committee for their support. We are grateful for the professional work of Rich Gerber and his colleagues at SoftConf (START), and of Priscilla Rasmussen from the ACL.

It has been an enormous privilege for us to see the scientific advances that will be presented at this conference. We would like to close with some advice for you, the conference attendees.

- The presenters have made valuable contributions to our science; their oral, poster and demo presentations are worth your time and attention.
- Talk to some people you haven't previously met. They may be your future collaborators!
- You can follow the conference on social media; we have a conference app and website where we will post any updates to the program, and our twitter handle is @naaclhlt.
- This event is run by a professional organization with a code of conduct[2]. If you observe or are the recipient of unprofessional behavior, you may contact any current member of the ACL or NAACL Executive Committees, the NAACL HLT general chair (Marilyn Walker), us (the program chairs), or Priscilla Rasmussen (acl@aclweb.org). We will hold your communications in strict confidence and consult you before taking any action.

We look forward to a wonderful conference!

NAACL HLT 2018 Program Co-Chairs
Heng Ji, RPI
Amanda Stent, Bloomberg

[2] `https://www.aclweb.org/adminwiki/index.php?title=Anti-Harassment_Policy`

Organizing Committee

General Chair

Marilyn Walker, University of California, Santa Cruz

Program Co-Chairs

Heng Ji, Rensselaer Polytechnic Institute
Amanda Stent, Bloomberg

Industry Track Co-Chairs

Srinivas Bangalore, Interactions
Jennifer Chu-Carroll, Elemental Cognition
Yunyao Li, IBM

Demonstration Co-Chairs

Yang Liu, University of Texas at Dallas
Tim Paek, Apple
Manasi Patwardhan, Tata Consultancy Services Research, India

Student Research Workshop Co-Chairs

Chairs:

Silvio Ricardo Cordeiro, Aix-Marseille Université
Shereen Oraby, University of California, Santa Cruz
Umashanthi Pavalanathan, Georgia Institute of Technology
Kyeongmin Rim, Brandeis University

Faculty Advisors:

Swapna Somasundaran, ETS Princeton
Sam Bowman, New York University

Workshop Co-Chairs

Marie Meteer, Brandeis University
Jason Williams, Microsoft Research

Tutorial Co-Chairs

Mohit Bansal, University of North Carolina
Rebecca Passonneau, Pennsylvania State University

Publications Co-Chairs

Stephanie Lukin, U.S. Army Research Laboratory
Margaret Mitchell, Google

Handbook Chair

Jonathan May, University of Southern California Information Sciences Institute

Local Arrangements Chair

Priscilla Rasmussen, ACL

Family-Friendly Program Co-Chairs

Chris Callison-Burch, University of Pennsylvania
Beth Hockey, Intel

International Sponsorships Co-Chairs

Alexis Palmer, University of North Texas
Jason Baldridge, Google

Local Sponsorships and Exhibits Co-Chairs

Kristy Hollingshead, Florida Institute for Human & Machine Cognition
Kristen Johnson, Purdue University
Parisa Kordjamshidi, Tulane University / Florida Institute for Human and Machine Cognition

Publicity and Social Media Co-Chairs

Yonatan Bisk, University of Washington
Wei Xu, Ohio State University

Website Chair

Ying Lin, Rensselaer Polytechnic Institute

Student Volunteer Coordinator

Lena Reed, University of California, Santa Cruz

Program Committee, Research Track

Area Chairs

Cognitive Modeling/Psycholinguistics
- Morteza Dehghani, University of Southern California
- Kristy Hollingshead, Institute for Human & Machine Cognition

Dialogue and Interactive Systems
- Yun-Nung (Vivian) Chen, National Taiwan University
- Gabriel Skantze, KTH Royal Institute of Technology

Discourse and Pragmatics
- Jacob Eisenstein, Georgia Institute of Technology
- Junyi (Jessy) Li, University of Texas at Austin
- Annie Louis, University of Edinburgh
- Yi Yang, Bloomberg

Generation
- Dimitra Gkatzia, Edinburgh Napier University
- Shashi Narayan, University of Edinburgh
- Michael White, Ohio State University

Information Extraction
- Mausam, Indian Institute of Technology Delhi
- Dan Bikel, Google
- Chia-Hui Chang, National Central University
- Bonan Min, BBN
- Aurélie Névéol, CNRS

Marius Pasca, Google
Hinrich Schütze, Ludwig Maximilian University of Munich
Avirup Sil, IBM Research AI
Michael Strube, HITS gGmbH

Machine Learning for NLP

Chris Dyer, Google DeepMind
Ozan Irsoy, Bloomberg
Tie-Yan Liu, Microsoft Research
Raymond Mooney, University of Texas at Austin

Machine Translation

Marine Carpuat, University of Maryland
Kyunghyun Cho, New York University
Daniel Marcu, Amazon
Taro Watanabe, Google
Deyi Xiong, Soochow University

NLP Applications

Jinho Choi, Emory University
Joel Tetreault, Grammarly

Phonology, Morphology and Word Segmentation

Jennifer Foster, Dublin City University
Barbara Plank, University of Groningen

Question Answering

Eugene Agichtein, Emory University
Hannaneh Hajishirzi, University of Washington
Idan Szpektor, Google

Semantics

Yoav Artzi, Cornell University
Mona Diab, George Washington University
Kevin Duh, Johns Hopkins University
Jonathan May, University of Southern California
Preslav Nakov, Qatar Computing Research Institute
Roi Reichart, Technion - Israel Institute of Technology
Dan Roth, University of Pennsylvania
Scott Wen-tau Yih, Allen Institute for Artificial Intelligence

Sentiment Analysis

Smaranda Muresan, Columbia University
Swapna Somasundaran, ETS Princeton

Social Media Analysis and Computational Social Science

Mark Dredze, Johns Hopkins University
Miles Osborne, Bloomberg
Alan Ritter, Ohio State University
Sara Rosenthal, IBM
William Yang Wang, University of California, Santa Barbara

Speech

Eric Fosler-Lussier, Ohio State University
Dilek Hakkani-Tur, Google
Mari Ostendorf, University of Washington

Summarization

George Giannakopoulos, NCSR “Demokritos"
Xiaojun Wan, Peking University
Lu Wang, Northeastern University

Tagging, Chunking, Syntax and Parsing

Michael Collins, Columbia University
Yoav Goldberg, Bar Ilan University
Daisuke Kawahara, Kyoto University
Emily Pitler, Google
Anders Søgaard, University of Copenhagen
Aline Villavicencio, University of Essex / Federal University of Rio Grande do Sul

Text Mining

Kai-wei Chang, University of Virginia
Jing Jiang, Singapore Management University
Zornitsa Kozareva, Google
Chin-Yew Lin, Microsoft Research Asia

Theory and Formalisms

David Chiang, University of Notre Dame
Daniel Gildea, University of Rochester
Giorgio Satta, University of Padua

Vision, Robotics and Other Grounding

Joyce Chai, Michigan State University
Vicente Ordonez, University of Virginia

Reviewers

Reviewers who are acknowledged by the program committee for providing one or more outstanding reviews are listed in bold. Individuals who (also) served as secondary reviewers are marked with † and those who (also) served as ad-hoc reviewers with *.

Mourad Abbas
Ahmed Abdelali
Asad Abdi
Muhammad Abdul-Mageed
Anne Abeille
Omri Abend
Abhishek Abhishek
Mohamed Abouelenien
José Abreu
Sallam Abualhaija
Amjad Abu-Jbara
Lasha Abzianidze
Oliver Adams
Gilles Adda
Yvonne Adesam
Stergos Afantenos
Naveed Afzal
Apoorv Agarwal
Swati Agarwal
Željko Agić
Eugene Agichtein
Manex Agirrezabal
Lars Ahrenberg
Salah Ait-Mokhtar
Yamen Ajjour†
Alan Akbik
Ahmet Aker
Cem Akkaya*
Ahmad Al Sallab
Özge Alacam
Domagoj Alagić†
Sameh Alansary
Iñaki Alegria
Nikolaos Aletras
Beatrice Alex
Theodora Alexopoulou
Enrique Alfonseca
Ahmed Ali
Dimitris Alikaniotis
Hend Al-Khalifa
Mehdi Allahyari
Miltiadis Allamanis
James Allan
Alexandre Allauzen
Miguel A. Alonso
Omar Alonso
Laura Alonso Alemany
Abdulrahman Alosaimy
Rami Al-Rfou
Edgar Altszyler
Nora Al-Twairesh
Bharat Ram Ambati
Waleed Ammar*
Dimitra Anastasiou
Tim Anderson
Jacob Andreas
Nicholas Andrews
Ion Androutsopoulos
Anietie Andy
Gabor Angeli*
Marianna Apidianaki
Emilia Apostolova
Jun Araki
Masahiro Araki
Shlomo Argamon
Piyush Arora
Raman Arora
Sanjeev Arora
Masayuki Asahara
Elliott Ash
Nicholas Asher
Giuseppe Attardi

Aitziber Atutxa Salazar†
Isabelle Augenstein
Michael Auli
AiTi Aw
Amittai Axelrod
Wilker Aziz
Homa B. Hashemi
Collin Baker
Omid Bakhshandeh
Niranjan Balasubramanian
Timothy Baldwin
Tyler Baldwin
Kalika Bali
Miguel Ballesteros
Ramy Baly
David Bamman
Rafael E. Banchs
Ritwik Banerjee
Mohit Bansal
Libby Barak
Denilson Barbosa
Alistair Baron
Marco Baroni
Alberto Barrón-Cedeño
Valerio Basile
Roberto Basili
Tanmay Basu
Daniel Bauer
Timo Baumann
Anshul Bawa†
John Bear
Frédéric Béchet
Barend Beekhuizen
Beata Beigman Klebanov
Yonatan Belinkov
Charley Beller
Anja Belz
Michael Bendersky
Jonathan Berant
Raffaella Bernardi
Delphine Bernhard
Yevgeni Berzak
Steven Bethard
Rahul Bhagat
Chandra Bhagavatula
Aditya Bhargava†
Pushpak Bhattacharyya
Chris Biemann
Ann Bies
Or Biran
Yonatan Bisk
Prakhar Biyani
Yuri Bizzoni†
Anders Björkelund
Philippe Blache
Alan W Black
Eduardo Blanco
Roi Blanco
Nate Blaylock
Reihane Boghrati*
Bernd Bohnet
Piotr Bojanowski
Ondřej Bojar
Danushka Bollegala
Marcel Bollmann
Kalina Bontcheva
Benjamin Börschinger
Florian Boudin
Fethi Bougares
Pierrette Bouillon
Gosse Bouma
Johan Boye
Kristy Boyer
David Bracewell
Ellen Breitholz
Chris Brew
Ted Briscoe
Chris Brockett
Julian Brooke
Pawel Budzianowski†
Paul Buitelaar
Harry Bunt
Wray Buntine
David Burkett
Jill Burstein
Miriam Butt
Donna Byron
José G. C. de Souza
Aoife Cahill
Deng Cai†
Ruket Cakici
Iacer Calixto*
Chris Callison-Burch
Nicoletta Calzolari
Jose Camacho-Collados
Leonardo Campillos Llanos*†
Marie Candito
Kai Cao*
Liangliang Cao
Yuan Cao
Ziqiang Cao
Cornelia Caragea
Giuseppe Carenini
Mark Carman
Xavier Carreras
Vitor Carvalho
Francisco Casacuberta
Taylor Cassidy
Vittorio Castelli
Damir Cavar
Asli Celikyilmaz
Daniel Cer
Özlem Çetinoğlu*
Soumen Chakrabarti
Tanmoy Chakraborty
Yllias Chali
Nathanael Chambers
Yee Seng Chan
Muthu Kumar Chandrasekaran
Angel Chang
Franklin Chang
Yin-Wen Chang
Yung-Chun Chang
Lidia S. Chao
Snigdha Chaturvedi
Wanxiang Che
Ciprian Chelba
Berlin Chen
Bin Chen
Boxing Chen
Changyou Chen
Chen Chen
Cheng Chen
Hsin-Hsi Chen
Jianfu Chen
John Chen
Kehai Chen†
Kuan-Yu Chen
Ping Chen
Qian Chen
Qingcai Chen
Wenliang Chen
Xinchi Chen
Yubo Chen
Zhiyuan Chen
Jianpeng Cheng
Pengxiang Cheng†
Weiwei Cheng
Hadda Cherroun
Colin Cherry
Emmanuele Chersoni†
Sean Chester
Jackie Chi Kit Cheung

Niyati Chhaya
Lianhua Chi
Hai Leong Chieu
Manoj Chinnakotla
Laura Chiticariu
Martin Chodorow
Eunsol Choi
Jinho D. Choi
Key-Sun Choi
Shamil Chollampatt
Monojit Choudhury
Christos Christodoulopoulos
Jason Chuang
Junyoung Chung
Tagyoung Chung
Kenneth Church
Philipp Cimiano
Alina Maria Ciobanu
Stephen Clark
Mathieu Cliche
Martin Cmejrek
Kevin Cohen
Shay B. Cohen
William Cohen
Michael Collins*
Çağrı Çöltekin*
John Conroy
Matthieu Constant
Paul Cook
Matthew Coole†
Robin Cooper
Silvio Cordeiro*
Mark Core
Marta R. Costa-jussà
Ryan Cotterell
Aaron Courville
Benoit Crabbé
Josep Crego
Dan Cristea
Danilo Croce
Paul Crook
Scott Crossley
Heriberto Cuayáhuitl
Lei Cui
Xiaodong Cui
Yiming Cui
Aron Culotta
Giovanni Da San Martino
Walter Daelemans
Andrew Dai
Bo Dai*
Beatrice Daille
Bhavana Dalvi
Marco Damonte
Falavigna Daniele
Kareem Darwish
Dipanjan Das
Pradipto Das
Sarthak Dash
Hal Daumé III
Johannes Daxenberger
Adrià de Gispert
Eric De La Clergerie
Marie-Catherine de Marneffe
Gerard de Melo
Renato de Mori
Edward Dearden†
Luciano Del Corro
Louise Deléger
Rodolfo Delmonte
Dina Demner-Fushman
Steve DeNeefe
John DeNero
Lingjia Deng
Zhi-Hong Deng
Pascal Denis
Michael Denkowski
Tejaswini Deoskar
Nina Dethlefs
Luis Fernando D'Haro
Giuseppe Di Fabbrizio
Weicong Ding
Stefanie Dipper
Nemanja Djuric
Simon Dobnik
Ellen Dodge
Jesse Dodge
Daxiang Dong
Li Dong
Bonnie Dorr
Doug Downey
Eduard Dragut
Mark Dras
Markus Dreyer
Aleksandr Drozd
Jinhua Du
Lan Du
Nan Duan
Ewan Dunbar
Long Duong
Nadir Durrani
Greg Durrett
Ondřej Dušek
Marc Dymetman
Seth Ebner†
Richard Eckart de Castilho
Judith Eckle-Kohler
Steffen Eger
Markus Egg
Koji Eguchi
Patrick Ehlen
Vladimir Eidelman
Andreas Eisele
Asif Ekbal
Layla El Asri
Roxanne El Baff†
Ahmed El Kholy
Shady Elbassuoni
Michael Elhadad
Mahmoud El-Haj
Hady Elsahar†
Tamer Elsayed
Micha Elsner
Akiko Eriguchi
Katrin Erk
Arash Eshghi
Ramy Eskander
Maxine Eskenazi
Keelan Evanini
Roger Evans
David Fagnan†
James Fan
Lisa Fan
Hao Fang
Stefano Faralli
Richárd Farkas
Noura Farra*
Manaal Faruqui
Nicolas R Fauceglia
Marcello Federico
Christian Federmann
Wes Feely†
Anna Feldman
Minwei Feng
Yansong Feng
Elisa Ferracane†
Francis Ferraro
Olivier Ferret
Oliver Ferschke
Simone Filice
Denis Filimonov
Katja Filippova
Andrew Finch

Gregory Finley
Nicholas FitzGerald
Eileen Fitzpatrick
Jeffrey Flanigan
Lucie Flekova
Michael Flor*
Radu Florian*
José A. R. Fonollosa
Mikel L. Forcada
George Foster
James Foulds
Anette Frank
Diego Frassinelli
Jesse Freitas
Lea Frermann
Annemarie Friedrich
Guohong Fu
Xiao Fu
Atsushi Fujii
Akinori Fujino
Atsushi Fujita
Sanae Fujita
Fumiyo Fukumoto
Richard Futrell
Robert Gaizauskas
Byron Galbraith
Michel Galley
Björn Gambäck
Michael Gamon
Kuzman Ganchev
Ashwinkumar Ganesan†
Debasis Ganguly
Yuze Gao†
Jesús Miguel García†
Marcos Garcia
Miguel A. García-Cumbreras
Claire Gardent
Matt Gardner
Dan Garrette
Guillermo Garrido
Justin Garten*
Milica Gašić
Tao Ge*
Kallirroi Georgila
Abbas Ghaddar†
Debanjan Ghosh*
Kevin Gimpel
Filip Ginter
Roxana Girju
James Glass
Michael Glass
Alfio Gliozzo†
Koldo Gojenola
Yoav Goldberg
Dan Goldwasser
Carlos Gómez-Rodríguez
Hugo Gonçalo Oliveira*
Hila Gonen
Samuel González-López
Kyle Gorman
Genevieve Gorrell
Isao Goto
Amit Goyal
Kartik Goyal
Pawan Goyal
Vishal Goyal
Thomas Graf
Yvette Graham
Floriana Grasso
Christophe Gravier
Nancy Green
Spence Green
Stephan Greene
Gregory Grefenstette
Eleni Gregoromichelaki
Ralph Grishman
Stig-Arne Grönroos†
Cyril Grouin
Jiatao Gu*
Anupam Guha
Curry Guinn
Hongyu Guo
Jiang Guo
Weiwei Guo
Xiaoxiao Guo
Yufan Guo
Nitish Gupta
Sonal Gupta
Suchin Gururangan†
Joakim Gustafson
Francisco Guzmán
Nizar Habash
Yaakov HaCohen-Kerner
Barry Haddow
Christian Hadiwinoto†
Matthias Hagen
Udo Hahn
Hannaneh Hajishirzi
Hazem Hajj
Kishaloy Halder
John Hale
David Hall
Keith Hall*
Michael Hammond
Thierry Hamon
Xianpei Han*
Mark Hansen
Jiangang Hao
Sanda Harabagiu
Christian Hardmeier
Sadid A. Hasan
Mohammed Hasanuzzaman
Mark Hasegawa-Johnson
Kazuma Hashimoto†
Eva Hasler
Hany Hassan
Helen Hastie
Bradley Hauer†
Catherine Havasi
Hangfeng He†
Hua He
Lihong He†
Wei He
Yifan He
Zhongjun He
Kenneth Heafield
Peter Heeman
Michael Heilman
Benjamin Heinzerling*
James Henderson
John Henderson
Matthew Henderson
James Hendler
Iris Hendrickx
Aurélie Herbelot
Ulf Hermjakob
Teresa Herrmann
Jonathan Herzig
Jack Hessel
Chris Hidey†
Felix Hieber
Ryuichiro Higashinaka
Derrick Higgins
Tsutomu Hirao
Keikichi Hirose
Graeme Hirst
Chris Hokamp†
Eric Holgate†
Nils Holzenberger†
Yu Hong
Enamul Hoque
Chiori Hori
Takaaki Hori

Veronique Hoste
Yufang Hou
Julian Hough
Dirk Hovy
Christine Howes
Shu-Kai Hsieh
Chun-Nan Hsu
Wen-Lian Hsu
Zhiting Hu
Xinyu Hua
Hen-Hsen Huang
Minlie Huang
Po-Sen Huang
Ruihong Huang
Shujian Huang
Xiaojiang Huang
Xiaolei Huang†
Xuanjing Huang
Zhongqiang Huang
Matthias Huck
Matt Huenerfauth
Mans Hulden
Rebecca Hwa
Mei-Yuh Hwang
Seung-won Hwang
Irina Illina†
Kenji Imamura
Kentaro Inui
Hitoshi Isahara
Alexei V. Ivanov
Mohit Iyyer
Sharmistha Jat†
Donghong JI
Yangfeng Ji
Di Jiang
Hui Jiang
Ridong Jiang
Xin Jiang
Charles Jochim†
Anders Johannsen
Richard Johansson
Michael Johnston
Kristiina Jokinen
Gareth Jones
Aditya Joshi
Mahesh Joshi
Shafiq Joty
Armand Joulin
Marcin Junczys-Dowmunt
David Jurgens
Kyo Kageura
Nobuhiro Kaji
Rasoul Kaljahi*
Hidetaka Kamigaito†
Hetunandan Kamisetty
Min-Yen Kan
Masahiro Kaneko†
Katharina Kann
Pinar Karagoz
Dimitri Kartsaklis
Lauri Karttunen
David Kauchak
Anna Kazantseva
Arefeh Kazemi
Mark Keane
Chris Kedzie
Andrew Kehler
Simon Keizer
John Kelleher
Ruth Kempson
Casey Kennington
Kian Kenyon-Dean
Kristian Kersting
Shahram Khadivi
Mitesh M. Khapra
Daniel Khashabi
Tushar Khot
Emre Kiciman
Douwe Kiela
Gunhee Kim
Jin-Dong Kim
Joo-Kyung Kim†
Jung-Jae Kim
Young-Bum Kim
Tracy Holloway King
Brian Kingsbury
Katrin Kirchhoff
Svetlana Kiritchenko
Julia Kiseleva
Dietrich Klakow
Guillaume Klein†
Jan Kleindienst
Alexandre Klementiev
Julien Kloetzer
Kevin Knight
Alistair Knott
Hayato Kobayashi
Philipp Koehn*
Rob Koeling
Jean-Pierre Koenig
Arne Köhn†
Varada Kolhatkar
Thomas Kollar
Alexander Koller
Mamoru Komachi
Kazunori Komatani
Alexandros Komninos†
Rik Koncel-Kedziorski
Grzegorz Kondrak
Lingpeng Kong
Ioannis Konstas
Mikhail Kopotev
Stefan Kopp
Valia Kordoni
Yannis Korkontzelos
Leila Kosseim
Katsunori Kotani
Mikhail Kozhevnikov
Emiel Krahmer
Martin Krallinger
Jayant Krishnamurthy
Vincent Kríž
Canasai Kruengkrai
Udo Kruschwitz
Germán Kruszewski
Lun-Wei Ku
Marco Kuhlmann
Roland Kuhn
Wu Kui†
Vivek Kulkarni
Shankar Kumar
Vaibhav Kumar
Jonathan Kummerfeld*
Gourab Kundu
Mikko Kurimo
Sadao Kurohashi
Robin Kurtz†
Mucahid Kutlu†
Tom Kwiatkowski
Oi Yee Kwong
Shibamouli Lahiri
Alice Lai
Wai Lam
Mathias Lambert
Patrik Lambert
Vasileios Lampos
Phillippe Langlais
Ni Lao
Guy Lapalme
Mirella Lapata
Shalom Lappin
Staffan Larsson
Jey Han Lau

Alberto Lavelli
Julia Lavid-López
Joseph Le Roux
Claudia Leacock
Chong Min Lee
Honglak Lee
Jihwan Lee†
John Lee
Kenton Lee
Moontae Lee
Sungjin Lee
Taesung Lee
Yoong Keok Lee
Young-Suk Lee
Fabrice Lefevre
Tao Lei
Alessandro Lenci
Chee Wee (Ben) Leong
Yves Lepage
James Lester
Johannes Leveling
Tomer Levinboim
Rivka Levitan
Sarah Ita Levitan
Gina-Anne Levow
Omer Levy
Mike Lewis
Binyang Li
Bofang Li†
Chen Li
Fangtao Li
Haizhou Li
Junhui Li
Lihong Li
Mu Li
Peifeng Li
Peng Li
Sheng Li
Sujian Li
Wenjie Li
Xiang Li*
Xiaoli Li
Xiaoqing Li†
Xiujun Li
Yanran Li
Yingyu Liang†
Constantine Lignos
Chu-Cheng Lin
Jimmy Lin
Junyang Lin†
Xiao Ling
Pierre Lison
Marina Litvak
Bing Liu*
Bing Liu
Can Liu
Changsong Liu
Jing Liu†
Kang Liu
Lemao Liu
Qi Liu†
Qun Liu
Shujie Liu
Ting Liu
Xiaodong Liu
Xueqing Liu†
Yang Liu
Karen Livescu
Elena Lloret*
Lajanugen Logeswaran†
Nikhil Londhe
José Lopes*
Oier Lopez de Lacalle
Adrian Pastor López Monroy
Aurelio López López
Anastassia Loukina
Di Lu*
Qin Lu
Wei Lu
Yi Luan
Nichola Lubold†
Michal Lukasik
Stephanie Lukin
Rebecca Lunsford
Weihua Luo
Minh-Thang Luong
Veronica Lynn†
Ji Ma
Mingbo Ma
Wei-Yun Ma
Xuezhe Ma
Andrew Maas
Brian Mac Namee
Klaus Macherey
Wolfgang Macherey
Nitin Madnani
Bernardo Magnini
Debanjan Mahata
Wolfgang Maier
Adam Makarucha
Nikolaos Malandrakis*
Andreas Maletti
Igor Malioutov
Fragkiskos Malliaros
Rob Malouf
Liliana Mamani Sanchez
Suresh Manandhar
Inderjeet Mani
Christopher D. Manning
Saab Mansour
Ramesh Manuvinakurike
Junhua Mao
Mitchell Marcus
Anna Margolis
Benjamin Marie
Erwin Marsi
David Martinez
Héctor Martínez Alonso*
André F. T. Martins
Bruno Martins
David Martins de Matos
Yuval Marton
Luis Marujo
Shigeki Matsubara
Yoichi Matsuyama
Yevgen Matusevych
Chandler May†
James Mayfield
Karen Mazidi
David McAllester
Andrew McCallum
Diana McCarthy
David McClosky
Bridget McInnes
Kathy McKeown
Brian McMahan
Louise McNally
Paul McNamee
Michael McTear
Julie Medero
Philipp Meerkamp
Oren Melamud
Gábor Melis
Arul Menezes
Fandong Meng
Helen Meng
Wolfgang Menzel
Angeliki Metallinou
Haitao Mi
Lisa Michaud
Stuart Middleton
Juliana Miehle†
Tomas Mikolov

Timothy Miller
Shachar Mirkin
Anand Mishra†
Teruko Mitamura
Jeff Mitchell*†
Prasenjit Mitra
Makoto Miwa
Daichi Mochihashi
Marie-Francine Moens
Abdelrahman Mohamed
Saif Mohammad*
Abidalrahman Moh'd
Behrang Mohit
Karo Moilanen
Diego Molla
Simonetta Montemagni
Andres Montoyo
Taesun Moon
Nafise Sadat Moosavi
Roser Morante
Véronique Moriceau
Emmanuel Morin
Hajime Morita
Alessandro Moschitti
Nasrin Mostafazadeh
Lili Mou
Danielle L Mowery
Matthew Mulholland*
Philippe Muller
Dragos Munteanu
Yugo Murawaki
Elena Musi
Seung-Hoon Na†
Seema Nagar
Masaaki Nagata
Ajay Nagesh
Saeed Najafi†
Satoshi Nakamura
Mikio Nakano
Yukiko Nakano
Ndapa Nakashole
Courtney Napoles
Jason Naradowsky
Karthik Narasimhan
Shrikanth Narayanan
Alexis Nasr
Vivi Nastase*
Roberto Navigli
Hamada Nayel
Claire Nédellec
Mark-Jan Nederhof
Arvind Neelakantan
Matteo Negri
Guenter Neumann
Dominick Ng
Hwee Tou Ng
Jun-Ping Ng
Vincent Ng
Axel-Cyrille Ngonga Ngomo
Dat Quoc Nguyen
Dong Nguyen*
Thien Huu Nguyen*
Viet-An Nguyen
Jian Ni
Garrett Nicolai
Massimo Nicosia
Vlad Niculae
Jian-Yun Nie
Jan Niehues
Rodney Nielsen
Takashi Ninomiya
Nobal Bikram Niraula
Hitoshi Nishikawa
Zheng-Yu Niu
Joakim Nivre
Chikashi Nobata
Hiroshi Noji
Pierre Nugues
Diarmuid Ó Séaghdha
Mick O'Donnell
Brendan O'Connor
Timothy O'Donnell
Stephan Oepen
Nir Ofek
Kemal Oflazer
Alice Oh
Jong-Hoon Oh
Kiyonori Ohtake
Naoaki Okazaki
Manabu Okumura
Hiroshi G. Okuno
Eda Okur†
Shereen Oraby*
Naoki Otani
Myle Ott
Jessica Ouyang†
Cecilia Ovesdotter Alm
Lilja Øvrelid
Gözde Özbal†
Ankur Padia†
Ulrike Pado
Sebastian Padó
Alexis Palmer
Alessio Palmero Aprosio
Sinno Jialin Pan
Alexander Panchenko†
Patrick Pantel
Aasish Pappu
Ivandré Paraboni
Ankur Parikh
Sunghyun Park†
Rebecca J. Passonneau
Siddharth Patwardhan
Michael J. Paul
Umashanthi Pavalanathan
Adam Pease
Pavel Pecina
Stephan Peitz
Jing Peng†
Xiaochang Peng
Gerald Penn
Alicia Pérez†
Laura Perez-Beltrachini
Verónica Pérez-Rosas
Bryan Perozzi
Matthew Peters
Slav Petrov
Volha Petukhova
Nghia The Pham
Maria Pia di Buono†
Scott Piao
Olivier Pietquin
Mohammad Taher Pilehvar
Yuval Pinter
Tommi Pirinen
Nikiforos Pittaras
Paul Piwek
Benjamin Piwowarski
Lahari Poddar†
Christian Pölitz
Massimo Poesio
Thierry Poibeau
Adam Poliak†
Heather Pon-Barry
Simone Paolo Ponzetto
Hoifung Poon
Ana-Maria Popescu
Marius Popescu
Andrei Popescu-Belis
Maja Popović
Fred Popowich
Rebecca Portnoff
Christopher Potts

Vinodkumar Prabhakaran*
Animesh Prasad
Adithya Pratapa†
Daniel Preoţiuc-Pietro
Patti Price
Emily Prud'hommeaux
Matthew Purver
Ashequl Qadir
Behrang QasemiZadeh
Tao Qin
Long Qiu
Minghui Qiu
Xipeng Qiu
Preethi Raghavan
Altaf Rahman
Rohan Ramanath
Carlos Ramisch
Nitin Ramrakhiyani†
Vivek Kumar Rangarajan Sridhar
Peter A. Rankel
Jinfeng Rao
Sudha Rao
Ari Rappoport
Mohammad Sadegh Rasooli
Antoine Raux
Sujith Ravi*
Kyle Rawlins
Manny Rayner
Paul Rayson
Marta Recasens
Siva Reddy
Sravana Reddy
Marek Rei
Ehud Reiter
David Reitter
Steve Renals
Ludovic Rheault
Giuseppe Riccardi
Matthew Richardson
Mark Riedl
Martin Riedl
Jason Riesa
Arndt Riester
German Rigau
Mattia Rigotti
Laura Rimell
Fabio Rinaldi
Eric Ringger
Michael Rist†
Alan Ritter
Brian Roark
Kirk Roberts
Melissa Roemmele
Marcus Rohrbach
Lina M. Rojas Barahona
Oleg Rokhlenko
Stephen Roller
Laurent Romary
Salvatore Romeo†
Marc-Antoine Rondeau
Eric Rosen
Andrew Rosenberg
Sophie Rosset*
Paolo Rosso
Benjamin Roth
Michael Roth
Johann Roturier
Aku Rouhe†
Salim Roukos
Bryan Routledge
Andrew Roxburgh†
Alla Rozovskaya
Nicholas Ruiz
Anna Rumshisky
Josef Ruppenhofer
Alexander Rush
Irene Russo
Attapol Rutherford
Kugatsu Sadamitsu
Markus Saers
Kenji Sagae
Benoît Sagot
Saurav Sahay
Hassan Sajjad
Keisuke Sakaguchi
Peña Saldarriaga†
Avneesh Saluja
Rajhans Samdani
Magali Sanches Duran
Enrico Santus
Murat Saraclar
Ruhi Sarikaya
Anoop Sarkar
Yutaka Sasaki
Ryohei Sasano
Prasanna Sattigeri
Nikunj Saunshi†
Christina Sauper
Asad Sayeed
Christian Scheible
Frank Schilder
Michael Schlichtkrull†
Natalie Schluter*
Allen Schmaltz
Helmut Schmid
Andrew Schneider†
Jodi Schneider
Alexandra Schofield
William Schuler
Sabine Schulte im Walde
Hannes Schulz
H. Andrew Schwartz
Lane Schwartz
Roy Schwartz*
Stephanie Schwartz
Holger Schwenk
Donia Scott
Pascale Sébillot
Natalia Segal†
Satoshi Sekine
Jean Senellart
Rico Sennrich
Chandramohan Senthilkumar
Minjoon Seo
Izhak Shafran
Cui Shaobo†
Vasu Sharma†
Serge Sharoff
Lanbo She*
Baoxu Shi
Shuming Shi
Yangyang Shi
Hiroyuki Shindo
Chaitanya Shivade
Ekaterina Shutova
Maryam Siahbani
Advaith Siddharthan
Carina Silberer
Miikka Silfverberg†
Fabrizio Silvestri
Michel Simard
Patrick Simianer†
Kiril Simov
Serra Sinem Tekiroglu†
Abhishek Singh†
Sameer Singh
Kairit Sirts
Sunayana Sitaram
Steve Skiena
Kevin Small
Ronnie Smith
Jan Šnajder
Artem Sokolov

Youngseo Son†
Hyun-Je Song
Linfeng Song
Radu Soricut
Michael Spranger
Richard Sproat
Vivek Srikumar
Somayajulu Sripada
Manfred Stede
Pontus Stenetorp
Amanda Stent*
Mark Stevenson
Brandon Stewart
Matthew Stone
Svetlana Stoyanchev
Veselin Stoyanov*
Carlo Strapparava
Karl Stratos
Kristina Striegnitz
Jannik Strötgen
Sara Stymne
Jinsong Su
Keh-Yih Su*
Katsuhito Sudoh
Elior Sulem†
Ming Sun
Xu Sun
Peter Sutton
Hisami Suzuki
Jun Suzuki
Yoshimi Suzuki
John Sylak-Glassman
György Szarvas
Stan Szpakowicz
Oscar Täckström
Hiroya Takamura
David Talbot
Partha Talukdar
Akihiro Tamura
Chenhao Tan
Takaaki Tanaka
Niket Tandon
Duyu Tang
Xavier Tannier
Yuka Tateisi
Rachael Tatman
Tatiane Tavares
Kapil Thadani
Mariët Theune
Paul Thompson
Sam Thomson
Christoph Tillmann
Ivan Titov
Erik Tjong Kim Sang
Takenobu Tokunaga
Marc Tomlinson
Sara Tonelli
Antonio Toral
Kentaro Torisawa
Marwan Torki
Samia Touileb
Trang Tran
David Traum
Adam Trischler†
Chen-Tse Tsai*
Richard Tzong-Han Tsai
Reut Tsarfaty
Ling Tsou†
Oren Tsur
Yoshimasa Tsuruoka
Yulia Tsvetkov
Zhaopeng Tu
Gökhan Tür*
Marco Turchi
Ferhan Ture
Martin Tutek†
Kiyotaka Uchimoto
Stefan Ultes
Lyle Ungar
Shyam Upadhyay
Masao Utiyama
Takehito Utsuro
Ozlem Uzuner
Naushad UzZaman
Raghuram Vadapalli
Alessandro Valitutti
Andreas van Cranenburgh
Rogier van Dalen
Tim Van de Cruys
Kees van Deemter
Lonneke van der Plas
Benjamin Van Durme
Emiel van Miltenburg
Rik van Noord†
Marten van Schijndel
Vincent Vandeghinste
David Vandyke
Vasudeva Varma
Sumithra Velupillai
Deepak Venugopal
Marc Verhagen
Yannick Versley
Karin Verspoor
Marc Vilain
David Vilar
David Vilares
Jesús Vilares
Martin Villalba
Veronika Vincze
Andreas Vlachos
Svitlana Volkova*
Clare Voss
Piek Vossen
Atro Voutilainen
Ivan Vulić
Yogarshi Vyas
V.G. Vinod Vydiswaran
Henning Wachsmuth
Joachim Wagner*
Byron Wallace
Matthew Walter
Xiaojun Wan
Baoxun Wang
Chang Wang
Chi Wang*
Jiang Wang
Jingjing Wang
Jinpeng Wang
Rui Wang
Tong Wang
Yanshan Wang
Yiou Wang
Yue Wang†
Zhiguo Wang
Zhongqing Wang
Leo Wanner
Nigel Ward
Shinji Watanabe
Taro Watanabe*
Christopher Waterson*
Bonnie Webber
Wouter Weerkamp
David Weir
Michael Wiegand
Rodrigo Wilkens*
Theresa Wilson
Shuly Wintner
Sam Wiseman
Guillaume Wisniewski
Travis Wolfe
Lawrence Wolf-Sonkin†
Kam-Fai Wong
Hua Wu

Jian Wu
Joern Wuebker†
Min Xiao
Shasha Xie
Wenduan Xu
Nianwen Xue
Adam Yala
Ikuya Yamada
Bishan Yang
Diyi Yang
Min Yang
Qian Yang
Yaqin Yang
Yi Yang
Yinfei Yang†
Roman Yangarber
Tae Yano
Jin-ge Yao
Mark Yatskar*
James Yifei Yang†
Wenpeng Yin*
Ding Ying
Dani Yogatama
Su-Youn Yoon
Naoki Yoshinaga
Dian Yu
Dianhai Yu
Jianfei Yu
Liang-Chih Yu
Mo Yu
François Yvon
Nasser Zalmout†
Roberto Zamparelli
Fabio Massimo Zanzotto
Klaus Zechner
Luke Zettlemoyer
Deniz Zeyrek
Feifei Zhai
Biao Zhang†
Boliang Zhang*
Dakun Zhang†
Hao Zhang
Jiajun Zhang
Jianwen Zhang
Lei Zhang
Longtu Zhang†
Meishan Zhang
Min Zhang
Qi Zhang
Sheng Zhang†
Tong Zhang
Wei Zhang
Wei Zhang
Yongfeng Zhang
Yue Zhang
Dongyan Zhao
Hai Zhao
Jun Zhao
Ran Zhao
Tiejun Zhao
Xiaoqing Zheng
Zilong Zheng†
Alisa Zhila
Dong Zhou
Jie Zhou
Junsheng Zhou
Nina Zhou
Xuanyu Zhou†
Muhua Zhu
Xiaodan Zhu
Andrea Zielinski
Heike Zinsmeister
Imed Zitouni
Michael Zock
Chengqing Zong
Bowei Zou
Arkaitz Zubiaga*
Ingrid Zukerman

Outstanding Paper Award Committee

Joyce Chai
Michael Collins
Jennifer Foster
Smaranda Muresan
Joel Tetreault (Chair)

Test-of-Time Paper Expert Award Committee

Marilyn Walker (Chair)
Antal van den Bosch
Chris Callison-Burch
Claire Cardie
Joyce Chai
Walter Daelemans
Katrin Erk
Pascale Fung
Iryna Gurevych
Katrin Kirchhoff
Yuji Matsumoto
Yusuke Miyao
Alessandro Moschitti
Massimo Poesio
Owen Rambow
Anoop Sarkar
Amanda Stent
Michael Strube
Lucy Vanderwende
Hua Wu
Chengqing Zong

Student Volunteers

Akhtar Shad
Amorim Evelin
Amplayo Reinald Kim
Basu Roy Chowdhury Somnath
Chatterjee Rajen
Chinnappa Dhivya
Choubey Prafulla
Cocos Anne
Dai Zeyu
Ezeani Ignatius
Fu Yao
Gella Spandana
Goo Chih-Wen
Gu Jiatao
Hao Shudong
Jiang Chao
Jiang Jyun-Yu
Jiang Youxuan
Juraska Juraj
Keith Katherine
Kenyon-Dean Kian
Kriz Reno
Kumar Vishwajeet
Maharjan Suraj
Mokhtari Shekoofeh
Nangia Nikita
Paula Felipe
Poddar Shivani
Rahgooy Taher
Schneider Andrew
Shen Yanyao
Srivastava Shashank
Strubell Emma
Su Shang-Yu
Szubert Ida
Tran Trang
Vempala Alakananda
Wiegreffe Sarah
Xiao Liqiang
Yasunaga Michihiro
Ye Hai
Yu Tao
Zhao Jieyu

Outstanding Papers

For NAACL HLT 2018 we recognize four outstanding research track papers (one of these will be named best paper). These four papers were selected by a committee composed of Joyce Chai (Michigan State University), Michael Collins (Columbia University), Jennifer Foster (Dublin City University), Smaranda Muresan (Columbia University) and Joel Tetreault (Grammarly; chair), all NAACL HLT 2018 area chairs with no conflicts with the candidate outstanding papers. The nine candidate papers were selected by the program chairs from nineteen papers nominated by the area chairs. These papers will be presented in a plenary session on the last day of the conference. Congratulations to the authors!

- *Deep Contextualized Word Representations*, by Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee and Luke Zettlemoyer
- *Learning to Map Context-Dependent Sentences to Executable Formal Queries*, by Alane Suhr, Srinivasan Iyer and Yoav Artzi
- *Neural Text Generation in Stories using Entity Representations as Context*, by Elizabeth Clark, Yangfeng Ji and Noah A. Smith
- *Recurrent Neural Networks as Weighted Language Recognizers*, by Yining Chen, Sorcha Gilroy, Andreas Maletti, Jonathan May and Kevin Knight

Test of Time Papers

For NAACL HLT 2018 we recognize three influential and inspiring Computational Linguistics (CL) papers which were published between 2002-2012 at the Association for Computational Linguistics (ACL) conferences (including ACL, NAACL, EACL, EMNLP and CONLL), workshops and journals (including TACL and CL), to recognize research that has had long-lasting influence until today, including positive impact on a subarea of CL, across subareas of CL, and outside of the CL research community. These papers may have proposed new research directions and new technologies, or released results and resources that have greatly benefit the community. Nineteen candidate test of time papers were nominated by our area chairs. Separate votes on these papers were held separately by two committees: an expert award committee consisting of all ACL and NAACL general chairs and program chairs and NAACL board members from 2013-2018 who did not have a conflict with the nominated papers, and a community award committee consisting of the 1000 authors who have published the most papers at ACL venues and who did not have a conflict with the nominated papers. These papers will be re-presented by the authors in a plenary session on the second day of the conference. Congratulations to the authors!

- *BLEU: a Method for Automatic Evaluation of Machine Translation*, by Kishore Papineni, Salim Roukos, Todd Ward and Wei-Jing Zhu
- *Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms*, by Michael Collins
- *Thumbs up?: Sentiment Classification using Machine Learning Techniques*, by Bo Pang, Lillian Lee and Shivakumar Vaithyanathan

Keynote Talk: Why 72?

Charles Yang
University of Pennsylvania

Biography
Charles is a Professor of Linguistics, Computer Science, and Psychology at the University of Pennsylvania and directs the Program in Cognitive Science. He has spent a long time to work out the tricks children use to learn languages and is now ready to try them out on machines. His most recent book, The Price of Linguistic Productivity, is the winner of the 2017 LSA Leonard Bloomfield award.

Keynote Talk: The Moment When the Future Fell Asleep

Kevin Knight
University of Southern California / Information Sciences Institute

Biography
Kevin is a professor of computer science at the University of Southern California and fellow of the Information Sciences Institute. He is a 2014 fellow of the ACL for foundational contributions to machine translation, to the application of automata for NLP, to decipherment of historical manuscripts, to semantics and to generation.

Keynote Talk: Google Assistant or My Assistant? Towards Personalized Situated Conversational Agents

Dilek Hakkani-Tür
Google Research

Abstract
Interacting with machines in natural language has been a holy grail since the beginning of computers. Given the difficulty of understanding natural language, only in the past couple of decades, we started seeing real user applications for targeted/limited domains. More recently, advances in deep learning based approaches enabled exciting new research frontiers for end-to-end goal-oriented conversational systems. However, personalization (i.e., learning to take actions from users and learning about users beyond memorizing simple attributes) remains a research challenge. In this talk, I'll review end-to-end situated dialogue systems research, with components for situated language understanding, dialogue state tracking, policy, and language generation. The talk will highlight novel approaches where dialogue is viewed as a collaborative game between a user and an agent in the presence of visual information. The situated conversational agent can be bootstrapped using user simulation (crawl), improved through interactions with crowd-workers (walk), and iteratively refined with real user interactions (run).

Biography
Dilek is a research scientist at Google Research Dialogue Group and has previously held positions at Microsoft Research, ICSI, and AT&T Labs – Research. She is a fellow of the IEEE and of ISCA. Her research interests include conversational AI, natural language and speech processing, spoken dialogue systems, and machine learning for language processing.

Table of Contents

Conference Program

June 2

07:30–08:45 Breakfast

08:45–09:00 Welcome from the Chairs

09:00–10:00 Keynote (sponsored by Toutiao AI Lab)

Why 72?
Charles Yang, University of Pennsylvania

10:00–10:30 Morning Coffee

10:30–11:30 Information Extraction 1

10:30–10:48 *Label-Aware Double Transfer Learning for Cross-Specialty Medical Named Entity Recognition*
Zhenghui Wang, Yanru Qu, Liheng Chen, Jian Shen, Weinan Zhang, Shaodian Zhang, Yimei Gao, Gen Gu, Ken Chen and Yong Yu

10:48–11:06 *Neural Fine-Grained Entity Type Classification with Hierarchy-Aware Loss*
Peng Xu and Denilson Barbosa

11:06–11:24 *Joint Bootstrapping Machines for High Confidence Relation Extraction*
Pankaj Gupta, Benjamin Roth and Hinrich Schütze

June 2 (continued)

10:30–11:30 Phonology, Morphology and Word Segmentation 1

10:30–10:48 *A Deep Generative Model of Vowel Formant Typology*
Ryan Cotterell and Jason Eisner

10:48–11:06 *Fortification of Neural Morphological Segmentation Models for Polysynthetic Minimal-Resource Languages*
Katharina Kann, Jesus Manuel Mager Hois, Ivan Vladimir Meza Ruiz and Hinrich Schütze

11:06–11:24 *Improving Character-Based Decoding Using Target-Side Morphological Information for Neural Machine Translation*
Peyman Passban, Qun Liu and Andy Way

10:30–11:30 Speech 1

10:30–10:48 *Parsing Speech: a Neural Approach to Integrating Lexical and Acoustic-Prosodic Information*
Trang Tran, Shubham Toshniwal, Mohit Bansal, Kevin Gimpel, Karen Livescu and Mari Ostendorf

10:48–11:06 *Tied Multitask Learning for Neural Speech Translation*
Antonios Anastasopoulos and David Chiang

11:06–11:24 *Please Clap: Modeling Applause in Campaign Speeches*
Jon Gillick and David Bamman

June 2 (continued)

10:30–12:00 Discourse and Pragmatics 1

Attentive Interaction Model: Modeling Changes in View in Argumentation
Yohan Jo, Shivani Poddar, Byungsoo Jeon, Qinlan Shen, Carolyn Rose and Graham Neubig

Automatic Focus Annotation: Bringing Formal Pragmatics Alive in Analyzing the Information Structure of Authentic Data
Ramon Ziai and Detmar Meurers

Dear Sir or Madam, May I Introduce the GYAFC Dataset: Corpus, Benchmarks and Metrics for Formality Style Transfer
Sudha Rao and Joel Tetreault

Improving Implicit Discourse Relation Classification by Modeling Inter-dependencies of Discourse Units in a Paragraph
Zeyu Dai and Ruihong Huang

10:30–12:00 Generation 1

A Deep Ensemble Model with Slot Alignment for Sequence-to-Sequence Natural Language Generation
Juraj Juraska, Panagiotis Karagiannis, Kevin Bowden and Marilyn Walker

A Melody-Conditioned Lyrics Language Model
Kento Watanabe, Yuichiroh Matsubayashi, Satoru Fukayama, Masataka Goto, Kentaro Inui and Tomoyasu Nakano

Discourse-Aware Neural Rewards for Coherent Text Generation
Antoine Bosselut, Asli Celikyilmaz, Xiaodong He, Jianfeng Gao, Po-Sen Huang and Yejin Choi

Natural Answer Generation with Heterogeneous Memory
Yao Fu and Yansong Feng

Query and Output: Generating Words by Querying Distributed Word Representations for Paraphrase Generation
Shuming Ma, Xu Sun, Wei Li, Sujian Li, Wenjie Li and Xuancheng Ren

Simplification Using Paraphrases and Context-Based Lexical Substitution
Reno Kriz, Eleni Miltsakaki, Marianna Apidianaki and Chris Callison-Burch

June 2 (continued)

Zero-Shot Question Generation from Knowledge Graphs for Unseen Predicates and Entity Types
Hady Elsahar, Christophe Gravier and Frederique Laforest

10:30–12:00 NLP Applications 1

Automated Essay Scoring in the Presence of Biased Ratings
Evelin Amorim, Marcia Cançado and Adriano Veloso

Content-Based Citation Recommendation
Chandra Bhagavatula, Sergey Feldman, Russell Power and Waleed Ammar

Looking Beyond the Surface: A Challenge Set for Reading Comprehension over Multiple Sentences
Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay and Dan Roth

Neural Automated Essay Scoring and Coherence Modeling for Adversarially Crafted Input
Youmna Farag, Helen Yannakoudakis and Ted Briscoe

QuickEdit: Editing Text & Translations by Crossing Words Out
David Grangier and Michael Auli

Tempo-Lexical Context Driven Word Embedding for Cross-Session Search Task Extraction
Procheta Sen, Debasis Ganguly and Gareth Jones

June 2 (continued)

11:30–12:30 Machine Learning 1

11:30–11:48 *Zero-Shot Sequence Labeling: Transferring Knowledge from Sentences to Tokens*
Marek Rei and Anders Søgaard

11:48–12:06 *Variable Typing: Assigning Meaning to Variables in Mathematical Text*
Yiannos Stathopoulos, Simon Baker, Marek Rei and Simone Teufel

11:30–12:30 Information Extraction 2

11:30–11:48 *Learning beyond Datasets: Knowledge Graph Augmented Neural Networks for Natural Language Processing*
Annervaz K M, Somnath Basu Roy Chowdhury and Ambedkar Dukkipati

11:48–12:06 *Comparing Constraints for Taxonomic Organization*
Anne Cocos, Marianna Apidianaki and Chris Callison-Burch

11:30–12:30 Machine Translation 1

11:30–11:48 *Improving Lexical Choice in Neural Machine Translation*
Toan Nguyen and David Chiang

11:48–12:06 *Universal Neural Machine Translation for Extremely Low Resource Languages*
Jiatao Gu, Hany Hassan, Jacob Devlin and Victor O.K. Li

12:06–12:24 *Classical Structured Prediction Losses for Sequence to Sequence Learning*
Sergey Edunov, Myle Ott, Michael Auli, David Grangier and Marc'Aurelio Ranzato

June 2 (continued)

12:30–14:00 Lunch

14:00–15:00 Industry Track Keynote

15:00–15:30 Afternoon Coffee

15:30–17:00 Machine Learning 2

15:30–15:48 *Deep Dirichlet Multinomial Regression*
Adrian Benton and Mark Dredze

15:30–17:00 Social Media and Computational Social Science 1

15:30–15:48 *Microblog Conversation Recommendation via Joint Modeling of Topics and Discourse*
Xingshan Zeng, Jing Li, Lu Wang, Nicholas Beauchamp, Sarah Shugars and Kam-Fai Wong

15:48–16:06 *Before Name-Calling: Dynamics and Triggers of Ad Hominem Fallacies in Web Argumentation*
Ivan Habernal, Henning Wachsmuth, Iryna Gurevych and Benno Stein

15:30–17:00 Vision, Robotics and Other Grounding 1

15:30–15:48 *Scene Graph Parsing as Dependency Parsing*
Yu-Siang Wang, Chenxi Liu, Xiaohui Zeng and Alan Yuille

15:48–16:06 *Learning Visually Grounded Sentence Representations*
Douwe Kiela, Alexis Conneau, Allan Jabri and Maximilian Nickel

16:06–16:24 *Comparatives, Quantifiers, Proportions: a Multi-Task Model for the Learning of Quantities from Vision*
Sandro Pezzelle, Ionut-Teodor Sorodoc and Raffaella Bernardi

16:24–16:42 *Being Negative but Constructively: Lessons Learnt from Creating Better Visual Question Answering Datasets*
Wei-Lun Chao, Hexiang Hu and Fei Sha

June 2 (continued)

15:30–17:00 Semantics 1

Abstract Meaning Representation for Paraphrase Detection
Fuad Issa, Marco Damonte, Shay B. Cohen, Xiaohui Yan and Yi Chang

attr2vec: Jointly Learning Word and Contextual Attribute Embeddings with Factorization Machines
Fabio Petroni, Vassilis Plachouras, Timothy Nugent and Jochen L. Leidner

Can Network Embedding of Distributional Thesaurus Be Combined with Word Vectors for Better Representation?
Abhik Jana and Pawan Goyal

Deep Neural Models of Semantic Shift
Alex Rosenfeld and Katrin Erk

Distributional Inclusion Vector Embedding for Unsupervised Hypernymy Detection
Haw-Shiuan Chang, Ziyun Wang, Luke Vilnis and Andrew McCallum

Mining Possessions: Existence, Type and Temporal Anchors
Dhivya Chinnappa and Eduardo Blanco

Neural Tensor Networks with Diagonal Slice Matrices
Takahiro Ishihara, Katsuhiko Hayashi, Hitoshi Manabe, Masashi Shimbo and Masaaki Nagata

Post-Specialisation: Retrofitting Vectors of Words Unseen in Lexical Resources
Ivan Vulić, Goran Glavaš, Nikola Mrkšić and Anna Korhonen

Unsupervised Learning of Sentence Embeddings Using Compositional n-Gram Features
Matteo Pagliardini, Prakhar Gupta and Martin Jaggi

June 2 (continued)

15:30–17:00 Sentiment Analysis 1

Learning Domain Representation for Multi-Domain Sentiment Classification
Qi Liu, Yue Zhang and Jiangming Liu

Learning Sentence Representations over Tree Structures for Target-Dependent Classification
Junwen Duan, Xiao Ding and Ting Liu

Relevant Emotion Ranking from Text Constrained with Emotion Relationships
Deyu Zhou, Yang Yang and Yulan He

Solving Data Sparsity for Aspect Based Sentiment Analysis Using Cross-Linguality and Multi-Linguality
Md Shad Akhtar, Palaash Sawant, Sukanta Sen, Asif Ekbal and Pushpak Bhattacharyya

SRL4ORL: Improving Opinion Role Labeling Using Multi-Task Learning with Semantic Role Labeling
Ana Marasović and Anette Frank

17:00–18:30 NLP Applications 2

17:00–17:18 *Approaching Neural Grammatical Error Correction as a Low-Resource Machine Translation Task*
Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha and Kenneth Heafield

17:18–17:36 *Robust Cross-Lingual Hypernymy Detection Using Dependency Context*
Shyam Upadhyay, Yogarshi Vyas, Marine Carpuat and Dan Roth

17:36–17:54 *Noising and Denoising Natural Language: Diverse Backtranslation for Grammar Correction*
Ziang Xie, Guillaume Genthial, Stanley Xie, Andrew Ng and Dan Jurafsky

June 2 (continued)

17:00–18:30 Question Answering 1

17:00–17:18 *Self-Training for Jointly Learning to Ask and Answer Questions*
Mrinmaya Sachan and Eric Xing

17:18–17:36 *The Web as a Knowledge-Base for Answering Complex Questions*
Alon Talmor and Jonathan Berant

17:36–17:54 *A Meaning-Based Statistical English Math Word Problem Solver*
Chao-Chun Liang, Yu-Shiang Wong, Yi-Chung Lin and Keh-Yih Su

17:00–18:30 SRW Highlights

June 3

07:45–08:45 Breakfast

08:45–09:00 Announcements

09:00–10:00 Keynote 2 (sponsored by Google)

The Moment When the Future Fell Asleep
Kevin Knight, University of Southern California / Information Sciences Institute

June 3 (continued)

10:00–10:30 Morning Coffee

10:30–11:30 Cognitive Modeling and Psycholinguistics 1

10:30–10:48 *Fine-Grained Temporal Orientation and its Relationship with Psycho-Demographic Correlates*
Sabyasachi Kamila, Mohammed Hasanuzzaman, Asif Ekbal, Pushpak Bhattacharyya and Andy Way

10:48–11:06 *Querying Word Embeddings for Similarity and Relatedness*
Fatemeh Torabi Asr, Robert Zinkov and Michael Jones

10:30–11:30 Summarization 1

10:30–10:48 *Semantic Structural Evaluation for Text Simplification*
Elior Sulem, Omri Abend and Ari Rappoport

10:48–11:06 *Entity Commonsense Representation for Neural Abstractive Summarization*
Reinald Kim Amplayo, Seonjae Lim and Seung-won Hwang

11:06–11:24 *Newsroom: A Dataset of 1.3 Million Summaries with Diverse Extractive Strategies*
Max Grusky, Mor Naaman and Yoav Artzi

10:30–11:30 Semantics 2

10:30–10:48 *Polyglot Semantic Parsing in APIs*
Kyle Richardson, Jonathan Berant and Jonas Kuhn

11:06–11:24 *Neural Models of Factuality*
Rachel Rudinger, Aaron Steven White and Benjamin Van Durme

June 3 (continued)

10:30–12:00 Information Extraction 3

Accurate Text-Enhanced Knowledge Graph Representation Learning
Bo An, Bo Chen, Xianpei Han and Le Sun

Acquisition of Phrase Correspondences Using Natural Deduction Proofs
Hitomi Yanaka, Koji Mineshima, Pascual Martínez-Gómez and Daisuke Bekki

Automatic Stance Detection Using End-to-End Memory Networks
Mitra Mohtarami, Ramy Baly, James Glass, Preslav Nakov, Lluís Màrquez and Alessandro Moschitti

Collective Entity Disambiguation with Structured Gradient Tree Boosting
Yi Yang, Ozan Irsoy and Kazi Shefaet Rahman

DeepAlignment: Unsupervised Ontology Matching with Refined Word Vectors
Prodromos Kolyvakis, Alexandros Kalousis and Dimitris Kiritsis

Efficient Sequence Learning with Group Recurrent Networks
Fei Gao, Lijun Wu, Li Zhao, Tao Qin, Xueqi Cheng and Tie-Yan Liu

FEVER: a Large-scale Dataset for Fact Extraction and VERification
James Thorne, Andreas Vlachos, Christos Christodoulopoulos and Arpit Mittal

Global Relation Embedding for Relation Extraction
Yu Su, Honglei Liu, Semih Yavuz, Izzeddin Gur, Huan Sun and Xifeng Yan

Implicit Argument Prediction with Event Knowledge
Pengxiang Cheng and Katrin Erk

Improving Temporal Relation Extraction with a Globally Acquired Statistical Resource
Qiang Ning, Hao Wu, Haoruo Peng and Dan Roth

Multimodal Named Entity Recognition for Short Social Media Posts
Seungwhan Moon, Leonardo Neves and Vitor Carvalho

June 3 (continued)

Nested Named Entity Recognition Revisited
Arzoo Katiyar and Claire Cardie

Simultaneously Self-Attending to All Mentions for Full-Abstract Biological Relation Extraction
Patrick Verga, Emma Strubell and Andrew McCallum

Supervised Open Information Extraction
Gabriel Stanovsky, Julian Michael, Luke Zettlemoyer and Ido Dagan

10:30–12:00 Tagging, Chunking, Syntax and Parsing 1

Embedding Syntax and Semantics of Prepositions via Tensor Decomposition
Hongyu Gong, Suma Bhat and Pramod Viswanath

From Phonology to Syntax: Unsupervised Linguistic Typology at Different Levels with Language Embeddings
Johannes Bjerva and Isabelle Augenstein

Monte Carlo Syntax Marginals for Exploring and Using Dependency Parses
Katherine Keith, Su Lin Blodgett and Brendan O'Connor

Neural Particle Smoothing for Sampling from Conditional Sequence Models
Chu-Cheng Lin and Jason Eisner

Neural Syntactic Generative Models with Exact Marginalization
Jan Buys and Phil Blunsom

Noise-Robust Morphological Disambiguation for Dialectal Arabic
Nasser Zalmout, Alexander Erdmann and Nizar Habash

Parsing Tweets into Universal Dependencies
Yijia Liu, Yi Zhu, Wanxiang Che, Bing Qin, Nathan Schneider and Noah A. Smith

Robust Multilingual Part-of-Speech Tagging via Adversarial Training
Michihiro Yasunaga, Jungo Kasai and Dragomir Radev

June 3 (continued)

Universal Dependency Parsing for Hindi-English Code-Switching
Irshad Bhat, Riyaz A. Bhat, Manish Shrivastava and Dipti Sharma

What's Going On in Neural Constituency Parsers? An Analysis
David Gaddy, Mitchell Stern and Dan Klein

11:30–12:30 Machine Learning 3

11:30–11:48 *Deep Generative Model for Joint Alignment and Word Representation*
Miguel Rios, Wilker Aziz and Khalil Simaan

12:06–12:24 *Learning Word Embeddings for Low-Resource Languages by PU Learning*
Chao Jiang, Hsiang-Fu Yu, Cho-Jui Hsieh and Kai-Wei Chang

11:30–12:30 Social Media and Computational Social Science 2

11:30–11:48 *Exploring the Role of Prior Beliefs for Argument Persuasion*
Esin Durmus and Claire Cardie

11:48–12:06 *Inducing a Lexicon of Abusive Words – a Feature-Based Approach*
Michael Wiegand, Josef Ruppenhofer, Anna Schmidt and Clayton Greenberg

12:06–12:24 *Author Commitment and Social Power: Automatic Belief Tagging to Infer the Social Context of Interactions*
Vinodkumar Prabhakaran, Premkumar Ganeshkumar and Owen Rambow

June 3 (continued)

11:30–12:30 Vision, Robotics and Other Grounding 2

12:30–14:00 Lunch

14:00–15:00 Industry Track Keynote

15:00–15:30 Afternoon Coffee

15:30–17:00 Text Mining 1

15:30–15:48 *Comparing Automatic and Human Evaluation of Local Explanations for Text Classification*
Dong Nguyen

15:48–16:06 *Deep Temporal-Recurrent-Replicated-Softmax for Topical Trends over Time*
Pankaj Gupta, Subburam Rajaram, Hinrich Schütze and Bernt Andrassy

16:06–16:24 *Lessons from the Bible on Modern Topics: Low-Resource Multilingual Topic Model Evaluation*
Shudong Hao, Jordan Boyd-Graber and Michael J. Paul

16:24–16:42 *Explainable Prediction of Medical Codes from Clinical Text*
James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun and Jacob Eisenstein

June 3 (continued)

15:30–17:00 Semantics 3

15:30–15:48 *A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference*
Adina Williams, Nikita Nangia and Samuel Bowman

15:48–16:06 *Filling Missing Paths: Modeling Co-occurrences of Word Pairs and Dependency Paths for Recognizing Lexical Semantic Relations*
Koki Washio and Tsuneaki Kato

16:06–16:24 *Specialising Word Vectors for Lexical Entailment*
Ivan Vulić and Nikola Mrkšić

16:24–16:42 *Cross-Lingual Abstract Meaning Representation Parsing*
Marco Damonte and Shay B. Cohen

15:30–17:00 Tagging, Chunking, Syntax and Parsing 2

15:30–15:48 *Sentences with Gapping: Parsing and Reconstructing Elided Predicates*
Sebastian Schuster, Joakim Nivre and Christopher D. Manning

15:48–16:06 *A Structured Syntax-Semantics Interface for English-AMR Alignment*
Ida Szubert, Adam Lopez and Nathan Schneider

16:06–16:24 *End-to-End Graph-Based TAG Parsing with Neural Networks*
Jungo Kasai, Robert Frank, Pauli Xu, William Merrill and Owen Rambow

16:24–16:42 *Colorless Green Recurrent Networks Dream Hierarchically*
Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen and Marco Baroni

June 3 (continued)

15:30–17:00 Machine Learning 4

Diverse Few-Shot Text Classification with Multiple Metrics
Mo Yu, Xiaoxiao Guo, Jinfeng Yi, Shiyu Chang, Saloni Potdar, Yu Cheng, Gerald Tesauro, Haoyu Wang and Bowen Zhou

Early Text Classification Using Multi-Resolution Concept Representations
Adrian Pastor López Monroy, Fabio A. González, Manuel Montes, Hugo Jair Escalante and Thamar Solorio

Multinomial Adversarial Networks for Multi-Domain Text Classification
Xilun Chen and Claire Cardie

Pivot Based Language Modeling for Improved Neural Domain Adaptation
Yftah Ziser and Roi Reichart

Reinforced Co-Training
Jiawei Wu, Lei Li and William Yang Wang

Tensor Product Generation Networks for Deep NLP Modeling
Qiuyuan Huang, Paul Smolensky, Xiaodong He, Li Deng and Dapeng Wu

The Context-Dependent Additive Recurrent Neural Net
Quan Hung Tran, Tuan Lai, Gholamreza Haffari, Ingrid Zukerman, Trung Bui and Hung Bui

June 3 (continued)

15:30–17:00 Machine Translation 2

Combining Character and Word Information in Neural Machine Translation Using a Multi-Level Attention
Huadong Chen, Shujian Huang, David Chiang, Xinyu Dai and Jiajun Chen

Dense Information Flow for Neural Machine Translation
Yanyao Shen, Xu Tan, Di He, Tao Qin and Tie-Yan Liu

Evaluating Discourse Phenomena in Neural Machine Translation
Rachel Bawden, Rico Sennrich, Alexandra Birch and Barry Haddow

Fast Lexically Constrained Decoding with Dynamic Beam Allocation for Neural Machine Translation
Matt Post and David Vilar

Guiding Neural Machine Translation with Retrieved Translation Pieces
Jingyi Zhang, Masao Utiyama, Eiichro Sumita, Graham Neubig and Satoshi Nakamura

Handling Homographs in Neural Machine Translation
Frederick Liu, Han Lu and Graham Neubig

Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets
Zhen Yang, Wei Chen, Feng Wang and Bo Xu

Neural Machine Translation for Bilingually Scarce Scenarios: a Deep Multi-Task Learning Approach
Poorya Zaremoodi and Gholamreza Haffari

Self-Attentive Residual Decoder for Neural Machine Translation
Lesly Miculicich Werlen, Nikolaos Pappas, Dhananjay Ram and Andrei Popescu-Belis

Target Foresight Based Attention for Neural Machine Translation
Xintong Li, Lemao Liu, Zhaopeng Tu, Shuming Shi and Max Meng

June 3 (continued)

15:30–17:00 Phonology, Morphology and Word Segmentation 2

Context Sensitive Neural Lemmatization with Lematus
Toms Bergmanis and Sharon Goldwater

Modeling Noisiness to Recognize Named Entities using Multitask Neural Networks on Social Media
Gustavo Aguilar, Adrian Pastor López Monroy, Fabio González and Thamar Solorio

Reusing Weights in Subword-Aware Neural Language Models
Zhenisbek Assylbekov and Rustem Takhanov

Simple Models for Word Formation in Slang
Vivek Kulkarni and William Yang Wang

Using Morphological Knowledge in Open-Vocabulary Neural Language Models
Austin Matthews, Graham Neubig and Chris Dyer

17:00–18:30 Test of Time Session (in honor of Aravind Joshi)

17:00–17:15 ***Awards and Remembrances***

17:15–17:40 *BLEU: a Method for Automatic Evaluation of Machine Translation (Test of Time)*
Kishore Papineni, Salim Roukos, Todd Ward and Wei-Jing Zhu, IBM Research

17:40–18:05 *Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms (Test of Time)*
Michael Collins, Columbia University

18:05–18:30 *Thumbs up?: Sentiment Classification using Machine Learning Techniques (Test of Time)*
Bo Pang, Lillian Lee, Shivakumar Vaithyanathan, Cornell University, IBM Research

June 4

07:45–08:45 Breakfast

08:45–09:00 Announcements

09:00–10:00 Keynote 3 (sponsored by Bloomberg)

Google Assistant or My Assistant? Towards Personalized Situated Conversational Agents
Dilek Hakkani-Tür

10:00–10:30 Morning Coffee

10:30–11:30 Information Extraction 4

10:30–10:48 *A Neural Layered Model for Nested Named Entity Recognition*
Meizhi Ju, Makoto Miwa and Sophia Ananiadou

10:48–11:06 *DR-BiLSTM: Dependent Reading Bidirectional LSTM for Natural Language Inference*
Reza Ghaeini, Sadid A. Hasan, Vivek Datla, Joey Liu, Kathy Lee, Ashequl Qadir, Yuan Ling, Aaditya Prakash, Xiaoli Fern and Oladimeji Farri

11:06–11:24 *KBGAN: Adversarial Learning for Knowledge Graph Embeddings*
Liwei Cai and William Yang Wang

June 4 (continued)

10:30–11:30 Semantics 4

10:30–15:48 *Multimodal Frame Identification with Multilingual Evaluation*
Teresa Botschen, Iryna Gurevych, Jan-Christoph Klie, Hatem Mousselly Sergieh and Stefan Roth

10:48–11:06 *Learning Joint Semantic Parsers from Disjoint Data*
Hao Peng, Sam Thomson, Swabha Swayamdipta and Noah A. Smith

11:06–11:24 *Identifying Semantic Divergences in Parallel Text without Annotations*
Yogarshi Vyas, Xing Niu and Marine Carpuat

10:30–11:30 Generation 2

10:30–10:48 *Bootstrapping Generators from Noisy Data*
Laura Perez-Beltrachini and Mirella Lapata

10:48–11:06 *SHAPED: Shared-Private Encoder-Decoder for Text Style Adaptation*
Ye Zhang, Nan Ding and Radu Soricut

11:06–11:24 *Generating Descriptions from Structured Data Using a Bifocal Attention Mechanism and Gated Orthogonalization*
Preksha Nema, Shreyas Shetty, Parag Jain, Anirban Laha, Karthik Sankaranarayanan and Mitesh M. Khapra

June 4 (continued)

10:30–12:00 Question Answering 2

CliCR: a Dataset of Clinical Case Reports for Machine Reading Comprehension
Simon Suster and Walter Daelemans

Learning to Collaborate for Question Answering and Asking
Duyu Tang, Nan Duan, Zhao Yan, Zhirui Zhang, Yibo Sun, Shujie Liu, Yuanhua Lv and Ming Zhou

Learning to Rank Question-Answer Pairs Using Hierarchical Recurrent Encoder with Latent Topic Clustering
Seunghyun Yoon, Joongbo Shin and Kyomin Jung

Supervised and Unsupervised Transfer Learning for Question Answering
Yu-An Chung, Hung-yi Lee and James Glass

Tracking State Changes in Procedural Text: a Challenge Dataset and Models for Process Paragraph Comprehension
Bhavana Dalvi, Lifu Huang, Niket Tandon, Wen-tau Yih and Peter Clark

10:30–12:00 Social Media and Computational Social Science 3

Combining Deep Learning and Topic Modeling for Review Understanding in Context-Aware Recommendation
Mingmin Jin, Xin Luo, Huiling Zhu and Hankz Hankui Zhuo

Deconfounded Lexicon Induction for Interpretable Social Science
Reid Pryzant, Kelly Shen, Dan Jurafsky and Stefan Wagner

Detecting Denial-of-Service Attacks from Social Media Text: Applying NLP to Computer Security
Nathanael Chambers, Ben Fry and James McMasters

The Importance of Calibration for Estimating Proportions from Annotations
Dallas Card and Noah A. Smith

June 4 (continued)

10:30–12:00 Summarization 2

A Dataset of Peer Reviews (PeerRead): Collection, Insights and NLP Applications
Dongyeop Kang, Waleed Ammar, Bhavana Dalvi, Madeleine van Zuylen, Sebastian Kohlmeier, Eduard Hovy and Roy Schwartz

Deep Communicating Agents for Abstractive Summarization
Asli Celikyilmaz, Antoine Bosselut, Xiaodong He and Yejin Choi

Encoding Conversation Context for Neural Keyphrase Extraction from Microblog Posts
Yingyi Zhang, Jing Li, Yan Song and Chengzhi Zhang

Estimating Summary Quality with Pairwise Preferences
Markus Zopf

Generating Topic-Oriented Summaries Using Neural Attention
Kundan Krishna and Balaji Vasan Srinivasan

Generative Bridging Network for Neural Sequence Prediction
Wenhu Chen, Guanlin Li, Shuo Ren, Shujie Liu, Zhirui Zhang, Mu Li and Ming Zhou

Higher-Order Syntactic Attention Network for Longer Sentence Compression
Hidetaka Kamigaito, Katsuhiko Hayashi, Tsutomu Hirao and Masaaki Nagata

Neural Storyline Extraction Model for Storyline Generation from News Articles
Deyu Zhou, Linsen Guo and Yulan He

Provable Fast Greedy Compressive Summarization with Any Monotone Submodular Function
Shinsaku Sakaue, Tsutomu Hirao, Masaaki Nishino and Masaaki Nagata

Ranking Sentences for Extractive Summarization with Reinforcement Learning
Shashi Narayan, Shay B. Cohen and Mirella Lapata

Relational Summarization for Corpus Analysis
Abram Handler and Brendan O'Connor

What's This Movie About? A Joint Neural Network Architecture for Movie Content Analysis
Philip John Gorinski and Mirella Lapata

Which Scores to Predict in Sentence Regression for Text Summarization?
Markus Zopf, Eneldo Loza Mencía and Johannes Fürnkranz

11:30–12:30 Dialogue and Interactive Systems 1

11:30–11:48 *A Hierarchical Latent Structure for Variational Conversation Modeling*
Yookoon Park, Jaemin Cho and Gunhee Kim

11:48–12:06 *Detecting Egregious Conversations between Customers and Virtual Agents*
Tommy Sandbank, Michal Shmueli-Scheuer, Jonathan Herzig, David Konopnicki, John Richards and David Piorkowski

12:06–12:24 *Learning to Disentangle Interleaved Conversational Threads with a Siamese Hierarchical Network and Similarity Ranking*
Jyun-Yu Jiang, Francine Chen, Yan-Ying Chen and Wei Wang

11:30–12:30 Information Extraction 5

11:30–11:48 *Variational Knowledge Graph Reasoning*
Wenhu Chen, Wenhan Xiong, Xifeng Yan and William Yang Wang

11:48–12:06 *Inducing Temporal Relations from Time Anchor Annotation*
Fei Cheng and Yusuke Miyao

12:06–12:24 *ELDEN: Improved Entity Linking Using Densified Knowledge Graphs*
Priya Radhakrishnan, Partha Talukdar and Vasudeva Varma

June 4 (continued)

11:30–12:30 Generation 3

11:30–11:48 *Interpretable Charge Predictions for Criminal Cases: Learning to Generate Court Views from Fact Descriptions*
Hai Ye, Xin Jiang, Zhunchen Luo and Wenhan Chao

11:48–12:06 *Delete, Retrieve, Generate: a Simple Approach to Sentiment and Style Transfer*
Juncen Li, Robin Jia, He He and Percy Liang

12:06–12:24 *Adversarial Example Generation with Syntactically Controlled Paraphrase Networks*
Mohit Iyyer, John Wieting, Kevin Gimpel and Luke Zettlemoyer

12:30–14:00 Lunch

13:00–14:00 *NAACL Business Meeting*
Julia Hockenmaier, University of Illinois at Urbana-Champaign

14:00–15:30 Sentiment Analysis 2

14:00–14:18 *Sentiment Analysis: It's Complicated!*
Kian Kenyon-Dean, Eisha Ahmed, Scott Fujimoto, Jeremy Georges-Filteau, Christopher Glasz, Barleen Kaur, Auguste Lalande, Shruti Bhanderi, Robert Belfer, Nirmal Kanagasabai, Roman Sarrazingendron, Rohit Verma and Derek Ruths

14:18–14:36 *Multi-Task Learning of Pairwise Sequence Classification Tasks over Disparate Label Spaces*
Isabelle Augenstein, Sebastian Ruder and Anders Søgaard

14:36–14:54 *Word Emotion Induction for Multiple Languages as a Deep Multi-Task Learning Problem*
Sven Buechel and Udo Hahn

14:54–15:12 *Human Needs Categorization of Affective Events Using Labeled and Unlabeled Data*
Haibo Ding and Ellen Riloff

June 4 (continued)

14:00–15:30 Discourse and Pragmatics 2

14:00–14:18 *The Argument Reasoning Comprehension Task: Identification and Reconstruction of Implicit Warrants*
Ivan Habernal, Henning Wachsmuth, Iryna Gurevych and Benno Stein

14:18–14:36 *Linguistic Cues to Deception and Perceived Deception in Interview Dialogues*
Sarah Ita Levitan, Angel Maredia and Julia Hirschberg

14:36–14:54 *Unified Pragmatic Models for Generating and Following Instructions*
Daniel Fried, Jacob Andreas and Dan Klein

14:54–15:12 *Hierarchical Structured Model for Fine-to-Coarse Manifesto Text Analysis*
Shivashankar Subramanian, Trevor Cohn and Timothy Baldwin

14:00–15:30 Tagging, Chunking, Syntax and Parsing 3

15:12–15:30 *Behavior Analysis of NLI Models: Uncovering the Influence of Three Factors on Robustness*
Ivan Sanchez, Jeff Mitchell and Sebastian Riedel

14:00–15:30 Cognitive Modeling and Psycholinguistics 2

Assessing Language Proficiency from Eye Movements in Reading
Yevgeni Berzak, Boris Katz and Roger Levy

Comparing Theories of Speaker Choice Using a Model of Classifier Production in Mandarin Chinese
Meilin Zhan and Roger Levy

Spotting Spurious Data with Neural Networks
Hadi Amiri, Timothy Miller and Guergana Savova

The Timing of Lexical Memory Retrievals in Language Production
Jeremy Cole and David Reitter

Unsupervised Induction of Linguistic Categories with Records of Reading, Speaking, and Writing
Maria Barrett, Ana Valeria Gonzalez-Garduño, Lea Frermann and Anders Søgaard

14:00–15:30 Dialogue and Interactive Systems 2

Challenging Reading Comprehension on Daily Conversation: Passage Completion on Multiparty Dialog
Kaixin Ma, Tomasz Jurczyk and Jinho D. Choi

Dialog Generation Using Multi-Turn Reasoning Neural Networks
Xianchao Wu, Ander Martinez and Momo Klyen

Dialogue Learning with Human Teaching and Feedback in End-to-End Trainable Task-Oriented Dialogue Systems
Bing Liu, Gokhan Tür, Dilek Hakkani-Tür, Pararth Shah and Larry Heck

LSDSCC: a Large Scale Domain-Specific Conversational Corpus for Response Generation with Diversity Oriented Evaluation Metrics
Zhen Xu, Nan Jiang, Bingquan Liu, Wenge Rong, Bowen Wu, Baoxun Wang, Zhuoran Wang and Xiaolong Wang

14:00–15:30 Text Mining 2

EMR Coding with Semi-Parametric Multi-Head Matching Networks
Anthony Rios and Ramakanth Kavuluru

Factors Influencing the Surprising Instability of Word Embeddings
Laura Wendlandt, Jonathan K. Kummerfeld and Rada Mihalcea

Mining Evidences for Concept Stock Recommendation
Qi Liu and Yue Zhang

June 4 (continued)

14:00–15:30 Speech 2

Binarized LSTM Language Model
Xuan Liu, Di Cao and Kai Yu

Conversational Memory Network for Emotion Recognition in Dyadic Dialogue Videos
Devamanyu Hazarika, Soujanya Poria, Amir Zadeh, Erik Cambria, Louis-Philippe Morency and Roger Zimmermann

How Time Matters: Learning Time-Decay Attention for Contextual Spoken Language Understanding in Dialogues
Shang-Yu Su, Pei-Chieh Yuan and Yun-Nung Chen

Towards Understanding Text Factors in Oral Reading
Anastassia Loukina, Van Rynald T. Liceralde and Beata Beigman Klebanov

14:00–15:30 Vision, Robotics and Other Grounding 3

Generating Bilingual Pragmatic Color References
Will Monroe, Jennifer Hu, Andrew Jong and Christopher Potts

Learning with Latent Language
Jacob Andreas, Dan Klein and Sergey Levine

Object Counts! Bringing Explicit Detections Back into Image Captioning
Josiah Wang, Pranava Swaroop Madhyastha and Lucia Specia

Quantifying the Visual Concreteness of Words and Topics in Multimodal Datasets
Jack Hessel, David Mimno and Lillian Lee

Speaker Naming in Movies
Mahmoud Azab, Mingzhe Wang, Max Smith, Noriyuki Kojima, Jia Deng and Rada Mihalcea

Stacking with Auxiliary Features for Visual Question Answering
Nazneen Fatema Rajani and Raymond Mooney

June 4 (continued)

15:30–16:00 Afternoon Coffee

17:00–18:15 Outstanding Paper Session (sponsored by Amazon)

17:00–17:18 *Deep Contextualized Word Representations*
Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee and Luke Zettlemoyer

17:18–17:36 *Learning to Map Context-Dependent Sentences to Executable Formal Queries*
Alane Suhr, Srinivasan Iyer and Yoav Artzi

17:36–17:54 *Neural Text Generation in Stories Using Entity Representations as Context*
Elizabeth Clark, Yangfeng Ji and Noah A. Smith

17:54–18:12 *Recurrent Neural Networks as Weighted Language Recognizers*
Yining Chen, Sorcha Gilroy, Andreas Maletti, Jonathan May and Kevin Knight

Label-aware Double Transfer Learning for Cross-Specialty Medical Named Entity Recognition

Zhenghui Wang[†], Yanru Qu[†], Liheng Chen[†], Jian Shen[†], Weinan Zhang[†*]
Shaodian Zhang[†‡], Yimei Gao[‡], Gen Gu[‡], Ken Chen[‡], Yong Yu[†]
[†]APEX Data and Knowledge Management Lab, Shanghai Jiao Tong University
[‡]Synyi LLC.
{felixwzh,wnzhang,shaodian}@apex.sjtu.edu.cn
chen.ken@synyi.com

Abstract

We study the problem of named entity recognition (NER) from electronic medical records, which is one of the most fundamental and critical problems for medical text mining. Medical records which are written by clinicians from different specialties usually contain quite different terminologies and writing styles. The difference of specialties and the cost of human annotation makes it particularly difficult to train a universal medical NER system. In this paper, we propose a label-aware double transfer learning framework (La-DTL) for cross-specialty NER, so that a medical NER system designed for one specialty could be conveniently applied to another one with minimal annotation efforts. The transferability is guaranteed by two components: (i) we propose label-aware MMD for feature representation transfer, and (ii) we perform parameter transfer with a theoretical upper bound which is also label aware. We conduct extensive experiments on 12 cross-specialty NER tasks. The experimental results demonstrate that La-DTL provides consistent accuracy improvement over strong baselines. Besides, the promising experimental results on non-medical NER scenarios indicate that La-DTL is potential to be seamlessly adapted to a wide range of NER tasks.

1 Introduction

The development of hospital information system and medical informatics drives the leverage of various medical data for a more efficient and intelligent medical care service. Among many kinds of medical data, electronic health records (EHRs) are one of the most valuable and informative data as they contain detailed information about the patients and the clinical practices. EHRs are essential to many intelligent clinical applications, such as hospital quality control and clinical decision support systems (Wu et al., 2015). Most of EHRs are recorded in an unstructured form, i.e., natural language. Hence, extracting structured information from EHRs using natural language processing (NLP), e.g., named entity recognition (NER) and entity linking, plays a fundamental role in medical informatics (Zhang and Elhadad, 2013). In this paper, we focus on medical NER from EHRs, which is a fundamental task and is widely studied in the research community (Nadeau and Sekine, 2007; Uzuner et al., 2011).

In practice, the difficulty of building a universally robust and high-performance medical NER system lies in the variety of medical terminologies and expressions among different departments of specialties and hospitals. However, building separate NER systems for so many specialties comes with a prohibitively high cost. The data privacy issue further discourages the sharing of the data across departments or hospitals, making it more difficult to train a canonical NER system to be applied everywhere. This raises a natural question: if we have sufficient annotated EHRs data in one *source* specialty, can we distill the knowledge and transfer it to help training models in a related *target* specialty with few annotations? By transferring the knowledge we can achieve higher performance in target specialties with lower annotation cost and bypass the data sharing concerns. This is commonly referred to as *transfer learning* (Pan and Yang, 2010).

Current state-of-the-art transfer learning methods for NER are mainly based on deep neural networks, which perform an end-to-end training to distill sequential dependency patterns in the natural language (Ma and Hovy, 2016; Lample et al., 2016). These transfer learning methods include (i) feature representation transfer (Peng and Dredze, 2017; Kulkarni et al., 2016), which normally lever-

*Weinan Zhang is the corresponding author.

Proceedings of NAACL-HLT 2018, pages 1–15
New Orleans, Louisiana, June 1 - 6, 2018.

ages deep neural networks to learn a close feature mapping between the source and target domains, and (ii) parameter transfer (Murthy et al., 2016; Yang et al., 2017), which performs parameter sharing or joint training to get the target-domain model parameters close to those of the source-domain model. To the best of our knowledge, there is no previous literature working on transfer learning for NER in the medical domain, or even in a larger scope, i.e., medical natural language processing.

In this paper, we propose a novel NER transfer learning framework, namely label-aware double transfer learning (La-DTL): (i) We leverage bidirectional long-short term memory (Bi-LSTM) network (Graves and Schmidhuber, 2005) to automatically learn the text representations, based on which we perform a label-aware feature representation transfer. We propose a variant of maximum mean discrepancy (MMD) (Gretton et al., 2012), namely label-aware MMD (La-MMD), to explicitly reduce the domain discrepancy of feature representations of tokens with the same label between two domains. (ii) Based on the learned feature representations from Bi-LSTM, two conditional random field (CRF) models are performed for sequence labeling for source and target domain separately, where parameter transfer learning is performed. Specifically, an upper bound of KL divergence between the source and target domain's CRF label distributions is added over the emission and transition matrices across the source and target CRF models to explore the shareable parts of the parameters. Both (i) and (ii) have a label-aware characteristic, which will be discussed later. We further argue that label-aware characteristic is crucial for transfer learning in sequence labeling problems, e.g., NER, because only when the corresponding labels are matched, can the "similar" contexts (i.e. feature representation) and model parameters be efficiently borrowed to improve the label prediction.

Extensive experiments are conducted on 12 cross-specialty medical NER tasks with real-world EHRs. The experimental results demonstrate that La-DTL provides consistent accuracy improvement over strong baselines, with overall 2.62% to 6.70% absolute F1-score improvement over the state-of-the-art methods. Besides, the promising experimental results on other two non-medical NER scenarios indicate that La-DTL has the potential to be seamlessly adapted to a wide range of NER tasks.

2 Related Works

Named Entity Recognition (NER) is fundamental in information extraction area which aims at automatic detection of named entities (e.g., person, organization, location and geo-political) in free text (Marrero et al., 2013). Many high-level applications such as entity linking (Moro et al., 2014) and knowledge graph construction (Hachey et al., 2011) could be built on top of an NER system. Traditional high-performance approaches include conditional random fields models (CRFs) (Lafferty et al., 2001), maximum entropy Markov models (MEMMs) (McCallum et al., 2000) and hidden Markov models (HMMs). Recently, many neural network-based models have been proposed (Collobert et al., 2011; Chiu and Nichols, 2016; Ma and Hovy, 2016; Lample et al., 2016), in which few feature engineering works are needed to train a high-performance NER system. The architecture of those neural network-based models are similar, where different neural networks (LSTMs, CNNs) at different levels (char- and word-level) are applied to learn feature representations, and on top of neural networks, a CRF model is employed to make label predictions.

Transfer Learning distills knowledge from a source domain to help create a high-performance learner for a target domain. Transfer learning algorithms are mainly categorized into three types, namely instance transfer, feature representation transfer and parameter transfer (Pan and Yang, 2010). Instance transfer normally samples or reweights source-domain samples to match the distribution of the target domain (Chen et al., 2011; Chu et al., 2013). Feature representation transfer typically learns a feature mapping which projects source and target domain data simultaneously onto a common feature space following similar distributions (Zhuang et al., 2015; Long et al., 2015; Shen et al., 2017). Parameter transfer normally involves a joint or constrained training for the models on source and target domains, usually introduce connections between source target parameters via sharing (Srivastava and Salakhutdinov, 2013), initialization (Perlich et al., 2014), or inter-model parameter penalty schemes (Zhang et al., 2016).

Transfer Learning for NER Training a high-performance NER system requires expensive and

time-consuming manually annotated data. But sufficient labeled data is critical for the generalization of an NER system, especially for neural network-based models. Thus, transfer learning for NER is a practically important problem. The first group of methods focuses on sharing model parameters but they differ in the training schemes. He and Sun (2017) proposed to train the parameter-shared model with source and target data jointly, while the learning rates for sentences from source domain are re-weighted by the similarity with target domain corpus. Yang et al. (2017) proposed a family of frameworks which share model parameters in hierarchical recurrent networks to handle cross-application, cross-lingual, and cross-domain transfer in sequence labeling tasks. Differently, Lee et al. (2017) first trained the model with source domain data and then fine-tuned the model with little annotated target domain data.

Domain adaptation method has been well studied in NER scenarios such as using distributed word representations (Kulkarni et al., 2016) and leveraging rule-based annotators (Chiticariu et al., 2010). Multi-task learning has also been studied to improve performance in multiple NER tasks by transferring meaningful knowledge from other tasks (Collobert et al., 2011; Peng and Dredze, 2016). To take the advantages of both domain adaptation and multi-task learning, Peng and Dredze (2017) proposed a multi-task domain adaptation model.

3 Preliminaries

This section briefly introduces bidirectional LSTM, conditional random field and maximum mean discrepancy, which are the building blocks of our transfer learning framework.

Bidirectional LSTM Recurrent neural networks (RNNs) are widely used in NLP tasks for their great capability to capture contextual information in sequence data. A widely used variant of RNNs is long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997), which incorporates input and forget gates to capture both long and short term dependencies. Furthermore, it will be beneficial if we process the sequence in not only a forward but also a backward way. Thus, bidirectional LSTM (Bi-LSTM) was employed in many previous works (Chiu and Nichols, 2016; Ma and Hovy, 2016; Lample et al., 2016) to capture bidirectional information in a sequence. More specifically, for token $\mathbf{x}_t$ (embedding vector) at timestep t in sequence $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$, the θ_b-parameterized Bi-LSTM recurrently updates hidden vectors $\mathbf{h}_t^{\rightarrow} = G^f_{\theta_b}(\mathbf{X}, \mathbf{h}_{t-1}^{\rightarrow})$ and $\mathbf{h}_t^{\leftarrow} = G^b_{\theta_b}(\mathbf{X}, \mathbf{h}_{t+1}^{\leftarrow})$ produced by a forward LSTM and a backward one, respectively. Then we concatenate $\mathbf{h}_t^{\rightarrow}$ and $\mathbf{h}_t^{\leftarrow}$ to $\mathbf{h}_t$ as the final hidden vector produced by Bi-LSTM:

$$\mathbf{h}_t = \mathbf{h}_t^{\rightarrow} \oplus \mathbf{h}_t^{\leftarrow}.$$

The representations learned from Bi-LSTM for sequence $\mathbf{X}$ is thus denoted as $\mathbf{H} = (\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n)$.

Conditional Random Field The goal of NER is to detect named entities in a sequence $\mathbf{X}$ by predicting a sequence of labels $\mathbf{y} = (y_1, y_2, ..., y_n)$. Conditional random field (CRF) is widely used to make joint labeling of the tokens in a sequence (Lafferty et al., 2001).

Recently, Lample et al. (2016) proposed to build a CRF layer on top of a Bi-LSTM so that the automatically learned feature representation $\mathbf{H} = (\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n)$ of the sequence can be directly fed into the CRF for sequence labeling. For a sequence of labels $\mathbf{y}$, given the hidden vector sequence $\mathbf{H}$, we define its θ_c-parametrized score function $s_{\theta_c}(\mathbf{H}, \mathbf{y})$ as:

$$s_{\theta_c}(\mathbf{H}, \mathbf{y}) = \sum_{i=1}^{n} \mathbf{E}_{i,y_i} + \sum_{i=1}^{n-1} \mathbf{A}_{y_i, y_{i+1}},$$

where $\mathbf{E}$ is the emission score matrix of size $n \times m$ (m is the number of unique labels), and is computed by $\mathbf{E} = \mathbf{HW}$ where $\mathbf{W}$ is the label emission parameter matrix; $\mathbf{A}$ is the label transition parameter matrix; thus $\theta_c = \{\mathbf{W}, \mathbf{A}\}$. We then define the conditional probability of label sequence $\mathbf{y}$ given $\mathbf{H}$ by a softmax over all possible label sequences in set $\mathcal{Y}(\mathbf{H})$ as:

$$\begin{aligned} p_{\theta_c}(\mathbf{y}|\mathbf{H}) &= \exp\{s_{\theta_c}(\mathbf{H}, \mathbf{y})\}/Z(\mathbf{H}) \\ &= \exp\{s_{\theta_c}(\mathbf{H}, \mathbf{y})\} \Big/ \sum_{\mathbf{y}' \in \mathcal{Y}(\mathbf{H})} \exp\{s_{\theta_c}(\mathbf{H}, \mathbf{y}')\}, \end{aligned} \quad (1)$$

where θ_c is omitted for simplification in the following part. The training objective in the CRF layer is to maximize the log-likelihood $\max_{\theta_c} \log p(\mathbf{y}|\mathbf{H})$. In the label prediction phase, we give the output label sequence $\mathbf{y}^*$ with the highest conditional probability $\mathbf{y}^* = \arg\max_{\mathbf{y}' \in \mathcal{Y}(\mathbf{H})} p(\mathbf{y}'|\mathbf{H})$ by dynamic programming (Sutton et al., 2012).

Maximum Mean Discrepancy Maximum Mean Discrepancy (Gretton et al., 2012) is a non-

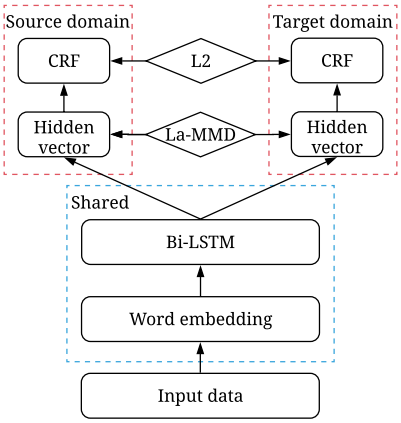


Figure 1: La-DTL framework overview: embedding and Bi-LSTM layers are shared across domains, predictors in red (upper) boxes are task-specific CRFs, with label-aware MMD and L2 constraints to perform feature representation transfer and parameter transfer.

parametric test statistic to measure the distribution discrepancy in terms of the distance between the kernel mean embeddings of two distributions p and q. The MMD is defined in particular function spaces that witness the difference in distributions

$$\text{MMD}(\mathcal{F}, p, q) = \sup_{f \in \mathcal{F}} (\mathbb{E}_{x \sim p}[f(x)] - \mathbb{E}_{y \sim q}[f(y)]).$$

By defining the function class $\mathcal{F}$ as the unit ball in a universal Reproducing Kernel Hilbert Space (RKHS), denoted by $\mathcal{H}$, it holds that $\text{MMD}[\mathcal{F}, p, q] = 0$ if and only if $p = q$. And then given two sets of samples $X = \{x_1, ..., x_m\}$ and $Y = \{y_1, ..., y_n\}$ independently and identically distributed (i.i.d.) from p and q on the data space $\mathcal{X}$, the empirical estimate of MMD can be written as the distance between the empirical mean embeddings after mapping to RKHS

$$\text{MMD}(X, Y) = \left\| \frac{1}{m} \sum_{i=1}^{m} \phi(x_i) - \frac{1}{n} \sum_{j=1}^{n} \phi(y_j) \right\|_{\mathcal{H}}, \quad (2)$$

where $\phi(\cdot) : \mathcal{X} \to \mathcal{H}$ is the nonlinear feature mapping that induces $\mathcal{H}$.

4 Methodology

In this section, we present a label-aware double transfer learning (La-DTL) framework and discuss its rationale.

4.1 Framework Overview

Figure 1 gives an overview of La-DTL for NER. From bottom up, each input sentence is converted into a sequence of embedding vectors, which are then fed into a Bi-LSTM to sequentially encode contextual information into fixed-length hidden vectors. The embedding and Bi-LSTM layers are shared among source/target domains. With label-aware maximum mean discrepancy (La-MMD) to reduce the feature representation discrepancy between two domains, the hidden vectors are directly fed into source/target domain specific CRF layers to predict the label sequence. We use domain constrained CRF layers to enhance the target domain performance.

More formally, let $\mathcal{D}_s = \{(\mathbf{X}_i^s, \mathbf{y}_i^s)\}_{i=1}^{N^s}$ be the training set of N^s samples from the source domain and $\mathcal{D}_t = \{(\mathbf{X}_i^t, \mathbf{y}_i^t)\}_{i=1}^{N^t}$ be the training set of N^t samples from the target domain, with $N^t \ll N^s$. Bi-LSTM encodes a sentence $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$ to hidden vectors $\mathbf{H} = (\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n)$. We occasionally use $\mathbf{H}(\mathbf{X})$ to denote the corresponding hidden vectors when feeding $\mathbf{X}$ into the Bi-LSTM. CRF decodes hidden vectors $\mathbf{H}$ to a label sequence $\hat{\mathbf{y}} = (\hat{y}_1, \hat{y}_2, ..., \hat{y}_n)$. Our goal is to improve label prediction accuracy on the target domain $\mathcal{D}_t$ by utilizing the knowledge from the source domain $\mathcal{D}_s$:

$$\begin{aligned} p(\mathbf{y}|\mathbf{X}) =& p(\mathbf{y}|\mathbf{H}(\mathbf{X})), \\ \log p(\mathbf{y}|\mathbf{H}) =& \sum_{i=1}^{n} \mathbf{E}_{i,y_i} + \sum_{i=1}^{n-1} \mathbf{A}_{y_i,y_{i+1}} - \log Z(\mathbf{H}). \end{aligned} \quad (3)$$

Thus training a transferable model $p(\mathbf{y}|\mathbf{X})$ requires both $\mathbf{H}(\mathbf{X})$ and $p(\mathbf{y}|\mathbf{H})$ to be transferable.

We use share word embedding and Bi-LSTM by approaching the feature representation distributions $p(\mathbf{h}|\mathcal{D}_s)$ and $p(\mathbf{h}|\mathcal{D}_t)$, i.e., the distributions of Bi-LSTM hidden vectors at each timestep of the sentences from the source and target domains respectively. The rationale behind it lies on the insufficiency of labeled target data. Even though LSTM has high capacity, its generalization ability highly relies on viewing "sufficient" data. Otherwise, LSTM is very likely to overfit the data. Training on both source and target data, the Bi-LSTM is expected to learn feature representations with high quality. Yosinski et al. (2014) provided a justification of this solution that sharing bottom layers is promising for transfer learning in practice.

With the sentences projected onto the same hidden space, the conditional distribution $p(\mathbf{h}^s|\mathcal{D}_s)$ and $p(\mathbf{h}^t|\mathcal{D}_t)$, however, may be distant because

LSTM hidden vectors contain contextual information which is different across domains. In order to reduce source/target discrepancy, we refine MMD (Gretton et al., 2012) with label constraints, i.e., label-aware MMD (La-MMD). Using La-MMD, the source/target hidden states are pushed to similar distributions to make the feature representation $\mathbf{H}(\mathbf{X})$ transfer feasible.

Based on the hidden vectors from Bi-LSTM, we adopt independent CRF layers for each domain. The rationale lies in the hypotheses that (i) the target domain predictor can better capture target data distribution which could be very unique; (ii) a good predictor trained on the source domain directly could be leveraged to assist the target domain predictor without directly borrowing the source domain training data to bypass the data privacy issue. With respect to the emission and transition score matrices $\sum \mathbf{E}_{i,y_i}$ and $\sum \mathbf{A}_{y_i,y_{i+1}}$, we adopt an upper bound between source/target domains, which helps the target domain predictor to be guided by the source domain predictor. Thus $p(\mathbf{y}|\mathbf{H})$ is also transferable.

There are also other transfer methods, including fine-tuning, sharing parameter directly (without constraints) (He and Sun, 2017; Lee et al., 2017; Yang et al., 2017), etc. However, simply sharing models may dismiss target specific instances.

4.2 Learning Objective

The learning objective is to minimize the following loss $\mathcal{L}$ with respect to parameters $\mathbf{\Theta} = \{\theta_b, \theta_c\}$:

$$\mathcal{L} = \mathcal{L}_c + \alpha\, \mathcal{L}_{\text{La-MMD}} + \beta\, \mathcal{L}_p + \gamma\, \mathcal{L}_r,$$

where $\mathcal{L}_c$ is the CRF loss, $\mathcal{L}_{\text{La-MMD}}$ is the La-MMD loss, $\mathcal{L}_p$ is the parameter similarity loss on CRF layers, and $\mathcal{L}_r$ is the regularization term, with α, β, γ as hyperparameters to balance loss terms.

The CRF loss is our ultimate objective predicting the label sequence given the input sentence, i.e., we minimize the negative log-likelihood of training samples from both source/target domains:

$$\mathcal{L}_c = -\frac{\varepsilon}{N^s}\sum_{i=1}^{N^s}\log p(\mathbf{y}_i^s|\mathbf{H}_i^s) - \frac{1-\varepsilon}{N^t}\sum_{i=1}^{N^t}\log p(\mathbf{y}_i^t|\mathbf{H}_i^t),$$

where $\mathbf{H}$ are hidden vectors obtained from Bi-LSTM, ε is the balance coefficient. The La-MMD loss $\mathcal{L}_{\text{La-MMD}}$ and parameter similarity loss $\mathcal{L}_p$ are discussed in Section 4.3 and 4.4, respectively. The regularization term is to generally control overfitting:

$$\mathcal{L}_r = \|\theta_b\|_2^2 + \|\theta_c\|_2^2.$$

We will provide the model convergence and hyperparameter study in Section 5.1.

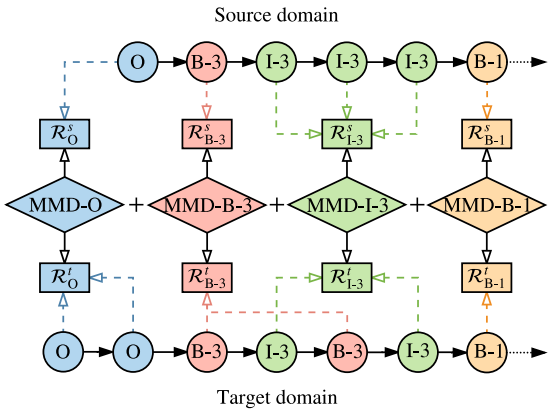


Figure 2: Illustration for La-MMD. MMD-y is computed between two domains' hidden representations with the same ground truth label y. A linear combination is then applied to each label-wise MMD to form La-MMD and the coefficient is set as $\mu_y = 1$.

4.3 Bi-LSTM Feature Representation Transfer

To learn transferable feature representations, the maximum mean discrepancy (MMD) which measures the distance between two distributions, has been widely used in domain adaptation scenarios (Long et al., 2015; Rozantsev et al., 2016). Almost all these works focus on reducing the *marginal* distribution distance between different domain features in an unsupervised manner to make them indistinguishable. However, considering a word is not evenly distributed conditioning on different labels, it may result in that the discriminative property of features from different domains may not be similar, which means that close source and target samples may not have the same label. Different from previous works, we propose label-aware MMD (La-MMD) in Eq. (5) to explicitly reduce the discrepancy between hidden representations with the same label, i.e., the linear combination of the MMD for each label. For each label class $y \in \mathcal{Y}_v$, where $\mathcal{Y}_v$ is the set of matched labels in two domains, we compute the squared population MMD between the hidden representations of source/target samples with the same label y:

$$\text{MMD}^2(\mathcal{R}_y^s, \mathcal{R}_y^t) = \frac{1}{(N_y^s)^2} \sum_{i,j=1}^{N_y^s} k(\mathbf{h}_i^s, \mathbf{h}_j^s) + \frac{1}{(N_y^t)^2} \sum_{i,j=1}^{N_y^t} k(\mathbf{h}_i^t, \mathbf{h}_j^t) - \frac{2}{N_y^s N_y^t} \sum_{i,j=1}^{N_y^s, N_y^t} k(\mathbf{h}_i^s, \mathbf{h}_j^t), \quad (4)$$

where $\mathcal{R}_y^s$ and $\mathcal{R}_y^t$ are sets of hidden representation $\mathbf{h}^s$ and $\mathbf{h}^t$ with corresponding number N_y^s and N_y^t. Eq. (4) can be easily derived by casting Eq. (2) into inner product form and applying $\langle \phi(x), \phi(y) \rangle_{\mathcal{H}} = k(x, y)$ where k is the reproducing kernel function (Gretton et al., 2012). For each label class, we compute the MMD loss in a normal manner. After that, we define the La-MMD loss as:

$$\mathcal{L}_{\text{La-MMD}} = \sum_{y \in \mathcal{Y}_v} \mu_y \cdot \text{MMD}^2(\mathcal{R}_y^s, \mathcal{R}_y^t), \quad (5)$$

where μ_y is the corresponding coefficient. The illustration of La-MMD is shown in Figure 2.

Once we have applied this La-MMD to our representations learned from Bi-LSTM, the representation distribution of instances with the same label from different domains should be close. Then the standard CRF layer which has a simple linear structure takes these similar representations as input and is likely to give a more transferable label decision for instances with the same label.

4.4 CRF Parameter Transfer

Simply sharing the CRF layer is non-promising when source/target data are diversely distributed. According to probability decomposition in Eq. (3), in order to transfer on source/target CRF layers, more specifically, $p(\mathbf{y}|\mathbf{H})$, we reduce the KL divergence from $p^t(\mathbf{y}|\mathbf{H})$ to $p^s(\mathbf{y}|\mathbf{H})$. But directly reducing $D_{\text{KL}}(p^s(\mathbf{y}|\mathbf{H})||p^t(\mathbf{y}|\mathbf{H}))$ is intractable, we tend to reduce its upper bound:

$$\begin{aligned} & D_{\text{KL}}(p^s(\mathbf{y}|\mathbf{H})||p^t(\mathbf{y}|\mathbf{H})) \\ =& \sum_{\mathbf{y} \in \mathcal{Y}(\mathbf{H})} p^s(\mathbf{y}|\mathbf{H}) \log(\frac{p^s(\mathbf{y}|\mathbf{H})}{p^t(\mathbf{y}|\mathbf{H})}) \\ =& -H(p^s(\mathbf{y}|\mathbf{H})) - \sum_{\mathbf{y} \in \mathcal{Y}(\mathbf{H})} p^s(\mathbf{y}|\mathbf{H}) \log p^t(\mathbf{y}|\mathbf{H}) \\ \leq& c(\|\mathbf{W}^s - \mathbf{W}^t\|_2^2 + \|\mathbf{A}^s - \mathbf{A}^t\|_2^2)^{\frac{1}{2}}, \end{aligned} \quad (6)$$

where $H(\cdot)$ is the entropy of distribution $(\cdot)$ and c is a constant. The detailed proof is provided in Appendix A.1. Since $c(\|\mathbf{W}^s - \mathbf{W}^t\|_2^2 + \|\mathbf{A}^s - \mathbf{A}^t\|_2^2)$ is the upper bound of $D_{\text{KL}}(p^s(\mathbf{y}|\mathbf{H})\|p^t(\mathbf{y}|\mathbf{H}))$,

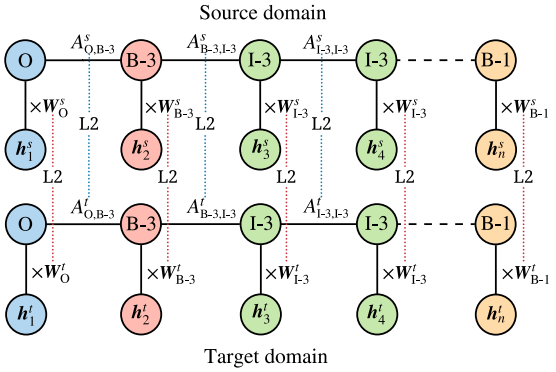


Figure 3: Illustration for CRF parameter transfer.

we conduct CRF parameter transfer by minimizing

$$\mathcal{L}_p = \|\mathbf{W}^s - \mathbf{W}^t\|_2^2 + \|\mathbf{A}^s - \mathbf{A}^t\|_2^2.$$

It turns out that a similar regularization term is applied in our CRF parameter transfer method and the regularization framework (RF) for domain adaptation (Lu et al., 2016). However, RF is proposed to generalize the feature augmentation method in (Daume III, 2007), and these two methods are only discussed from a perspective of the parameter. There is no guarantee that two models having similar parameters yields similar output distributions. In this work, we discuss the model behavior in CRF conditions, and we successfully prove that two CRF models having similar parameters (in Euclidean space) yields similar output distributions. In another word, our method guarantees transferability in the model behavior level, while previous works are limited in parameter level.

The CRF parameter transfer is illustrated in Figure 3, which is also label-aware since the L2 constraint is added over parameters corresponding to the same label in two domains, e.g., $\mathbf{W}_O^s$ and $\mathbf{W}_O^t$.

4.5 Training

We train La-DTL in an end-to-end manner with mini-batch AdaGrad (Duchi et al., 2011). One mini-batch contains training samples from both domains, otherwise the computation of $\mathcal{L}_{\text{La-MMD}}$ can not be performed. During training, word (and character) embeddings are fine-tuned to adjust real data distribution. During both training and decoding (testing) of CRF layers, we use dynamic programming to compute the normalizer in Eq. (1) and infer the label sequence.

Department	# Train	# Dev	# Test
Cardiology	3,004	601	601
Respiratory	3,025	605	606
Neurology	932	187	187
Gastroenterology	1,517	303	304
Sum	8,478	1,696	1,698

Table 1: Sentence numbers for *CM-NER* corpus.

5 Experiments

In this section, we evaluate La-DTL[1] and other baseline methods on 12 cross-specialty NER problems based on real-world datasets. The experimental results show that La-DTL steadily outperforms other baseline models in all tasks significantly. We also conduct further ablation study and robustness study. We evaluate La-DTL on two more non-medical NER transfer tasks to validate its general efficacy over a wide range of applications.

5.1 Cross-Specialty NER

Datasets We collected a Chinese medical NER (*CM-NER*) corpus for our experiments. This corpus contains 1600 de-identified EHRs of our affiliated hospital from four different specialties in four departments: Cardiology (500), Respiratory (500), Neurology (300) and Gastroenterology (300), and the research had been reviewed and approved by the ethics committee. Named entities are annotated in the BIOES format (Begin, Inside, Outside, End and Single), with 30 types in total. The statistics of *CM-NER* is shown in Table 1.

Baselines The following methods are compared. For a fair comparison, we implement La-DTL and baselines with the same base model introduced in (Lample et al., 2016) but with different transfer techniques.

- **Non-transfer** uses the target domain labeled data only.
- **Domain mask** and **Linear projection** belong to the same framework proposed by Peng and Dredze (2017) but have different implementations at the projection layer, which aims to produce shared feature representations among different domains through a linear transformation.
- **Re-training** is proposed by Lee et al. (2017), where an artificial neural networks (ANNs) is first trained on the source domain and then re-trained on the target domain.
- **Joint-training** is a transfer learning method proposed by Yang et al. (2017) where different tasks are trained jointly.
- **CD-learning** is a cross-domain learning method proposed by He and Sun (2017), where each source domain training example's learning rate is re-weighted.

Experimental Settings We use 23,217 unlabeled clinical records to train the word embeddings (word2vec) at 128 dimensions using skip-gram model (Mikolov et al., 2013). The hidden state size is set to be 200 for word-level Bi-LSTM. We evaluate La-DTL for cross-specialty NER with *CM-NER* in 12 transfer tasks, results shown in Table 2. For each task, we take the whole source domain training set $\mathcal{D}_s$ and 10% sentences of the target domain training set $\mathcal{D}_t$ as training data. We use the development set in target domain to search hyper-parameters including training epochs. We then take the models to make the prediction in target domain test set and use F1-score as the evaluation metric. Statistical significance has been determined using a randomization version of the paired sample t-test (Cohen, 1995).

Results and Discussion From the results of 12 cross-specialty NER tasks shown in Table 2, we find that La-DTL outperforms all the strong baselines in all the 12 cross-specialty transfer learning tasks, with 2.62% to 6.70% F1-score lift over state-of-the-art baseline methods. Meanwhile, Linear projection and Domain mask (Peng and Dredze, 2017) do not perform as good as other three baselines, which may be because such linear transformation methods are likely to weaken the representations. While other three baseline methods all share the whole model between source/target domains but differ in the training schemes and performance.

To better understand the transferability of La-DTL, we also evaluate three variants of La-DTL: La-MMD, CRF-L2, and MMD-CRF-L2. La-MMD and CRF-L2 have the same networks and loss function as La-DTL but with different building blocks: La-MMD has $\beta = 0$, while CRF-L2 has $\alpha = 0$. In MMD-CRF-L2, we replace La-MMD loss $\mathcal{L}_{\text{La-MMD}}$ in La-DTL with a vanilla MMD loss:

$$\mathcal{L}_{\text{MMD}} = \text{MMD}^2(\mathcal{R}^s, \mathcal{R}^t),$$

[1] https://github.com/felixwzh/La-DTL

Method	C→R	C→N	C→G	R→C	R→N	R→G	N→C	N→R	N→G	G→C	G→R	G→N	AVG
Non-transfer	67.20	54.51	49.01	65.63	54.51	49.01	65.63	67.20	49.01	65.63	67.20	54.51	59.09
Linear projection (Peng and Dredze, 2017)	69.01	67.02	57.40	69.79	65.87	57.71	67.70	68.77	51.33	68.00	69.65	61.12	64.45
Domain mask (Peng and Dredze, 2017)	70.76	63.97	58.62	70.18	64.27	58.16	67.93	69.89	56.18	68.87	69.89	63.49	65.18
CD-learning (He and Sun, 2017)	71.38	64.01	56.72	72.17	64.91	58.14	68.99	71.13	56.27	70.17	71.76	62.06	65.64
Re-training (Lee et al., 2017)	72.45	70.55	59.58	72.56	68.59	60.94	69.60	70.08	56.58	70.14	71.90	66.01	67.42
Joint-training (Yang et al., 2017)	69.82	70.49	63.52	71.45	67.03	67.71	70.96	71.43	60.54	69.68	71.55	68.15	68.53
La-MMD	73.08	69.48	59.86	72.53	70.28	60.16	71.31	73.04	57.94	69.80	73.99	67.19	68.22
CRF-L2	73.34	71.52	60.17	72.43	69.72	67.61	69.76	71.54	59.96	69.75	71.82	67.30	68.74
MMD-CRF-L2	73.05	72.35	60.80	72.65	69.87	66.82	70.25	71.75	58.98	70.48	73.98	67.43	69.03
La-DTL	**73.59**†	**72.91**†	**64.60**†	**73.88**†	**73.01**†	**70.17**†	**73.08**†	**73.11**†	**62.14**†	**71.61**†	**74.21**†	**71.49**†	**71.15**

Table 2: Results (F1-score %) of 12 cross-specialty medical NER tasks. C, R, N, G are short for the department of Cardiology, Respiratory, Neurology, and Gastroenterology, respectively. † indicates La-DTL outperforms the 6 baselines significantly ($p < 0.05$).

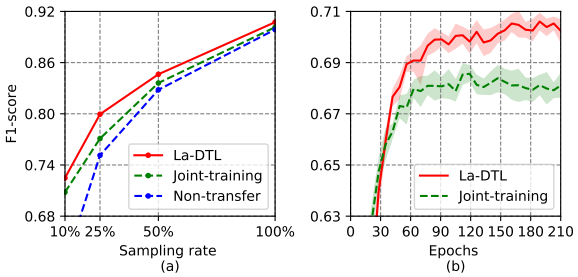


Figure 4: (a) F1-score of La-DTL, Joint-training and Non-transfer method in C→R task with different sampling rate. (b) The learning curve of La-DTL and Joint-training in C→R task.

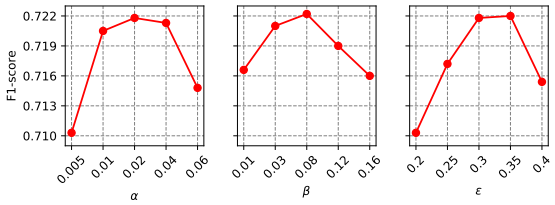


Figure 5: Hyperparameter study for α, β, and ε.

where $\mathcal{R}^s$ and $\mathcal{R}^t$ are sets of hidden representation from source and target domain. Results in Table 2 show that: (i) Using La-MMD alone does achieve satisfactory performance since it outperforms the best baseline Joint-training (Yang et al., 2017) in 7 of 12 tasks. And it has a significant improvement over Domain mask and Linear projection methods (Peng and Dredze, 2017), which indicates that using La-MMD to reduce the domain discrepancy of feature representations in sequence tagging tasks is promising. (ii) CRF-L2 is also a promising method when transferring between NER tasks, and it improves the La-MMD method significantly when these two methods are combined to form La-DTL. (iii) Label-aware characteristic is important in sequence labeling problems because there is an obvious performance drop when La-MMD is replaced with a vanilla MMD in La-DTL. But MMD-CRF-L2 still has very competitive performance compared to all the baseline methods. This shows positive empirical evidence that transferring knowledge at both Bi-LSTM feature representation level and CRF parameter level for NER tasks is better than transferring knowledge at only one of these two levels, as discussed in Section 4.1.

Robustness to Target Domain Data Sparsity We further study the sparsity problem (target domain) of La-DTL in C→R task comparing to Joint-training (Yang et al., 2017) and Non-transfer method. We evaluate La-DTL with different data volume (sampling rate: 10%, 25%, 50%, 100%) on the target domain training set. Results are shown in Figure 4(a). We observe that La-DTL outperforms Joint-training and Non-transfer results under all circumstances, and the improvement of La-DTL is more significant when the sampling rate is lower.

To show La-DTL's convergence and significant improvement over Joint-training, we repeat the 10% sampling rate experiment for 10 times with 10 random seeds. The F1-score on the target domain development set for two methods with a 95% confidence interval is shown in Figure 4(b) where La-DTL outperforms Joint-training method significantly.

Hyperparameter Study We study the influence of three key hyperparameters in La-DTL: α, β, and ε in C→R task with 10% target domain sampling rate. We first apply a rough grid search for the three hyperparameters, and the result is ($\alpha = 0.02, \beta = 0.03, \varepsilon = 0.3$). We then fix two hyperparameters and test the third one in a finer granularity. The results in Figure 5 indicate that setting $\alpha \in [0.01, 0.04]$ could better leverage La-MMD and further setting $\beta \in [0.03, 0.12]$ and $\varepsilon \in [0.3, 0.4]$ yields the best empirical perfor-

Corpus	# Train	# Dev	# Test
SighanNER	23,182	-	4,636
WeiboNER	1,350	270	270
CoNLL 2003	14,987	3,466	3,684
TwitterNER	1,900	240	254

Table 3: Sentence numbers for non-medical corpora.

Method	F1-score
Non-transfer	54.78
Linear projection (Peng and Dredze, 2017)*	56.40
Linear projection (Peng and Dredze, 2017)	56.99
Domain mask (Peng and Dredze, 2017)*	56.80
Domain mask (Peng and Dredze, 2017)	56.32
CD-learning (He and Sun, 2017)*	52.05
CD-learning (He and Sun, 2017)	56.46
Re-training (Lee et al., 2017)	55.36
Joint-training (Yang et al., 2017)	56.80
La-DTL	**57.74**

Table 4: Results (F1-score %) of *WeiboNER* transfer. $*$ indicates the result reported in the corresponding reference.

mance. This shows that we need to balance the learning objective of the source and target domains for better transferability.

5.2 NER Transfer Experiment on Non-medical Corpus

To show La-DTL could be applied in a wide range of NER transfer learning scenarios, we make experiments on two non-medical NER tasks. Corpora's details are shown in Table 3.

***WeiboNER* Transfer** Following He and Sun (2017); Peng and Dredze (2017), we transfer knowledge from *SighanNER* (MSR corpus of the sixth SIGHAN Workshop on Chinese language processing) to *WeiboNER* (a social media NER corpus) (Peng and Dredze, 2015). Results in Table 4 show that La-DTL outperforms all the baseline methods in Chinese social media domain.

***TwitterNER* Transfer** Following Yang et al. (2017) we transfer knowledge from CoNLL 2003 English NER (Tjong Kim Sang and De Meulder, 2003) to *TwitterNER* (Ritter et al., 2011). Since the entity types in these two corpora cannot be exactly matched, La-DTL and Joint-training (Yang et al., 2017) can be applied directly in this case while other baselines can not. Because the CRF parameter transfer of La-DTL is label-aware, and Joint-training simply leverages two independent CRF layers. The results are shown in Table 5, where La-DTL again outperforms Joint-training, indicating that La-DTL could be applied seamlessly to transfer learning scenarios with mismatched label sets and languages like English.

Method	F1-score
Non-transfer	34.65
Joint-training (Yang et al., 2017)*	43.24
La-DTL	**45.71**

Table 5: Results (F1-score %) of *TwitterNER* transfer. $*$ indicates the result reported in the corresponding reference.

6 Conclusions

In this paper, we propose La-DTL, a label-aware double transfer learning framework, to conduct both Bi-LSTM feature representation transfer and CRF parameter transfer with label-aware constraints for cross-specialty medical NER tasks. To our best knowledge, this is the first work on transfer learning for medical NER in cross-specialty scenario. Experiments on 12 cross-specialty NER tasks show that La-DTL provides consistent performance improvement over strong baselines. We further perform a set of experiments on different target domain data size, hyperparameter study and other non-medical NER tasks, where La-DTL shows great robustness and wide efficacy. For future work, we plan to jointly perform NER and entity linking for better cross-specialty media structural information extraction.

Acknowledgments

The work done by SJTU is sponsored by Synyi-SJTU Innovation Program, National Natural Science Foundation of China (61632017, 61702327, 61772333) and Shanghai Sailing Program (17YF1428200).

References

Minmin Chen, Kilian Q Weinberger, and John Blitzer. 2011. Co-training for domain adaptation. In *Advances in Neural Information Processing Systems 24*, pages 2456–2464. Curran Associates, Inc.

Laura Chiticariu, Rajasekar Krishnamurthy, Yunyao Li, Frederick Reiss, and Shivakumar Vaithyanathan. 2010. Domain adaptation of rule-based annotators for named-entity recognition tasks. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1002–1012, Cambridge, MA. Association for Computational Linguistics.

Jason Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Wen-Sheng Chu, Fernando De la Torre, and Jeffery F Cohn. 2013. Selective transfer machine for personalized facial action unit detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3515–3522.

Paul R Cohen. 1995. *Empirical methods for artificial intelligence*, volume 139. MIT press Cambridge, MA.

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

Hal Daume III. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263. Association for Computational Linguistics.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. A kernel two-sample test. *J. Mach. Learn. Res.*, 13:723–773.

Ben Hachey, Will Radford, and James R. Curran. 2011. Graph-based named entity linking with wikipedia. In *Proceedings of the 12th International Conference on Web Information System Engineering*, WISE'11, pages 213–226, Berlin, Heidelberg. Springer-Verlag.

Hangfeng He and Xu Sun. 2017. A unified model for cross-domain and semi-supervised named entity recognition in chinese social media. In *AAAI*, pages 3216–3222.

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

Vivek Kulkarni, Yashar Mehdad, and Troy Chevalier. 2016. Domain adaptation for named entity recognition in online media with word embeddings. *arXiv preprint arXiv:1612.00148*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Ji Young Lee, Franck Dernoncourt, and Peter Szolovits. 2017. Transfer learning for named-entity recognition with neural networks. *arXiv preprint arXiv:1705.06273*.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. 2015. Learning transferable features with deep adaptation networks. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 97–105, Lille, France. PMLR.

Wei Lu, Hai Leong Chieu, and Jonathan Löfgren. 2016. A general regularization framework for domain adaptation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 950–954, Austin, Texas. Association for Computational Linguistics.

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

Mnica Marrero, Julin Urbano, Sonia Snchez-Cuadrado, Jorge Morato, and Juan Miguel Gmez-Berbs. 2013. Named entity recognition: Fallacies, challenges and opportunities. *Computer Standards & Interfaces*, 35(5):482 – 489.

Andrew McCallum, Dayne Freitag, and Fernando C. N. Pereira. 2000. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 591–598, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, 2:231–244.

V Murthy, Mitesh Khapra, Pushpak Bhattacharyya, et al. 2016. Sharing network parameters for crosslingual named entity recognition. *arXiv preprint arXiv:1607.00198*.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Trans. on Knowl. and Data Eng.*, 22(10):1345–1359.

Nanyun Peng and Mark Dredze. 2015. Named entity recognition for chinese social media with jointly trained embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 548–554, Lisbon, Portugal. Association for Computational Linguistics.

Nanyun Peng and Mark Dredze. 2016. Improving named entity recognition for chinese social media with word segmentation representation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 149–155, Berlin, Germany. Association for Computational Linguistics.

Nanyun Peng and Mark Dredze. 2017. Multi-task domain adaptation for sequence tagging. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 91–100, Vancouver, Canada. Association for Computational Linguistics.

Claudia Perlich, Brian Dalessandro, Troy Raeder, Ori Stitelman, and Foster Provost. 2014. Machine learning for targeted display advertising: Transfer learning in action. *Mach. Learn.*, 95(1):103–127.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: An experimental study. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Artem Rozantsev, Mathieu Salzmann, and Pascal Fua. 2016. Beyond sharing weights for deep domain adaptation. *arXiv preprint arXiv:1603.06432*.

Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. 2017. Wasserstein distance guided representation learning for domain adaptation. *arXiv preprint arXiv:1707.01217*.

Nitish Srivastava and Ruslan R Salakhutdinov. 2013. Discriminative transfer learning with tree-based priors. In *Advances in Neural Information Processing Systems 26*, pages 2094–2102. Curran Associates, Inc.

Charles Sutton, Andrew McCallum, et al. 2012. An introduction to conditional random fields. *Foundations and Trends® in Machine Learning*, 4(4):267–373.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Özlem Uzuner, Brett R South, Shuying Shen, and Scott L DuVall. 2011. 2010 i2b2/va challenge on concepts, assertions, and relations in clinical text. *Journal of the American Medical Informatics Association*, 18(5):552–556.

Yonghui Wu, Min Jiang, Jianbo Lei, and Hua Xu. 2015. Named entity recognition in chinese clinical text using deep neural network. *Studies in health technology and informatics*, 216:624.

Zhilin Yang, Ruslan Salakhutdinov, and William W Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. In *ICLR*.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3320–3328, Cambridge, MA, USA. MIT Press.

Shaodian Zhang and Noémie Elhadad. 2013. Unsupervised biomedical named entity recognition: Experiments with clinical and biological texts. *Journal of biomedical informatics*, 46(6):1088–1098.

Weinan Zhang, Ulrich Paquet, and Katja Hofmann. 2016. Collective noise contrastive estimation for policy transfer learning. In *AAAI*, pages 1408–1414.

Fuzhen Zhuang, Xiaohu Cheng, Ping Luo, Sinno Jialin Pan, and Qing He. 2015. Supervised representation learning: Transfer learning with deep autoencoders. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 4119–4125. AAAI Press.

A Appendix

A.1 Detailed Proof

Recall the bound as in Eq. (6):

Lemma A.1. *$c_1(\|\mathbf{W}^s - \mathbf{W}^t\|_2^2 + \|\mathbf{A}^s - \mathbf{A}^t\|_2^2)$ is the upper bound of $(s^s(\mathbf{H}, \mathbf{y}) - s^t(\mathbf{H}, \mathbf{y}))^2$.*

Proof of Lemma A.1. $\otimes$ refers to convolutional product, $\mathbf{H}^W, \mathbf{H}^A$ are mask matrices corresponding to the given hidden vectors $\mathbf{H}$, and c_1 is a constant. We have:

$$
\begin{aligned}
&(s^s(\mathbf{H}, \mathbf{y}) - s^t(\mathbf{H}, \mathbf{y}))^2 \\
=&(\sum_{i=1}^{n} \mathbf{E}^s_{i,y_i} + \sum_{i=1}^{n-1} \mathbf{A}^s_{y_i,y_{i+1}} - \sum_{i=1}^{n} \mathbf{E}^t_{i,y_i} - \sum_{i=1}^{n-1} \mathbf{A}^t_{y_i,y_{i+1}})^2 \\
=&(\mathbf{W}^s \otimes \mathbf{H}^W + \mathbf{A}^s \otimes \mathbf{H}^A - \mathbf{W}^t \otimes \mathbf{H}^W - \mathbf{A}^t \otimes \mathbf{H}^A)^2 \\
=&((\mathbf{W}^s - \mathbf{W}^t) \otimes \mathbf{H}^W + (\mathbf{A}^s - \mathbf{A}^t) \otimes \mathbf{H}^A)^2 \\
\leq&2((\mathbf{W}^s - \mathbf{W}^t) \otimes \mathbf{H}^W)^2 + 2((\mathbf{A}^s - \mathbf{A}^t) \otimes \mathbf{H}^A)^2 \\
=&2(\sum_{i,j} (\mathbf{W}^s - \mathbf{W}^t)_{i,j} \cdot \mathbf{H}^W_{i,j})^2 + 2(\sum_{p,q} (\mathbf{A}^s - \mathbf{A}^t)_{p,q} \cdot \mathbf{H}^A_{p,q})^2 \\
\leq&2(\sum_{i,j} (\mathbf{W}^s - \mathbf{W}^t)^2_{i,j} \cdot \sum_{i,j} (\mathbf{H}^W_{i,j})^2) + 2(\sum_{p,q} (\mathbf{A}^s - \mathbf{A}^t)^2_{p,q} \cdot \sum_{p,q} (\mathbf{H}^A_{p,q})^2) \\
=&2(\|\mathbf{W}^s - \mathbf{W}^t\|_2^2 \cdot \|\mathbf{H}^W\|_2^2) + 2(\|\mathbf{A}^s - \mathbf{A}^t\|_2^2 \cdot \|\mathbf{H}^A\|_2^2) \\
\leq&c_1(\|\mathbf{W}^s - \mathbf{W}^t\|_2^2 + \|\mathbf{A}^s - \mathbf{A}^t\|_2^2).
\end{aligned}
$$

□

Lemma A.2. *$c(\|\mathbf{W}^s - \mathbf{W}^t\|_2^2 + \|\mathbf{A}^s - \mathbf{A}^t\|_2^2)^{\frac{1}{2}}$ is the upper bound of $D_{KL}(p^s(\mathbf{y}|\mathbf{H})||p^t(\mathbf{y}|\mathbf{H}))$.*

Proof of Lemma A.2. With Lemma. (A.1), we set $\varepsilon = (c_1(\|\mathbf{W}^s - \mathbf{W}^t\|_2^2 + \|\mathbf{A}^s - \mathbf{A}^t\|_2^2))^{\frac{1}{2}} \geq 0$ and $c = 2c_1^{\frac{1}{2}}$, and we have:

$$
s^s(\mathbf{H}, \mathbf{y}) - \varepsilon \leq s^t(\mathbf{H}, \mathbf{y}) \leq s^s(\mathbf{H}, \mathbf{y}) + \varepsilon, \tag{7}
$$

$$
\log\{\sum_{\mathbf{y}' \in \mathcal{Y}(\mathbf{H})} \exp[s^s(\mathbf{H}, \mathbf{y}')]\} - \varepsilon \leq \log\{\sum_{\mathbf{y}' \in \mathcal{Y}(\mathbf{H})} \exp[s^t(\mathbf{H}, \mathbf{y}')]\} \leq \log\{\sum_{\mathbf{y}' \in \mathcal{Y}(\mathbf{H})} \exp[s^s(\mathbf{H}, \mathbf{y}')]\} + \varepsilon. \tag{8}
$$

With Eq. (7) and Eq. (8), we can derive

$$
\begin{aligned}
& -\sum_{\mathbf{y}\in\mathcal{Y}(\mathbf{H})} p^s(\mathbf{y}|\mathbf{H})\log p^t(\mathbf{y}|\mathbf{H}) \\
=& -\sum_{\mathbf{y}\in\mathcal{Y}(\mathbf{H})} p^s(\mathbf{y}|\mathbf{H})\log\frac{\exp[s^t(\mathbf{H},\mathbf{y})]}{\sum_{\mathbf{y}'\in\mathcal{Y}(\mathbf{H})}\exp[s^t(\mathbf{H},\mathbf{y}')]} \\
=& -\sum_{\mathbf{y}\in\mathcal{Y}(\mathbf{H})} p^s(\mathbf{y}|\mathbf{H})\big\{s^t(\mathbf{H},\mathbf{y})-\log\{\sum_{\mathbf{y}'\in\mathcal{Y}(\mathbf{H})}\exp[s^t(\mathbf{H},\mathbf{y}')]\}\big\} \\
\leq& -\sum_{\mathbf{y}\in\mathcal{Y}(\mathbf{H})} p^s(\mathbf{y}|\mathbf{H})\big\{s^s(\mathbf{H},\mathbf{y})-\varepsilon-\log\{\sum_{\mathbf{y}'\in\mathcal{Y}(\mathbf{H})}\exp[s^s(\mathbf{H},\mathbf{y}')]\}-\varepsilon\big\} \\
=& -\sum_{\mathbf{y}\in\mathcal{Y}(\mathbf{H})} p^s(\mathbf{y}|\mathbf{H})\big\{\log\frac{\exp[s^s(\mathbf{H},\mathbf{y})]}{\sum_{\mathbf{y}'\in\mathcal{Y}(\mathbf{H})}\exp[s^s(\mathbf{H},\mathbf{y}')]}-2\varepsilon\big\} \\
=& -\sum_{\mathbf{y}\in\mathcal{Y}(\mathbf{H})} p^s(\mathbf{y}|\mathbf{H})\big\{\log p^s(\mathbf{y}|\mathbf{H})-2\varepsilon\big\} \\
=& H(p^s(\mathbf{y}|\mathbf{H}))+2\varepsilon.
\end{aligned}
$$

Finally, we have

$$
\begin{aligned}
& D_{\mathrm{KL}}(p^s(\mathbf{y}|\mathbf{H})||p^t(\mathbf{y}|\mathbf{H})) \\
=& \sum_{\mathbf{y}\in\mathcal{Y}(\mathbf{H})} p^s(\mathbf{y}|\mathbf{H})\log(\frac{p^s(\mathbf{y}|\mathbf{H})}{p^t(\mathbf{y}|\mathbf{H})}) \\
=& -H(p^s(\mathbf{y}|\mathbf{H}))-\sum_{\mathbf{y}\in\mathcal{Y}(\mathbf{H})} p^s(\mathbf{y}|\mathbf{H})\log p^t(\mathbf{y}|\mathbf{H}) \\
\leq& -H(p^s(\mathbf{y}|\mathbf{H}))+H(p^s(\mathbf{y}|\mathbf{H}))+2\varepsilon \\
=& c(\|\mathbf{W}^s-\mathbf{W}^t\|_2^2+\|\mathbf{A}^s-\mathbf{A}^t\|_2^2)^{\frac{1}{2}}.
\end{aligned}
$$

□

A.2 Case Analysis

In clinical practice, patients with specific diseases would be assigned to different departments, and specialist doctors in their department may pay more attention to the specific disease. When writing a medical chart, these specific diseases and related clinical findings would have a more detailed description. Therefore, some medical terms would have enriched meanings in different departments accordingly. For example, patients with rheumatic heart disease are often treated in the department of Cardiology. The term, "rheumatic", a modifier, describes and limits the type of "heart disease". In English, "rheumatic" is an adjective modifying "heart disease". However, in Chinese, "rheumatic heart disease" can be regarded as two diseases, "rheumatism" and "heart disease". In the department of Cardiology, "rheumatic heart disease" is usually mentioned as a single term. While in other departments, "rheumatism" and "heart disease" are mostly two independent named entities in annotated datasets. As such, it is difficult to train an NER model to capture the relationship between "rheumatism" and "heart disease", and band them as a whole. In the training set of our study, the diagnostic term "rheumatic heart disease" (including synonym) is mentioned for 17 times in Dept. Cardiology, 16 times in Dept. Respiratory, none in Dept. Neurology and 3 times in Dept. Gastroenterology. We use the data from the first 3 departments as source domain training set respectively, and the data from Dept. Gastroenterology as the target domain training set. We test our models on the test set from Dept. Gastroenterology, where "rheumatic heart disease" is mentioned 3 times, and compare the results across models

Disease	Transfer Task	# disease term in source domain training set	# disease term in target domain training set	# disease term in target domain test set	# accurate labeling without transfer	# accurate labeling with transfer
rheumatic heart disease	C→G N→G R→G	17 0 16	0	3	0	3 0 3
pulmonary heart disease	C→G N→G R→G	4 0 24	0	2	0	2 0 2
coronary atherosclerotic heart disease	G→N C→N R→N	5 136 23	0	15	10	3 15 11

Table 6: Case analysis for cross-specialty medical NER tasks. C, R, N, G are short for department of Cardiology, Respiratory, Neurology, and Gastroenterology, respectively.

with/without transfer learning. As expected, models with source training data from Dept. Cardiovascular and Respiration correctly predict all these entities, but the model using source data from Dept. Neurology fails and so does a model without transfer learning.

Patients with pulmonary heart disease were often referred to Dept. Respiratory and Dept. Cardiology. In our training set, "pulmonary heart disease" (including synonym) is labeled for 24 times in Dept. Respiratory and 4 times in Dept. Cardiology. In English, "pulmonary" modified "heart disease". In Chinese, "pulmonary heart disease" contains body structure "lung" and disease name "heart disease". The model trained with the source set from both from department of respiratory and cardiology could correctly recognize the relation between lung and heart disease and predict the entity in the test set from Dept. Gastroenterology.

Similarly, "coronary atherosclerotic heart disease" contains two disease names, "coronary atherosclerosis" and "heart disease". Training model using source set from a department where the terms are enriched could improve the performance of recognizing the whole entity.

A.3 Medical Experiments Details

The 30 entity types for medical domain are: Symptom, Disease, Examination, Treatment, Laboratory index, Products, Body structure, Frequency, Negative word, Value, Trend, Modification, Temporal word, Noun of locality, Degree modifier, Probability, Object, Organism, Location, Person, Pronoun, Privacy information, Accident, Action, Header, Instrument and material, Non-physiological structure, Dosage, Scale, and Preposition.

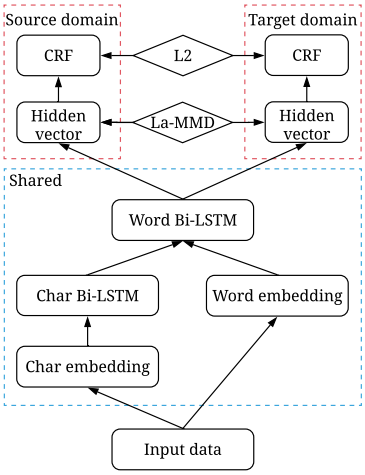


Figure 6: La-DTL framework for language like English.

A.4 Non-medical Experiments Details

WeiboNER Transfer

Both *SighanNER* and *WeiboNER* are annotated in the BIO format (Begin, Inside and Outside), but there is one more entity type (geo-political) in *WeiboNER*. For a fair comparison, we follow Peng and Dredze (2017); He and Sun (2017) to merge geo-political entities and locations in *WeiboNER*, to match different labeling schemes between *WeiboNER* and *SighanNER*. We use the inconsistencies fixed second version of *WeiboNER* data and word embeddings provided by *WeiboNER*'s developers (Peng and Dredze, 2015)[2] in this experiment.

TwitterNER Transfer

To show that La-DTL could be applied in transfer learning for NER scenario with mismatched

[2] https://github.com/hltcoe/golden-horse

named entity types and languages like English, we conduct this experiment transfer from CoNLL 2003 English NER to *TwitterNER*. The four entity types in CoNLL 2003 English NER are LOC, PER, ORG, and MISC. The ten entity types in *TwitterNER* are company, facility, geo-loc, movie, musicartist, other, person, product, sportsteam, and tvshow.

The Joint-training method (Yang et al., 2017) separates the CRF layers for each domain to bypass the label mismatch problem. Since our La-DTL is label-aware, we match four pairs of named entities between two CoNLL 2003 English NER and *TwitterNER*: LOC with geo-loc, PER with person, ORG with company and MISC with other to compute $\mathcal{L}_{\text{La-MMD}}$ and $\mathcal{L}_p$, and leave six named entities unmatched. Following Yang et al. (2017), We leverage char-level Bi-LSTM to generate better word representations, concatenate it with pre-trained word embeddings and feed concatenated embeddings to the word-level Bi-LSTM. The framework used for language like English is illustrated in Figure 6.

We also convert all characters to lowercase and use the same word embeddings provided by Yang et al. (2017)[3]. Also, we concatenate the training set and the development set for both domains and sample the same 10% from *TwitterNER* as (Yang et al., 2017) to be target domain training data. Since Yang et al. (2017) merge training and development set into training data, both Yang et al. (2017) and we report the best performance in the target domain test set.

[3] https://github.com/kimiyoung/transfer

Neural Fine-Grained Entity Type Classification with Hierarchy-Aware Loss

Peng Xu
Department of Computing Science
University of Alberta
Edmonton, Canada
pxu4@ualberta.ca

Denilson Barbosa
Department of Computing Science
University of Alberta
Edmonton, Canada
denilson@ualberta.ca

Abstract

The task of Fine-grained Entity Type Classification (FETC) consists of assigning types from a hierarchy to entity mentions in text. Existing methods rely on distant supervision and are thus susceptible to noisy labels that can be *out-of-context* or *overly-specific* for the training sentence. Previous methods that attempt to address these issues do so with heuristics or with the help of hand-crafted features. Instead, we propose an end-to-end solution with a neural network model that uses a variant of cross-entropy loss function to handle *out-of-context* labels, and hierarchical loss normalization to cope with *overly-specific* ones. Also, previous work solve FETC a multi-label classification followed by ad-hoc post-processing. In contrast, our solution is more elegant: we use public word embeddings to train a single-label that jointly learns representations for entity mentions and their context. We show experimentally that our approach is robust against noise and consistently outperforms the state-of-the-art on established benchmarks for the task.

1 Introduction

Fine-grained Entity Type Classification (FETC) aims at labeling entity mentions in context with one or more specific types organized in a hierarchy (e.g., **actor** as a subtype of **artist**, which in turn is a subtype of **person**). Fine-grained types help in many applications, including relation extraction (Mintz et al., 2009), question answering (Li and Roth, 2002), entity linking (Lin et al., 2012), knowledge base completion (Dong et al., 2014) and entity recommendation (Yu et al., 2014). Because of the high cost in labeling large training corpora with fine-grained types, current FETC systems resort to distant supervision (Mintz et al., 2009) and annotate mentions in the training corpus with *all types* associated with the entity in a knowledge graph. This is illustrated in Figure 1, with three training sentences about entity *Steve Kerr*. Note that while the entity belongs to three fine-grained types (**person**, **athlete**, and **coach**), some sentences provide evidence of only some of the types: **person** and **coach** from **S1**, **person** and **athlete** from **S2**, and just **person** for **S3**. Clearly, direct distant supervision leads to noisy training data which can hurt the accuracy of the FETC model.

One kind of noise introduced by distant supervision is assigning labels that are *out-of-context* (**athlete** in **S1** and **coach** in **S2**) for the sentence. Current FETC systems sidestep the issue by either ignoring *out-of-context* labels or using simple pruning heuristics like discarding training examples with entities assigned to multiple types in the knowledge graph. However, both strategies are inelegant and hurt accuracy. Another source of noise introduced by distant supervision is when the type is *overly-specific* for the context. For instance, example **S3** does not support the inference that Mr. Kerr is either an **athlete** or a **coach**. Since existing knowledge graphs give more attention to notable entities with more specific types, *overly-specific* labels bias the model towards popular subtypes instead of generic ones, *i.e.*, preferring **athlete** over **person**. Instead of correcting for this bias, most existing FETC systems ignore the problem and treat each type equally and independently, ignoring that many types are semantically related.

Besides failing to handle noisy training data there are two other limitations of previous FETC approaches we seek to address. First, they rely on hand-crafted features derived from various NLP tools; therefore, the inevitable errors introduced by these tools propagate to the FETC systems via the training data. Second, previous systems treat FETC as a multi-label classification problem: during type inference they predict a plausibility score for each type, and, then, either classify types

Proceedings of NAACL-HLT 2018, pages 16–25
New Orleans, Louisiana, June 1 - 6, 2018.

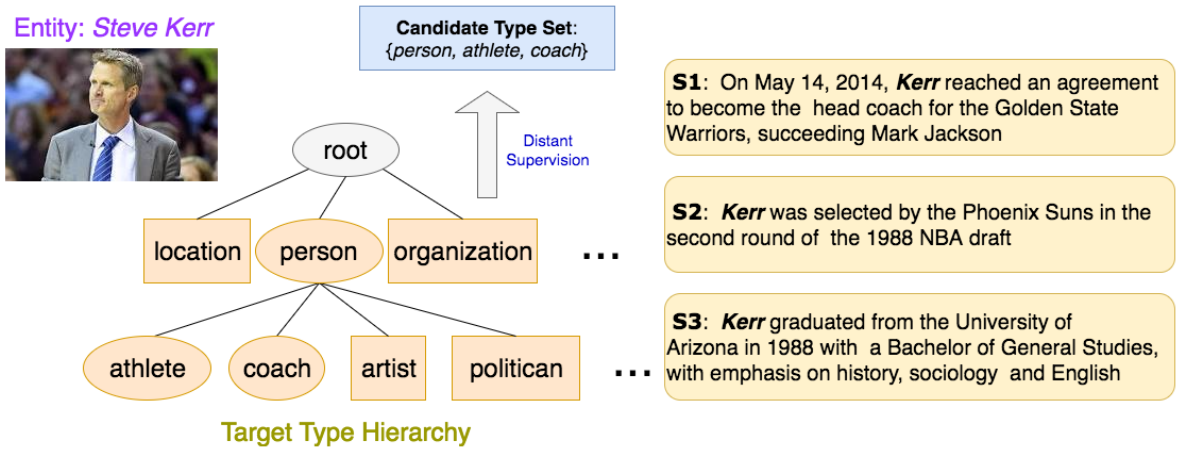


Figure 1: With distant supervision, all the three mentions of *Steve Kerr* shown are labeled with the same types in oval boxes in the target type hierarchy. While only part of the types are correct: **person** and **coach** for **S1**, **person** and **athlete** for **S2**, and just **person** for **S3**.

with scores above a threshold (Mintz et al., 2009; Gillick et al., 2014; Shimaoka et al., 2017) or perform a top-down search in the given type hierarchy (Ren et al., 2016a; Abhishek et al., 2017).

Contributions: We propose a neural network based model to overcome the drawbacks of existing FETC systems mentioned above. With publicly available word embeddings as input, we learn two different entity representations and use bidirectional long-short term memory (LSTM) with attention to learn the context representation. We propose a variant of cross entropy loss function to handle *out-of-context* labels automatically during the training phase. Also, we introduce hierarchical loss normalization to adjust the penalties for correlated types, allowing our model to understand the type hierarchy and alleviate the negative effect of *overly-specific* labels.

Moreover, in order to simplify the problem and take advantage of previous research on hierarchical classification, we transform the multi-label classification problem to a single-label classification problem. Based on the assumption that each mention can only have one *type-path* depending on the context, we leverage the fact that type hierarchies are forests, and represent each *type-path* uniquely by the terminal type (which might not be a leaf node). For Example, *type-path* **root-person-coach** can be represented as just **coach**, while **root-person** can be unambiguously represented as the non-leaf **person**.

Finally, we report on an experimental validation against the state-of-the-art on established benchmarks that shows that our model can adapt to noise in training data and consistently outperform previous methods. In summary, we describe a single, much simpler and more elegant neural network model that attempts FETC "end-to-end" without post-processing or ad-hoc features and improves on the state-of-the-art for the task.

2 Related Work

Fine-Grained Entity Type Classification: The first work to use distant supervision (Mintz et al., 2009) to induce a large - but noisy - training set and manually label a significantly smaller dataset to evaluate their FETC system, was Ling and Weld (2012) who introduced both a training and evaluation dataset FIGER (GOLD). They used a linear classifier perceptron for multi-label classification. While initial work largely assumed that mention assignments could be done independently of the mention context, Gillick et al. (2014) introduced the concept of context-dependent FETC where the types of a mention are constrained to what can be deduced from its context and introduced a new OntoNotes-derived (Weischedel et al., 2011) manually annotated evaluation dataset. In addition, they addressed the problem of label noise induced by distant supervision and proposed three label cleaning heuristics. Yogatama et al. (2015) proposed an embedding-based model where user-defined features and labels were embedded into a low dimensional feature space to facilitate information sharing among labels. Ma et al. (2016) presented a label embedding method that incor-

	Attentive	AFET	LNR	AAA	**NFETC**
no hand-crafted features	—	—	—	✓	✓
uses attentive neural network	✓	—	—	—	✓
adopts single label setting	—	—	—	—	✓
handles *out-of-context* noise	—	✓	✓	✓	✓
handles *overly-specifc* noise	—	✓	✓	—	✓

Table 1: Summary comparison to related FETC work. FETC systems listed in the table: (1) **Attentive** (Shimaoka et al., 2017); (2) **AFET** (Ren et al., 2016a); (3) **LNR** (Ren et al., 2016b); (4) **AAA** (Abhishek et al., 2017).

porates prototypical and hierarchical information to learn pre-trained label embeddings and adpated a zero-shot framework that can predict both seen and previously unseen entity types.

Shimaoka et al. (2016) proposed an attentive neural network model that used LSTMs to encode the context of an entity mention and used an attention mechanism to allow the model to focus on relevant expressions in such context. Shimaoka et al. (2017) summarizes many neural architectures for FETC task. These models ignore the *out-of-context noise*, that is, they assume that all labels obtained via distant supervision are "correct" and appropriate for every context in the training corpus. In our paper, a simple yet effective variant of cross entropy loss function is proposed to handle the problem of *out-of-context noise*.

Ren et al. (2016a) have proposed AFET, an FETC system, that separates the loss function for *clean* and *noisy* entity mentions and uses label-label correlation information obtained by given data in its parametric loss function. Considering the noise reduction aspects for FETC systems, Ren et al. (2016b) introduced a method called LNR to reduce label noise without data loss, leading to significant performance gains on both the evaluation dataset of FIGER(GOLD) and OntoNotes. Although these works consider both *out-of-context noise* and *overly-specific noise*, they rely on hand-crafted features which become an impediment to further improvement of the model performance. For LNR, because the noise reduction step is separated from the FETC model, the inevitable errors introduced by the noise reduction will be propagated into the FETC model which is undesirable. In our FETC system, we handle the problem induced from *irrelevant noise* and *overly-specific noise* seamlessly inside the model and avoid the usage of hand-crafted features.

Most recently, following the idea from AFET, Abhishek et al. (2017) proposed a simple neural network model which incorporates noisy label information using a variant of non-parametric hinge loss function and gain great performance improvement on FIGER(GOLD). However, their work overlooks the effect of *overly-specific noise*, treating each type label equally and independently when learning the classifiers and ignores possible correlations among types.

Hierarchical Loss Function: Due to the intrinsic type hierarchy existing in the task of FETC, it is natural to adopt the idea of hierarchical loss function to adjust the penalties for FETC mistakes depending on how far they are in the hierarchy. The penalty for predicting **person** instead of **athlete** should less than the penalty for predicting **organization**. To the best of our knowledge, the first use of a hierarchical loss function was originally introduced in the context of document categorization with support vector machines (Cai and Hofmann, 2004). However, that work assumed that weights to control the hierarchical loss would be solicited from domain experts, which is inapplicable for FETC. Instead, we propose a method called hierarchical loss normalization which can overcome the above limitations and be incorporated with cross entropy loss used in our neural architecture.

Table 1 provides a summary comparison of our work against the previous state-of-the-art in fine grained entity typing.

3 Background and Problem

Our task is to automatically reveal the type information for entity mentions in context. The input is a knowledge graph Ψ with schema $\mathcal{Y}_\Psi$, whose types are organized into a type hierarchy $\mathcal{Y}$, and an automatically labeled training corpus $\mathcal{D}$ obtained by distant supervision with $\mathcal{Y}$. The output is a *type-path* in $\mathcal{Y}$ for each named entity mentioned in a test sentence from a corpus $\mathcal{D}_t$.

More precisely, a labeled corpus for entity type classification consists of a set of extracted *entity mentions* $\{m_i\}_{i=1}^N$ (*i.e.*, token spans representing entities in text), the *context* (*e.g.,* sentence, paragraph) of each mention $\{c_i\}_{i=1}^N$, and the *candidate*

type sets $\{\mathcal{Y}_i\}_{i=1}^N$ automatically generated for each mention.

We represent the training corpus using a set of mention-based triples $\mathcal{D} = \{(m_i, c_i, \mathcal{Y}_i)\}_{i=1}^N$.

If $\mathcal{Y}_i$ is free of *out-of-context noise*, the type labels for each m_i should form a single *type-path* in $\mathcal{Y}_i$. However, $\mathcal{Y}_i$ may contain *type-paths* that are irrelevant to m_i in c_i if there exists *out-of-context noise*.

We denote the type set including all terminal types for each *type-path* as the target type set $\mathcal{Y}_i^t$. In the example type hierarchy shown in Figure 1, if $\mathcal{Y}_i$ contains types **person**, **athlete**, **coach**, $\mathcal{Y}_i^t$ should contain **athlete**, **coach**, but not **person**. In order to understand the trade-off between the effect of *out-of-context noise* and the size of the training set, we report on experiments with two different training sets: $\mathcal{D}_{filtered}$ only with triples whose $\mathcal{Y}_i$ form a single *type-path* in $\mathcal{D}$, and $\mathcal{D}_{raw}$ with all triples.

We formulate fine-grained entity classification problem as follows:

Definition 1 *Given an entity mention* $m_i = (w_p, \ldots, w_t)$ $(p, t \in [1, T], p \leq t)$ *and its context* $c_i = (w_1, \ldots, w_T)$ *where* T *is the context length, our task is to predict its most specific type* $\hat{y}_i$ *depending on the context.*

In practice, c_i is generated by truncating the original context with words beyond the context window size C both to the left and to the right of m_i. Specifically, we compute a probability distribution over all the $K = |\mathcal{Y}|$ types in the target type hierarchy $\mathcal{Y}$. The type with the highest probability is classified as the predicted type $\hat{y}_i$ which is the terminal type of the predicted *type-path*.

4 Methodology

This section details our Neural Fine-Grained Entity Type Classification (NFETC) model.

4.1 Input Representation

As stated in Section 3, the input is an entity mention m_i with its context c_i. First, we transform each word in the context c_i into a real-valued vector to provide lexical-semantic features. Given a word embedding matrix W^{wrd} of size $d_w \times |V|$, where V is the input vocabulary and d_w is the size of word embedding, we map every w_i to a column vector $\mathbf{w}_i^d \in \mathbb{R}^{d_w}$.

To additionally capture information about the relationship to the target entities, we incorporate word position embeddings (Zeng et al., 2014) to reflect relative distances between the i-th word to the entity mention. Every relative distance is mapped to a randomly initialized position vector in $\mathbb{R}^{d_p}$, where d_p is the size of position embedding. For a given word, we obtain the position vector $\mathbf{w}_i^p$. The overall embedding for the i-th word is $\mathbf{w}_i^E = [(\mathbf{w}_i^d)^\top, (\mathbf{w}_i^p)^\top]^\top$.

4.2 Context Representation

For the context c_i, we want to apply a non-linear transformation to the vector representation of c_i to derive a context feature vector $h_i = f(c_i; \theta)$ given a set of parameters θ. In this paper, we adopt bidirectional LSTM with d_s hidden units as $f(c_i; \theta)$. The network contains two sub-networks for the forward pass and the backward pass respectively. Here, we use element-wise sum to combine the forward and backward pass outputs. The output of the i-th word in shown in the following equation:

$$h_i = [\overrightarrow{h_i} \oplus \overleftarrow{h_i}] \tag{1}$$

Following Zhou et al. (2016), we employ word-level attention mechanism, which makes our model able to softly select the most informative words during training. Let H be a matrix consisting of output vectors $[h_1, h_2, \ldots, h_T]$ that the LSTM produced. The context representation r is formed by a weighted sum of these output vectors:

$$G = \tanh(H) \tag{2}$$

$$\alpha = softmax(w^\top G) \tag{3}$$

$$r_c = H\alpha^\top \tag{4}$$

where $H \in \mathbb{R}^{d_s \times T}$, w is a trained parameter vector. The dimension of w, α, r_c are d_s, T, d_s respectively.

4.3 Mention Representation

Averaging encoder: Given the entity mention $m_i = (w_p, \ldots, w_t)$ and its length $L = t - p + 1$, the averaging encoder computes the average word embedding of the words in m_i. Formally, the averaging representation r_a of the mention is computed as follows:

$$r_a = \frac{1}{L} \sum_{i=p}^{t} \mathbf{w}_i^d \tag{5}$$

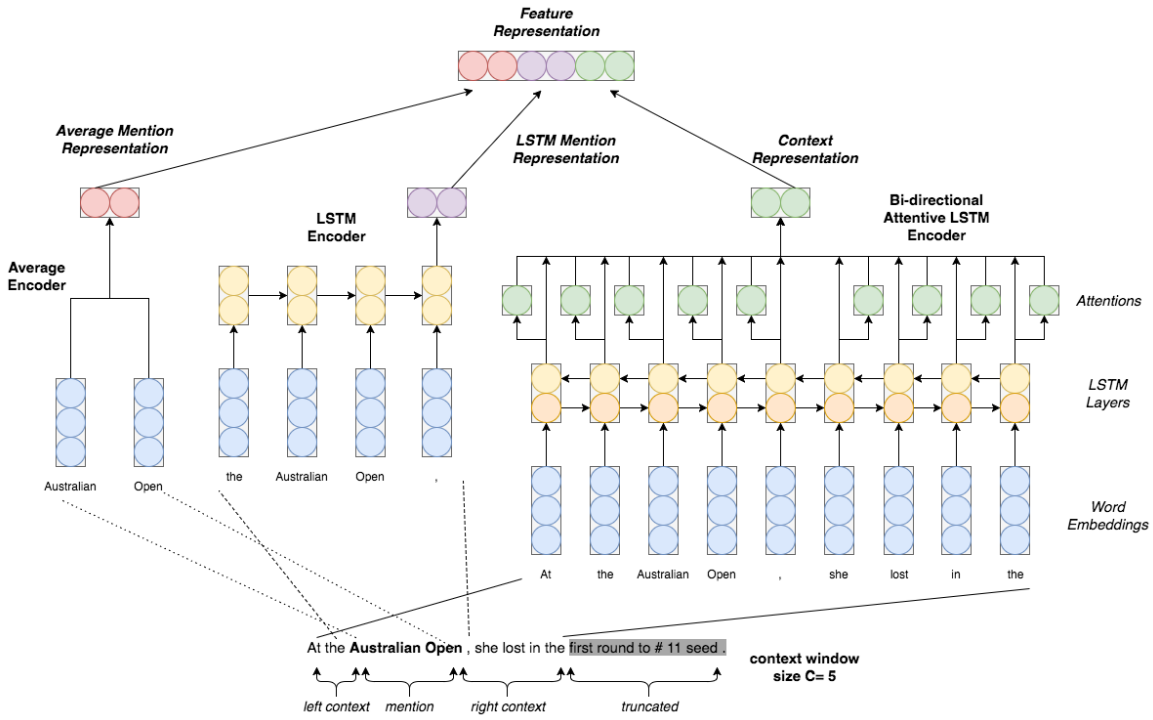


Figure 2: The architecture of the NFETC model.

This relatively simple method for composing the mention representation is motivated by it being less prone to overfitting (Shimaoka et al., 2017).

LSTM encoder: In order to capture more semantic information from the mentions, we add one token before and another after the target entity to the mention. The extended mention can be represented as $m_i^* = (w_{p-1}, w_p, \ldots, w_t, w_{t+1})$. The standard LSTM is applied to the mention sequence from left to right and produces the outputs $h_{p-1}, \ldots, h_{t+1}$. The last output h_{t+1} then serves as the LSTM representation r_l of the mention.

4.4 Optimization

We concatenate context representation and two mention representations together to form the overall feature representation of the input $R = [r_c, r_a, r_l]$. Then we use a softmax classifier to predict $\hat{y}_i$ from a discrete set of classes for a entity mention m and its context c with R as input:

$$\hat{p}(y|m,c) = \text{softmax}(WR + b) \quad (6)$$

$$\hat{y} = \arg\max_y \hat{p}(y|m,c) \quad (7)$$

where W can be treated as the learned type embeddings and b is the bias.

The traditional cross-entropy loss function is represented as follows:

$$J(\theta) = -\frac{1}{N}\sum_{i=1}^{N} \log(\hat{p}(y_i|m_i,c_i)) + \lambda\|\Theta\|^2 \quad (8)$$

where y_i is the only element in $\mathcal{Y}_i^t$ and $(m_i, c_i, \mathcal{Y}_i) \in \mathcal{D}_{filtered}$. λ is an L2 regularization hyperparameter and Θ denotes all parameters of the considered model.

In order to handle data with *out-of-context noise* (in other words, with multiple labeled types) and take full advantage of them, we introduce a simple yet effective variant of the cross-entropy loss:

$$J(\theta) = -\frac{1}{N}\sum_{i=1}^{N} \log(\hat{p}(y_i^*|m_i,c_i)) + \lambda\|\Theta\|^2 \quad (9)$$

where $y_i^* = \arg\max_{y \in \mathcal{Y}_i^t} \hat{p}(y|m_i,c_i)$ and $(m_i, c_i, \mathcal{Y}_i) \in \mathcal{D}_{raw}$. With this loss function, we assume that the type with the highest probability among $\mathcal{Y}_i^t$ during training as the correct type. If there is only one element in $\mathcal{Y}_i^t$, this loss function is equivalent to the cross-entropy loss function. Wherever there are multiple elements, it can filter the less probable types based on the local context automatically.

4.5 Hierarchical Loss Normalization

Since the fine-grained types tend to form a forest of type hierarchies, it is unreasonable to treat every type equally. Intuitively, it is better to predict an ancestor type of the true type than some other unrelated type. For instance, if one example is labeled as **athlete**, it is reasonable to predict its type as **person**. However, predicting other high level types like **location** or **organization** would be inappropriate. In other words, we want the loss function to penalize less the cases where types are related. Based on the above idea, we adjust the estimated probability as follows:

$$p^*(\hat{y}|m,c) = p(\hat{y}|m,c) + \beta * \sum_{t \in \Gamma} p(t|m,c) \quad (10)$$

where Γ is the set of ancestor types along the *type-path* of $\hat{y}$, β is a hyperparameter to tune the penalty. Afterwards, we re-normalize it back to a probability distribution, a process which we denote as *hierarchical loss normalization*.

As discussed in Section 1, there exists *overly-specific noise* in the automatically labeled training sets which hurt the model performance severely. With *hierarchical loss normalization*, the model will get less penalty when it predicts the actual type for one example with *overly-specific noise*. Hence, it can alleviate the negative effect of *overly-specific noise* effectively. Generally, *hierarchical loss normalization* can make the model somewhat understand the given type hierarchy and learn to detect those overly-specific cases. During classification, it will make the models prefer generic types unless there is a strong indicator for a more specific type in the context.

4.6 Regularization

Dropout, proposed by Hinton et al. (2012), prevents co-adaptation of hidden units by randomly omitting feature detectors from the network during forward propagation. We employ both input and output dropout on LSTM layers. In addition, we constrain L2-norms for the weight vectors as shown in Equations 8, 9 and use early stopping to decide when to stop training.

5 Experiments

This section reports an experimental evaluation of our NFETC approach using the previous state-of-the-art as baselines.

	FIGER(GOLD)	OntoNotes
# types	113	89
# raw training mentions	2009898	253241
# raw testing mentions	563	8963
% filtered training mentions	64.46	73.13
% filtered testing mentions	88.28	94.00
Max hierarchy depth	2	3

Table 2: Statistics of the datasets

5.1 Datasets

We evaluate the proposed model on two standard and publicly available datasets, provided in a pre-processed tokenized format by Shimaoka et al. (2017). Table 2 shows statistics about the benchmarks. The details are as follows:

- **FIGER(GOLD):** The training data consists of Wikipedia sentences and was automatically generated with distant supervision, by mapping Wikipedia identifiers to Freebase ones. The test data, mainly consisting of sentences from news reports, was manually annotated as described by Ling and Weld (2012).

- **OntoNotes:** The OntoNotes dataset consists of sentences from newswire documents present in the OntoNotes text corpus (Weischedel et al., 2013). DBpedia spotlight (Daiber et al., 2013) was used to automatically link entity mention in sentences to Freebase. Manually annotated test data was shared by Gillick et al. (2014).

Because the type hierarchy can be somewhat understood by our proposed model, the quality of the type hierarchy can also be a key factor to the performance of our model. We find that the type hierarchy for FIGER(GOLD) dataset following Freebase has some flaws. For example, **software** is not a subtype of **product** and **government** is not a subtype of **organization**. Following the proposed type hierarchy of Ling and Weld (2012), we refine the Freebase-based type hierarchy. The process is a one-to-one mapping for types in the original dataset and we didn't add or drop any type or sentence in the original dataset. As a result, we can directly compare the results of our proposed model with or without this refinement.

Aside from the advantages brought by adopting the single label classification setting, we can see one disadvantage of this setting based on Table 2. That is, the performance upper bounds of

our proposed model are no longer 100%: for example, the best strict accuracy we can get in this setting is 88.28% for **FIGER(GOLD)**. However, as the strict accuracy of state-of-the-art methods are still nowhere near 80% (Table 3), the evaluation we perform is still informative.

5.2 Baselines

We compared the proposed model with state-of-the-art FETC systems [1]: (1) **Attentive** (Shimaoka et al., 2017); (2) **AFET** (Ren et al., 2016a); (3) **LNR+FIGER** (Ren et al., 2016b); (4) **AAA** (Abhishek et al., 2017).

We compare these baselines with variants of our proposed model: (1) **NFETC(f)**: basic neural model trained on $\mathcal{D}_{filtered}$ (recall Section 4.4); (2) **NFETC-hier(f)**: neural model with hierarichcal loss normalization trained on $\mathcal{D}_{filtered}$. (3) **NFETC(r)**: neural model with proposed variant of cross-entropy loss trained on $\mathcal{D}_{raw}$; (4) **NFETC-hier(r)**: neural model with proposed variant of cross-entropy loss and hierarchical loss normalization trained on $\mathcal{D}_{raw}$.

5.3 Experimental Setup

For evaluation metrics, we adopt the same criteria as Ling and Weld (2012), that is, we evaluate the model performance by strict accuracy, loose macro, and loose micro F-scores. These measures are widely used in existing FETC systems (Shimaoka et al., 2017; Ren et al., 2016b,a; Abhishek et al., 2017).

We use pre-trained word embeddings that were not updated during training to help the model generalize to words not appearing in the training set. For this purpose, we used the freely available 300-dimensional cased word embedding trained on 840 billion tokens from the Common Crawl supplied by Pennington et al. (2014). For both datasets, we randomly sampled 10% of the test set as a development set, on which we do the hyperparameters tuning. The remaining 90% is used for final evaluation. We run each model with the well-tuned hyperparameter setting five times and report their average strict accuracy, macro F1 and micro F1 on the test set. The proposed model was implemented using the TensorFlow framework. [2]

[1]The results of the baselines are all as reported in their corresponding papers.

[2]The code to replicate the work is available at: https://github.com/billy-inn/NFETC

Parameter	FIGER(GOLD)	OntoNotes
lr	0.0002	0.0002
d_p	85	20
d_s	180	440
p_i	0.7	0.5
p_o	0.9	0.5
λ	0.0	0.0001
β	0.4	0.3

Table 4: Hyperparameter Settings

5.4 Hyperparameter Setting

In this paper, we search different hyperparameter settings for FIGER(GOLD) and OntoNotes separately, considering the differences between the two datasets. The hyperparameters include the learning rate lr for Adam Optimizer, size of word position embeddings (WPE) d_p, state size for LSTM layers d_s, input dropout keep probability p_i and output dropout keep probability p_o for LSTM layers [3], L2 regularization parameter λ and parameter to tune hierarchical loss normalization β. The values of these hyperparameters, obtained by evaluating the model performance on the development set, for each dataset can be found in Table 4.

5.5 Performance comparison and analysis

Table 3 compares our models with other state-of-the-art FETC systems on FIGER(GOLD) and OntoNotes. The proposed model performs better than the existing FETC systems, consistently on both datasets. This indicates benefits of the proposed representation scheme, loss function and hierarchical loss normalization.

Discussion about *Out-of-context Noise*: For dataset FIGER(GOLD), the performance of our model with the proposed variant of cross-entropy loss trained on $\mathcal{D}_{raw}$ is significantly better than the basic neural model trained on $\mathcal{D}_{filtered}$, suggesting that the proposed variant of the cross-entropy loss function can make use of the data with *out-of-context noise* effectively. On the other hand, the improvement introduced by our proposed variant of cross-entropy loss is not as significant for the OntoNotes benchmark. This may be caused by the fact that OntoNotes is much smaller than FIGER(GOLD) and proportion of examples without *out-of-context noise* are also higher, as shown in Table 2.

[3]Following TensorFlow terminology.

Model	FIGER(GOLD)			OntoNotes		
	Strict Acc.	Macro F1	Micro F1	Strict Acc.	Macro F1	Micro F1
Attentive	59.68	78.97	75.36	51.74	70.98	64.91
AFET	53.3	69.3	66.4	55.1	71.1	64.7
LNR+FIGER	59.9	76.3	74.9	57.2	71.5	66.1
AAA	65.8	**81.2**	77.4	52.2	68.5	63.3
NFETC(f)	57.9 ± 1.3	78.4 ± 0.8	75.0 ± 0.7	54.4 ± 0.3	71.5 ± 0.4	64.9 ± 0.3
NFETC-hier(f)	68.0 ± 0.8	$\mathbf{81.4 \pm 0.8}$	77.9 ± 0.7	59.6 ± 0.2	$\mathbf{76.1 \pm 0.2}$	69.7 ± 0.2
NFETC(r)	56.2 ± 1.0	77.2 ± 0.9	74.3 ± 1.1	54.8 ± 0.4	71.8 ± 0.4	65.0 ± 0.4
NFETC-hier(r)	$\mathbf{68.9 \pm 0.6}$	$\mathbf{81.9 \pm 0.7}$	$\mathbf{79.0 \pm 0.7}$	$\mathbf{60.2 \pm 0.2}$	$\mathbf{76.4 \pm 0.1}$	$\mathbf{70.2 \pm 0.2}$

Table 3: Strict Accuracy, Macro F1 and Micro F1 for the models tested on the FIGER(GOLD) and OntoNotes datasets.

Test Sentence	Ground Truth
S1: ***Hopkins*** said four fellow elections is curious , considering the ...	**Person**
S2: ... for WiFi communications across all ***the SD cards.***	**Product**
S3: A handful of professors in the ***UW*** Department of Chemistry ...	**Educational Institution**
S4: Work needs to be done and, in ***Washington state***, ...	**Province**
S5: ***ASC*** Director Melvin Taing said that because the commission is ...	**Organization**

Table 5: Examples of test sentences in FIGER(GOLD) where the entity mentions are marked as bold italics.

Investigations on *Overly-Specific Noise*: With hierarchical loss normalization, the performance of our models are consistently better no matter whether trained on $\mathcal{D}_{raw}$ or $\mathcal{D}_{filtered}$ on both datasets, demonstrating the effectiveness of this hierarchical loss normalization and showing that *overly-specific noise* has a potentially significant influence on the performance of FETC systems.

5.6 T-SNE Visualization of Type Embeddings

By visualizing the learned type embeddings (Figure 3), we can observe that the parent types are mixed with their subtypes and forms clear distinct clusters without hierarchical loss normalization, making it hard for the model to distinguish subtypes like **actor** or **athlete** from their parent types **person**. This also biases the model towards the most popular subtype. While the parent types tend to cluster together and the general pattern is more complicated with hierarchical loss normalization. Although it's not as easy to interpret, it hints that our model can learn rather subtle intricacies and correlations among types latent in the data with the help of hierarchical loss normalization, instead of sticking to a pre-defined hierarchy.

5.7 Error Analysis on FIGER(GOLD)

Since there are only 563 sentences for testing in FIGER(GOLD), we look into the predictions for all the test examples of all variants of our model. Table 5 shows 5 examples of test sentence. Without hierarchical loss normalization, our model will make too aggressive predictions for S1 with **Politician** and for S2 with **Software**. This kind of mistakes are very common and can be effectively reduced by introducing hierarchical loss normalization leading to significant improvements on the model performance. Using the changed loss function to handle multi-label (noisy) training data can help the model distinguish ambiguous cases. For example, our model trained on $\mathcal{D}_{filtered}$ will misclassify S5 as **Title**, while the model trained on $\mathcal{D}_{raw}$ can make the correct prediction.

However, there are still some errors that can't be fixed with our model. For example, our model cannot make correct predictions for S3 and S4 due to the fact that our model doesn't know that ***UW*** is an abbreviation of *University of Washington* and ***Washington state*** is the name of a province. In addition, the influence of *overly-specific noise* can only be alleviated but not eliminated. Sometimes, our model will still make too aggressive or conservative predictions. Also, mixing up very ambiguous entity names is inevitable in this task.

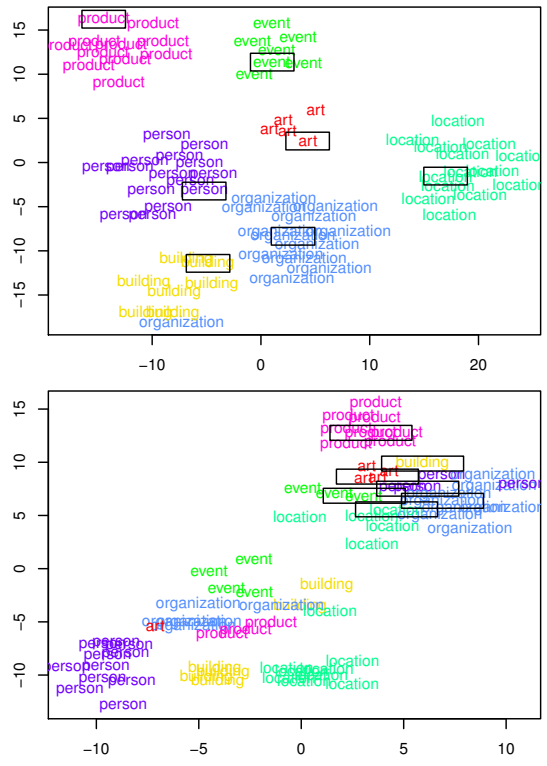


Figure 3: T-SNE visualization of the type embeddings learned from FIGER(GOLD) dataset where subtypes share the same color as their parent type. The seven parent types are shown in the black boxes. The below sub-figure uses the hierarchical loss normalization, while the above not.

6 Conclusion and Further Work

In this paper, we studied two kinds of noise, namely *out-of-context noise* and *overly-specific noise*, for noisy type labels and investigate their effects on FETC systems. We proposed a neural network based model which jointly learns representations for entity mentions and their context. A variant of cross-entropy loss function was used to handle *out-of-context noise*. Hierarchical loss normalization was introduced into our model to alleviate the effect of *overly-specific noise*. Experimental results on two publicly available datasets demonstrate that the proposed model is robust to these two kind of noise and outperforms previous state-of-the-art methods significantly.

More work can be done to further develop hierarchical loss normalization since currently it's very simple. Considering type information is valuable in various NLP tasks, we can incorporate results produced by our FETC system to other tasks, such as relation extraction, to check our model's effectiveness and help improve other tasks' performance. In addition, tasks like relation extraction are complementary to the task of FETC and therefore may have potentials to be digged to help improve the performance of our system in return.

Acknowledgments

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

Abhishek Abhishek, Ashish Anand, and Amit Awekar. 2017. Fine-grained entity type classification by jointly learning representations and label embeddings. *In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)* pages 797–807.

Lijuan Cai and Thomas Hofmann. 2004. Hierarchical document categorization with support vector machines. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*. ACM, pages 78–87.

Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N Mendes. 2013. Improving efficiency and accuracy in multilingual entity extraction. *Proceedings of the 9th International Conference on Semantic Systems* pages 121–124.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* pages 601–610.

Dan Gillick, Nevena Lazic, Kuzman Ganchev, Jesse Kirchner, and David Huynh. 2014. Context-dependent fine-grained entity type tagging. *arXiv preprint arXiv:1412.1820* .

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

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, pages 1–7.

Thomas Lin, Oren Etzioni, et al. 2012. No noun phrase left behind: detecting and typing unlinkable entities. *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning* pages 893–903.

Xiao Ling and Daniel S Weld. 2012. Fine-grained entity recognition. *AAAI* .

Yukun Ma, Erik Cambria, and Sa Gao. 2016. Label embedding for zero-shot fine-grained named entity typing. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. pages 171–180.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP* pages 1003–1011.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *EMNLP* 14(1532–1543).

Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han. 2016a. Afet: Automatic fine-grained entity typing by hierarchical partial-label embedding. *EMNLP* 16(17).

Xiang Ren, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, and Jiawei Han. 2016b. Label noise reduction in entity typing by heterogeneous partial-label embedding. *KDD* .

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2016. An attentive neural architecture for fine-grained entity type classification. *arXiv preprint arXiv:1604.05525* .

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2017. Neural architectures for fine-grained entity type classification. *In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)* .

Ralph Weischedel, Eduard Hovy, Mitchell Marcus, Martha Palmer, Robert Belvin, Sameer Pradhan, Lance Ramshaw, and Nianwen Xue. 2011. Ontonotes: A large training corpus for enhanced processing. *Handbook of Natural Language Processing and Machine Translation. Springer* .

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA* .

Dani Yogatama, Daniel Gillick, and Nevena Lazic. 2015. Embedding methods for fine grained entity type classification. *ACL (2)* pages 291–296.

Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. 2014. Personalized entity recommendation: A heterogeneous information network approach. *Proceedings of the 7th ACM international conference on Web search and data mining* pages 283–292.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation classification via convolutional deep neural network. *COLING* pages 2335–2344.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. *The 54th Annual Meeting of the Association for Computational Linguistics* .

Joint Bootstrapping Machines for High Confidence Relation Extraction

Pankaj Gupta[1,2], Benjamin Roth[2], Hinrich Schütze[2]
[1]Corporate Technology, Machine-Intelligence (MIC-DE), Siemens AG Munich, Germany
[2]CIS, University of Munich (LMU) Munich, Germany
pankaj.gupta@siemens.com | pankaj.gupta@campus.lmu.de
{beroth, inquiries}@cis.lmu.de

Abstract

Semi-supervised bootstrapping techniques for relationship extraction from text iteratively expand a set of initial seed instances. Due to the lack of labeled data, a key challenge in bootstrapping is semantic drift: if a false positive instance is added during an iteration, then all following iterations are contaminated. We introduce BREX, a new bootstrapping method that protects against such contamination by highly effective confidence assessment. This is achieved by using entity and template seeds jointly (as opposed to just one as in previous work), by expanding entities and templates in parallel and in a mutually constraining fashion in each iteration and by introducing higher-quality similarity measures for templates. Experimental results show that BREX achieves an F_1 that is 0.13 (0.87 vs. 0.74) better than the state of the art for four relationships.

1 Introduction

Traditional semi-supervised bootstrapping relation extractors (REs) such as BREDS (Batista et al., 2015), SnowBall (Agichtein and Gravano, 2000) and DIPRE (Brin, 1998) require an initial set of seed *entity pairs* for the target binary relation. They find occurrences of positive seed entity pairs in the corpus, which are converted into extraction patterns, i.e., *extractors*, where we define an extractor as a cluster of instances generated from the corpus. The initial seed entity pair set is expanded with the relationship entity pairs newly extracted by the extractors from the text iteratively. The augmented set is then used to extract new relationships until a stopping criterion is met.

Due to lack of sufficient labeled data, rule-based systems dominate commercial use (Chiticariu et al., 2013). Rules are typically defined by creating patterns around the entities (entity extraction) or entity pairs (relation extraction). Recently, supervised machine learning, especially deep learning techniques (Gupta et al., 2015; Nguyen and Grishman, 2015; Vu et al., 2016a,b; Gupta et al., 2016), have shown promising results in entity and relation extraction; however, they need sufficient hand-labeled data to train models, which can be costly and time consuming for web-scale extractions. Bootstrapping machine-learned rules can make extractions easier on large corpora. Thus, open information extraction systems (Carlson et al., 2010; Fader et al., 2011; Mausam et al., 2012; Mesquita et al., 2013; Angeli et al., 2015) have recently been popular for domain specific or independent pattern learning.

Hearst (1992) used hand written rules to generate more rules to extract hypernym-hyponym pairs, without distributional similarity. For entity extraction, Riloff (1996) used seed entities to generate extractors with heuristic rules and scored them by counting positive extractions. Prior work (Lin et al., 2003; Gupta et al., 2014) investigated different extractor scoring measures. Gupta and Manning (2014) improved scores by introducing expected number of negative entities.

Brin (1998) developed the bootstrapping relation extraction system DIPRE that generates extractors by clustering contexts based on string matching. SnowBall (Agichtein and Gravano, 2000) is inspired by DIPRE but computes a TF-IDF representation of each context. BREDS (Batista et al., 2015) uses word embeddings (Mikolov et al., 2013) to bootstrap relationships.

Related work investigated adapting extractor scoring measures in bootstrapping entity extraction with either entities or *templates* (Table 1) as seeds (Table 2). The state-of-the-art relation extractors bootstrap with only seed entity pairs and suffer due to a surplus of unknown extractions and the lack of labeled data, leading to low confidence extractors. This in turn leads to to low confidence in the system output. Prior RE sys-

Proceedings of NAACL-HLT 2018, pages 26–36
New Orleans, Louisiana, June 1 - 6, 2018.

BREE	Bootstrapping Relation Extractor with *Entity pair*
BRET	Bootstrapping Relation Extractor with *Template*
BREJ	Bootstrapping Relation Extractor in Joint learning
type	a named entity type, e.g., `person`
typed entity	a typed entity, e.g., <"Obama",`person`>
entity pair	a pair of two typed entities
template	a triple of vectors ($\vec{v}_{-1}$, $\vec{v}_0$, $\vec{v}_1$) and an entity pair
instance	entity pair and template (types must be the same)
γ	instance set extracted from corpus
i	a member of γ, i.e., an instance
$x(i)$	the entity pair of instance i
$\mathfrak{t}(i)$	the template of instance i
G_p	a set of positive seed entity pairs
G_n	a set of negative seed entity pairs
$\mathfrak{G}_p$	a set of positive seed templates
$\mathfrak{G}_n$	a set of negative seed templates
$\mathcal{G}$	$< G_p, G_n, \mathfrak{G}_p, \mathfrak{G}_n >$
k_{it}	number of iterations
λ_{cat}	cluster of instances (*extractor*)
cat	category of *extractor* λ
λ_{NNHC}	Non-Noisy-High-Confidence extractor (True Positive)
λ_{NNLC}	Non-Noisy-Low-Confidence extractor (True Negative)
λ_{NHC}	Noisy-High-Confidence extractor (False Positive)
λ_{NLC}	Noisy-Low-Confidence extractor (False Negative)

Table 1: Notation and definition of key terms

tems do not focus on improving the extractors' scores. In addition, SnowBall and BREDS used a weighting scheme to incorporate the importance of contexts around entities and compute a similarity score that introduces additional parameters and does not generalize well.

Contributions. **(1)** We propose a *Joint Bootstrapping Machine*[1] (JBM), an alternative to the entity-pair-centered bootstrapping for relation extraction that can take advantage of both entity-pair and template-centered methods to jointly learn extractors consisting of instances due to the occurrences of both entity pair and template seeds. It scales up the number of positive extractions for *non-noisy* extractors and boosts their confidence scores. We focus on improving the scores for *non-noisy-low-confidence* extractors, resulting in higher *recall*. The relation extractors bootstrapped with entity pair, template and joint seeds are named as *BREE*, *BRET* and *BREJ* (Table 1), respectively.

(2) Prior work on embedding-based context comparison has assumed that relations have *consistent syntactic expression* and has mainly addressed synonymy by using embeddings (e.g.,"acquired" – "bought"). In reality, there is *large variation in the syntax* of how relations are expressed, e.g., "MSFT to acquire NOK for $8B" vs. "MSFT earnings hurt by NOK acquisition". We introduce cross-context similarities that compare all parts of the context (e.g., "to acquire" and "acquisition") and show that these perform better (in terms of recall) than measures assuming consistent syntactic expression of relations.

(3) Experimental results demonstrate a 13% gain in $F1$ score on average for four relationships and suggest eliminating four parameters, compared to the state-of-the-art method.

The *motivation* and *benefits* of the proposed JBM for relation extraction is discussed in depth in section 2.3. The method is applicable for both entity and relation extraction tasks. However, in *context of relation extraction*, we call it *BREJ*.

2 Method

2.1 Notation and definitions

We first introduce the notation and terms (Table 1).

Given a relationship like "x acquires y", the task is to extract pairs of entities from a corpus for which the relationship is true. We assume that the arguments of the relationship are typed, e.g., x and y are organizations. We run a named entity tagger in preprocessing, so that the types of all candidate entities are given. The objects the bootstrapping algorithm generally handles are therefore *typed entities* (an entity associated with a type).

For a particular sentence in a corpus that states that the relationship (e.g., "acquires") holds between x and y, a *template* consists of three vectors that represent the context of x and y. $\vec{v}_{-1}$ represents the context before x, $\vec{v}_0$ the context between x and y and $\vec{v}_1$ the context after y. These vectors are simply sums of the embeddings of the corresponding words. A template is "typed", i.e., in addition to the three vectors it specifies the types of the two entities. An *instance* joins an entity pair and a template. The types of entity pair and template must be the same.

The first step of bootstrapping is to extract a set of instances from the input corpus. We refer to this set as γ. We will use i and j to refer to instances. $x(i)$ is the entity pair of instance i and $\mathfrak{t}(\mathrm{i})$ is the template of instance i.

A required input to our algorithm are sets of positive and negative seeds for either entity pairs (G_p and G_n) or templates ($\mathfrak{G}_p$ and $\mathfrak{G}_n$) or both. We define $\mathcal{G}$ to be a tuple of all four seed sets.

We run our bootstrapping algorithm for k_{it} iterations where k_{it} is a parameter.

[1] github.com/pgcool/Joint-Bootstrapping-Machines

A key notion is the similarity between two instances. We will experiment with different similarity measures. The baseline is (Batista et al., 2015)'s measure given in Figure 4, first line: the similarity of two instances is given as a weighted sum of the dot products of their before contexts ($\vec{v}_{-1}$), their between contexts ($\vec{v}_0$) and their after contexts ($\vec{v}_1$) where the weights w_p are parameters. We give this definition for instances, but it also applies to templates since only the context vectors of an instance are used, not the entities.

The similarity between an instance i and a cluster λ of instances is defined as the maximum similarity of i with any member of the cluster; see Figure 2, right, Eq. 5. Again, there is a straightforward extension to a cluster of templates: see Figure 2, right, Eq. 6.

The extractors Λ can be categorized as follows:

$$\Lambda_{NNHC} = \{\lambda \in \Lambda | \underbrace{\lambda \mapsto \Re}_{non-noisy} \wedge \text{cnf}(\lambda, \mathcal{G}) \geqslant \tau_{cnf}\} \quad (1)$$

$$\Lambda_{NNLC} = \{\lambda \in \Lambda | \lambda \mapsto \Re \wedge \text{cnf}(\lambda, \mathcal{G}) < \tau_{cnf}\} \quad (2)$$

$$\Lambda_{NHC} = \{\lambda \in \Lambda | \underbrace{\lambda \nmapsto \Re}_{noisy} \wedge \text{cnf}(\lambda, \mathcal{G}) \geqslant \tau_{cnf}\} \quad (3)$$

$$\Lambda_{NLC} = \{\lambda \in \Lambda | \lambda \nmapsto \Re \wedge \text{cnf}(\lambda, \mathcal{G}) < \tau_{cnf}\} \quad (4)$$

where $\Re$ is the relation to be bootstrapped. The λ_{cat} is a member of Λ_{cat}. For instance, a λ_{NNLC} is called as a *non-noisy-low-confidence* extractor if it represents the target relation (i.e., $\lambda \mapsto \Re$), however with the confidence below a certain threshold (τ_{cnf}). Extractors of types Λ_{NNHC} and Λ_{NLC} are desirable, those of types Λ_{NHC} and Λ_{NNLC} undesirable within bootstrapping.

2.2 The Bootstrapping Machines: BREX

To describe BREX (Figure 1) in its most general form, we use the term *item* to refer to an entity pair, a template or both.

The input to BREX (Figure 2, left, line 01) is a set γ of instances extracted from a corpus and $\mathcal{G}_{\text{seed}}$, a structure consisting of one set of positive and one set of negative seed items. $\mathcal{G}_{\text{yield}}$ (line 02) collects the items that BREX extracts in several iterations. In each of k_{it} iterations (line 03), BREX first initializes the cache $\mathcal{G}_{\text{cache}}$ (line 04); this cache collects the items that are extracted in this iteration. The design of the algorithm balances elements that ensure high recall with elements that ensure high precision.

High recall is achieved by starting with the seeds and making three "hops" that consecutively consider order-1, order-2 and order-3 neighbors

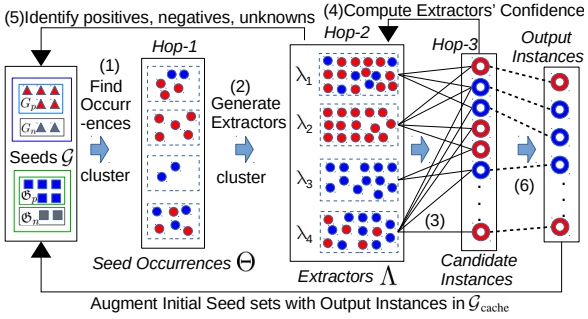


Figure 1: Joint Bootstrapping Machine. The red and blue filled circles/rings are the instances generated due to seed entity pairs and templates, respectively. Each dashed rectangular box represents a cluster of instances. Numbers indicate the flow. Follow the notations from Table 1 and Figure 2.

of the seeds. On line 05, we make the first hop: all instances that are similar to a seed are collected where "similarity" is defined differently for different BREX configurations (see below). The collected instances are then clustered, similar to work on bootstrapping by Agichtein and Gravano (2000) and Batista et al. (2015). On line 06, we make the second hop: all instances that are within τ_{sim} of a hop-1 instance are added; each such instance is only added to one cluster, the closest one; see definition of μ: Figure 2, Eq. 8. On line 07, we make the third hop: we include all instances that are within τ_{sim} of a hop-2 instance; see definition of ψ: Figure 2, Eq. 7. In summary, every instance that can be reached by three hops from a seed is being considered at this point. A cluster of hop-2 instances is named as *extractor*.

High precision is achieved by imposing, on line 08, a stringent check on each instance before its information is added to the cache. The core function of this check is given in Figure 2, Eq. 9. This definition is a soft version of the following hard max, which is easier to explain:

$$\text{cnf}(i, \Lambda, \mathcal{G}) \approx \max_{\{\lambda \in \Lambda | i \in \psi(\lambda)\}} \text{cnf}(i, \lambda, \mathcal{G})$$

We are looking for a cluster λ in Λ that licenses the extraction of i with high confidence. $\text{cnf}(i, \lambda, \mathcal{G})$ (Figure 2, Eq. 10), the *confidence* of a single cluster (i.e., extractor) λ for an instance, is defined as the product of the overall reliability of λ (which is independent of i) and the similarity of i to λ, the second factor in Eq. 10, i.e., $\text{sim}(i, \lambda)$. This factor $\text{sim}(i, \lambda)$ prevents an extraction by a cluster whose members are all distant from the instance – even if the cluster itself is highly reliable.

Algorithm: *BREX*

01 INPUT: γ, $\mathcal{G}_{\text{seed}}$
02 $\mathcal{G}_{\text{yield}} := \mathcal{G}_{\text{seed}}$
03 for k_{it} iterations:
04 $\quad \mathcal{G}_{\text{cache}} := \varnothing$
05 $\quad \Theta := \biguplus(\{i \in \gamma | \text{match}(i, \mathcal{G}_{\text{yield}})\}$
06 $\quad \Lambda := \{\mu(\theta, \Theta) | \theta \in \Theta\}$
07 $\quad$ for each $i \in \bigcup_{\lambda \in \Lambda} \psi(\lambda)$:
08 $\quad\quad$ if check$(i, \Lambda, \mathcal{G}_{\text{yield}})$:
09 $\quad\quad\quad$ add$(i, \mathcal{G}_{\text{cache}})$
10 $\quad \mathcal{G}_{\text{yield}} \cup= \mathcal{G}_{\text{cache}}$
11 OUTPUT: $\mathcal{G}_{\text{yield}}$, Λ

$$\text{sim}(i, \lambda) = \max_{i' \in \lambda} \text{sim}(i, i') \quad (5)$$

$$\text{sim}(i, \mathfrak{G}) = \max_{\mathfrak{t} \in \mathfrak{G}} \text{sim}(i, \mathfrak{t}) \quad (6)$$

$$\psi(\lambda) = \{i \in \gamma | \text{sim}(i, \lambda) \geqslant \tau_{\text{sim}}\} \quad (7)$$

$$\mu(\theta, \Theta) = \{i \in \gamma | \text{sim}(i, \theta) = d \wedge d = \max_{\theta \in \Theta} \text{sim}(i, \theta) \geqslant \tau_{\text{sim}}\} \quad (8)$$

$$\text{cnf}(i, \Lambda, \mathcal{G}) = 1 - \prod_{\{\lambda \in \Lambda | i \in \psi(\lambda)\}} (1 - \text{cnf}(i, \lambda, \mathcal{G})) \quad (9)$$

$$\text{cnf}(i, \lambda, \mathcal{G}) = \text{cnf}(\lambda, \mathcal{G}) \text{sim}(i, \lambda) \quad (10)$$

$$\text{cnf}(\lambda, \mathcal{G}) = \frac{1}{1 + w_n \frac{N_+(\lambda, \mathcal{G}_n)}{N_+(\lambda, \mathcal{G}_p)} + w_u \frac{N_0(\lambda, \mathcal{G})}{N_+(\lambda, \mathcal{G}_p)}} \quad (11)$$

$$N_0(\lambda, \mathcal{G}) = |\{i \in \lambda | x(i) \notin (G_p \cup G_n)\}| \quad (12)$$

Figure 2: BREX algorithm (left) and definition of key concepts (right)

	BREE	**BRET**	**BREJ**
Seed Type	*Entity pairs*	*Templates*	*Joint (Entity pairs + Templates)*
(i) $N_+(\lambda, \mathcal{G}_l)$	$\lvert\{i \in \lambda \lvert x(i) \in G_l\}\rvert$	$\lvert\{i \in \lambda \lvert \text{sim}(i, \mathfrak{G}_l) \geqslant \tau_{\text{sim}}\}\rvert$	$\lvert\{i \in \lambda \lvert x(i) \in G_l\}\rvert + \lvert\{i \in \lambda \lvert \text{sim}(i, \mathfrak{G}_l) \geqslant \tau_{\text{sim}}\}\rvert$
(ii) (w_n, w_u)	$(1.0, 0.0)$	$(1.0, 0.0)$	$(1.0, 0.0)$
05 match$(i, \mathcal{G})$	$x(i) \in G_p$	$\text{sim}(i, \mathfrak{G}_p) \geqslant \tau_{\text{sim}}$	$x(i) \in G_p \vee \text{sim}(i, \mathfrak{G}_p) \geqslant \tau_{\text{sim}}$
08 check$(i, \Lambda, \mathcal{G})$	$\text{cnf}(i, \Lambda, \mathcal{G}) \geqslant \tau_{\text{cnf}}$	$\text{cnf}(i, \Lambda, \mathcal{G}) \geqslant \tau_{\text{cnf}}$	$\text{cnf}(i, \Lambda, \mathcal{G}) \geqslant \tau_{\text{cnf}} \wedge \text{sim}(i, \mathfrak{G}_p) \geqslant \tau_{\text{sim}}$
09 add$(i, \mathcal{G})$	$G_p \cup= \{x(i)\}$	$\mathfrak{G}_p \cup= \{\mathfrak{x}(i)\}$	$G_p \cup= \{x(i)\}$, $\mathfrak{G}_p \cup= \{\mathfrak{x}(i)\}$

Figure 3: BREX configurations

The first factor in Eq. 10, i.e., $\text{cnf}(\lambda, \mathcal{G})$, assesses the reliability of a cluster λ: we compute the ratio $\frac{N_+(\lambda, \mathcal{G}_n)}{N_+(\lambda, \mathcal{G}_p)}$, i.e., the ratio between the number of instances in λ that match a negative and positive gold seed, respectively; see Figure 3, line (i). If this ratio is close to zero, then likely false positive extractions are few compared to likely true positive extractions. For the simple version of the algorithm (for which we set $w_n = 1$, $w_u = 0$), this results in $\text{cnf}(\lambda, \mathcal{G})$ being close to 1 and the reliability measure it not discounted. On the other hand, if $\frac{N_+(\lambda, \mathcal{G}_n)}{N_+(\lambda, \mathcal{G}_p)}$ is larger, meaning that the relative number of likely false positive extractions is high, then $\text{cnf}(\lambda, \mathcal{G})$ shrinks towards 0, resulting in progressive discounting of $\text{cnf}(\lambda, \mathcal{G})$ and leading to *non-noisy-low-confidence* extractor, particularly for a reliable λ. Due to lack of labeled data, the scoring mechanism cannot distinguish between noisy and non-noisy extractors. Therefore, an extractor is judged by its ability to extract more positive and less negative extractions. Note that we carefully designed this precision component to give good assessments while at the same time making maximum use of the available seeds. The reliability statistics are computed on λ, i.e., on hop-2 instances (not on hop-3 instances). The ratio $\frac{N_+(\lambda, \mathcal{G}_n)}{N_+(\lambda, \mathcal{G}_p)}$ is computed on instances that directly match a gold seed – this is the most reliable information we have available.

After all instances have been checked (line 08) and (if they passed muster) added to the cache (line 09), the inner loop ends and the cache is merged into the yield (line 10). Then a new loop (lines 03–10) of hop-1, hop-2 and hop-3 extensions and cluster reliability tests starts.

Thus, the algorithm consists of k_{it} iterations. There is a tradeoff here between τ_{sim} and k_{it}. We will give two extreme examples, assuming that we want to extract a fixed number of m instances where m is given. We can achieve this goal either by setting k_{it}=1 and choosing a small τ_{sim}, which will result in very large hops. Or we can achieve this goal by setting τ_{sim} to a large value and running the algorithm for a larger number of k_{it}. The flexibility that the two hyperparameters k_{it} and τ_{sim} afford is important for good performance.

$$\text{sim}_{\text{match}}(i,j) = \sum_{p\in\{-1,0,1\}} w_p \vec{v}_p(i)\vec{v}_p(j) \quad ; \quad \text{sim}_{cc}^{asym}(i,j) = \max_{p\in\{-1,0,1\}} \vec{v}_p(i)\vec{v}_0(j) \tag{13}$$

$$\text{sim}_{cc}^{sym1}(i,j) = \max\left(\max_{p\in\{-1,0,1\}} \vec{v}_p(i)\vec{v}_0(j), \max_{p\in\{-1,0,1\}} \vec{v}_p(j)\vec{v}_0(i)\right) \tag{14}$$

$$\text{sim}_{cc}^{sym2}(i,j) = \max\left(\left(\vec{v}_{-1}(i) + \vec{v}_1(i)\right)\vec{v}_0(j), \left(\vec{v}_{-1}(j) + \vec{v}_1(j)\right)\vec{v}_0(i), \vec{v}_0(i)\vec{v}_0(j)\right) \tag{15}$$

Figure 4: Similarity measures. These definitions for instances equally apply to templates since the definitions only depend on the "template part" of an instance, i.e., its vectors. (value is 0 if types are different)

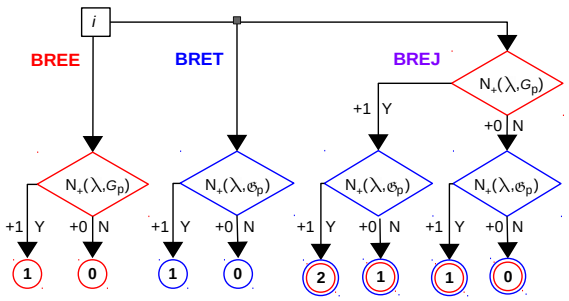


Figure 5: Illustration of Scaling-up Positive Instances. i: an instance in extractor, λ. Y: YES and N: NO

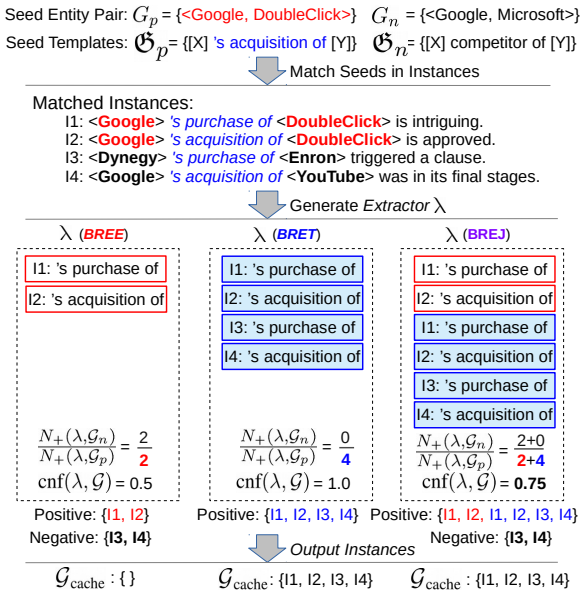


Figure 6: An illustration of scaling positive extractions and computing confidence for a non-noisy extractor generated for *acquired* relation. The dashed rectangular box represents an extractor λ, where λ (BREJ) is *hybrid* with 6 instances. Text segments matched with seed template are shown in italics. Unknowns (bold in black) are considered as negatives. $\mathcal{G}_{cache}$ is a set of output instances where $\tau_{cnf} = 0.70$.

2.3 BREE, BRET and BREJ

The main contribution of this paper is that we propose, as an alternative to entity-pair-centered BREE (Batista et al., 2015), template-centered BRET as well as BREJ (Figure 1), an instantiation of BREX that can take advantage of both entity pairs and templates. The differences and advantages of BREJ over BREE and BRET are:

(1) Disjunctive Matching of Instances: The first difference is realized in how the three algorithms match instances with seeds (line 05 in Figure 3). BREE checks whether the entity pair of an instance is one of the entity pair seeds, BRET checks whether the template of an instance is one of the template seeds and BREJ checks whether the disjunction of the two is true. The disjunction facilitates a higher hit rate in matching instances with seeds. The introduction of a few handcrafted templates along with seed entity pairs allows BREJ to leverage discriminative patterns and learn similar ones via distributional semantics. In Figure 1, the joint approach results in *hybrid* extractors Λ that contain instances due to seed occurrences Θ of both entity pairs and templates.

(2) Hybrid Augmentation of Seeds: On line 09 in Figure 3, we see that the bootstrapping step is defined in a straightforward fashion: the entity pair of an instance is added for BREE, the template for BRET and both for BREJ. Figure 1 demonstrates the hybrid augmentation of seeds via *red* and *blue* rings of *output instances*.

(3) Scaling Up Positives in Extractors: As discussed in section 2.2, a good measure of the quality of an extractor is crucial and N_+, the number of instances in an extractor λ that match a seed, is an important component of that. For BREE and BRET, the definition follows directly from the fact that these are entity-pair and template-centered instantiations of BREX, respectively. However, the disjunctive matching of instances for an extractor with entity pair and template seeds in BREJ (Figure 3 line "(i)") boosts the likelihood of finding positive instances. In Figure 5, we demonstrate computing the count of positive instances

Relationship	*Seed Entity Pairs*	*Seed Templates*
acquired	{Adidas;Reebok},{Google;DoubleClick}, {Widnes;Warrington},{Hewlett-Packard;Compaq}	{[X] acquire [Y]},{[X] acquisition [Y]},{[X] buy [Y]}, {[X] takeover [Y]},{[X] merger with [Y]}
founder-of	{CNN;Ted Turner},{Facebook;Mark Zuckerberg}, {Microsoft;Paul Allen},{Amazon;Jeff Bezos},	{[X] founded by [Y]},{[X] co-founder [Y]},{[X] started by [Y]}, {[X] founder of [Y]},{[X] owner of [Y]}
headquartered	{Nokia;Espoo},{Pfizer;New York}, {United Nations;New York},{NATO;Brussels},	{[X] based in [Y]},{[X] headquarters in [Y]},{[X] head office in [Y]}, {[X] main office building in [Y]},{[X] campus branch in [Y]}
affiliation	{Google;Marissa Mayer},{Xerox;Ursula Burns}, { Microsoft;Steve Ballmer},{Microsoft;Bill Gates},	{[X] CEO [Y]},{[X] resign from [Y]},{[X] founded by [Y]}, {[X] worked for [Y]},{[X] chairman director [Y]}

Table 2: Seed Entity Pairs and Templates for each relation. [X] and [Y] are slots for entity type tags.

$N_+(\lambda, \mathcal{G})$ for an extractor λ within the three systems. Observe that an instance i in λ can scale its $N_+(\lambda, \mathcal{G})$ by a factor of maximum 2 in BREJ if i is matched in both entity pair and template seeds. The reliability cnf$(\lambda, \mathcal{G})$ (Eq. 11) of an extractor λ is based on the ratio $\frac{N_+(\lambda,\mathcal{G}_n)}{N_+(\lambda,\mathcal{G}_p)}$, therefore suggesting that the scaling boosts its confidence.

In Figure 6, we demonstrate with an example how the joint bootstrapping scales up the positive instances for a *non-noisy* extractor λ, resulting in λ_{NNHC} for BREJ compared to λ_{NNLC} in BREE.

Due to unlabeled data, the instances not matching in seeds are considered either to be ignored/unknown N_0 or negatives in the confidence measure (Eq. 11). The former leads to high confidences for noisy extractors by assigning high scores, the latter to low confidences for non-noisy extractors by penalizing them. For a simple version of the algorithm in the illustration, we consider them as negatives and set $w_n = 1$. Figure 6 shows the three extractors (λ) generated and their confidence scores in BREE, BRET and BREJ. Observe that the scaling up of positives in BREJ due to BRET extractions (without w_n) discounts cnf$(\lambda, \mathcal{G})$ relatively lower than BREE. The discounting results in λ_{NNHC} in BREJ and λ_{NNLC} in BREE. The discounting in BREJ is adapted for *non-noisy* extractors facilitated by BRET in generating mostly non-noisy extractors due to stringent checks (Figure 3, line "(i)" and 05). Intuitively, the intermixing of non-noisy extractors (i.e., *hybrid*) promotes the scaling and boosts recall.

2.4 Similarity Measures

The before ($\vec{v}_{-1}$) and after ($\vec{v}_1$) contexts around the entities are highly sparse due to large variation in the syntax of how relations are expressed. SnowBall, DIPRE and BREE assumed that the between ($\vec{v}_0$) context mostly defines the syntactic expression for a relation and used weighted mechanism on the three contextual similarities in pairs, sim$_{match}$ (Figure 4). They assigned higher weights to the similarity in between ($p = 0$) contexts, that resulted in lower recall. We introduce attentive (max) similarity across all contexts (for example, $\vec{v}_{-1}(i)\vec{v}_0(j)$) to automatically capture the large variation in the syntax of how relations are expressed, without using any weights. We investigate asymmetric (Eq 13) and symmetric (Eq 14 and 15) similarity measures, and name them as *cross-context attentive* (sim$_{cc}$) similarity.

	ORG-ORG	*ORG-PER*	*ORG-LOC*
count	58,500	75,600	95,900

Table 3: Count of entity-type pairs in corpus

Parameter	*Description/ Search*	*Optimal*
$\|v_{-1}\|$	maximum number of tokens in before context	2
$\|v_0\|$	maximum number of tokens in between context	6
$\|v_1\|$	maximum number of tokens in after context	2
τ_{sim}	similarity threshold [0.6, 0.7, 0.8]	0.7
τ_{cnf}	instance confidence thresholds [0.6, 0.7, 0.8]	0.7
w_n	weights to negative extractions [0.0, 0.5, 1.0, 2.0]	0.5
w_u	weights to unknown extractions [0.0001, 0.00001]	0.0001
k_{it}	number of bootstrapping epochs	3
dim_{emb}	dimension of embedding vector, V	300
PMI	PMI threshold in evaluation	0.5
Entity Pairs	Ordered Pairs (OP) or Bisets (BS)	OP

Table 4: Hyperparameters in BREE, BRET and BREJ

3 Evaluation

3.1 Dataset and Experimental Setup

We re-run BREE (Batista et al., 2015) for **baseline** with a set of 5.5 million news articles from AFP and APW (Parker et al., 2011). We use processed dataset of 1.2 million sentences (released by BREE) containing at least two entities linked to FreebaseEasy (Bast et al., 2014). We extract four relationships: *acquired* (ORG-ORG), *founder-of* (ORG-PER), *headquartered* (ORG-LOC) and *affiliation* (ORG-PER) for Organization (ORG), Person (PER) and Location (LOC) entity types. We bootstrap relations in BREE, BRET and BREJ, each with 4 similarity measures using seed entity

	Relationships	$\#out$	P	R	$F1$	$\#out$	P	R	$F1$	$\#out$	P	R	$F1$	$\#out$	P	R	$F1$
		baseline: BREE+sim$_{match}$				**config$_2$**: BREE+sim$_{cc}^{asym}$				**config$_3$**: BREE+sim$_{cc}^{sym1}$				**config$_4$**: BREE+sim$_{cc}^{sym2}$			
BREE	acquired	2687	0.88	0.48	0.62	5771	0.88	0.66	0.76	3471	0.88	0.55	0.68	3279	0.88	0.53	0.66
	founder-of	628	0.98	0.70	0.82	9553	0.86	0.95	0.89	1532	0.94	0.84	0.89	1182	0.95	0.81	0.87
	headquartered	16786	0.62	0.80	0.69	21299	0.66	0.85	0.74	17301	0.70	0.83	0.76	9842	0.72	0.74	0.73
	affiliation	20948	0.99	0.73	0.84	27424	0.97	0.78	0.87	36797	0.95	0.82	0.88	28416	0.97	0.78	0.87
	avg	10262	0.86	0.68	0.74	16011	0.84	0.81	0.82	14475	0.87	0.76	0.80	10680	0.88	0.72	0.78
		config$_5$: BRET+sim$_{match}$				**config$_6$**: BRET+sim$_{cc}^{asym}$				**config$_7$**: BRET+sim$_{cc}^{sym1}$				**config$_8$**: BRET+sim$_{cc}^{sym2}$			
BRET	acquired	4206	0.99	0.62	0.76	15666	0.90	0.85	0.87	18273	0.87	0.86	0.87	14319	0.92	0.84	0.87
	founder-of	920	0.97	0.77	0.86	43554	0.81	0.98	0.89	41978	0.81	0.99	0.89	46453	0.81	0.99	0.89
	headquartered	3065	0.98	0.55	0.72	39267	0.68	0.92	0.78	36374	0.71	0.91	0.80	56815	0.69	0.94	0.80
	affiliation	20726	0.99	0.73	0.85	28822	0.99	0.79	0.88	44946	0.96	0.85	0.90	33938	0.97	0.81	0.89
	avg	7229	0.98	0.67	0.80	31827	0.85	0.89	0.86	35393	0.84	0.90	0.86	37881	0.85	0.90	0.86
		config$_9$: BREJ+sim$_{match}$				**config$_{10}$**: BREJ+sim$_{cc}^{asym}$				**config$_{11}$**: BREJ+sim$_{cc}^{sym1}$				**config$_{12}$**: BREJ+sim$_{cc}^{sym2}$			
BREJ	acquired	20186	0.82	**0.87**	**0.84**	35553	0.80	**0.92**	0.86	22975	0.86	0.89	0.87	22808	0.85	0.90	**0.88**
	founder-of	45005	0.81	**0.99**	**0.89**	57710	0.81	**1.00**	**0.90**	50237	0.81	0.99	0.89	45374	0.82	0.99	0.90
	headquartered	47010	0.64	**0.93**	**0.76**	66563	0.68	**0.96**	**0.80**	60495	0.68	0.94	0.79	57853	0.68	0.94	0.79
	affiliation	40959	0.96	**0.84**	**0.89**	57301	0.94	**0.88**	**0.91**	55811	0.94	0.87	0.91	51638	0.94	0.87	0.90
	avg	**38290**	0.81	**0.91**	**0.85**	**54282**	0.81	**0.94**	**0.87**	47380	0.82	0.92	0.87	44418	0.82	0.93	0.87

Table 5: Precision (P), Recall (R) and $F1$ compared to the state-of-the-art (*baseline*). $\#out$: count of output instances with cnf$(i, \Lambda, \mathcal{G}) \geqslant 0.5$. **avg**: average. **Bold** and underline: Maximum due to BREJ and sim$_{cc}$, respectively.

pairs and templates (Table 2). See Tables 3, 4 and 5 for the count of candidates, hyperparameters and different configurations, respectively.

Our evaluation is based on Bronzi et al. (2012)'s framework to estimate precision and recall of large-scale RE systems using FreebaseEasy (Bast et al., 2014). Also following Bronzi et al. (2012), we use Pointwise Mutual Information (PMI) (Turney, 2001) to evaluate our system automatically, in addition to relying on an external knowledge base. We consider only extracted relationship instances with confidence scores cnf$(i, \Lambda, \mathcal{G})$ equal or above 0.5. We follow the same approach as BREE (Batista et al., 2015) to detect the correct order of entities in a relational triple, where we try to identify the presence of passive voice using part-of-speech (POS) tags and considering any form of the verb to be, followed by a verb in the past tense or past participle, and ending in the word 'by'. We use GloVe (Pennington et al., 2014) embeddings.

3.2 Results and Comparison with baseline

Table 5 shows the experimental results in the three systems for the different relationships with *ordered* entity pairs and similarity measures (sim$_{match}$, sim$_{cc}$). Observe that BRET (config$_5$) is *precision-oriented* while BREJ (config$_9$) *recall-oriented* when compared to BREE (baseline). We see the number of output instances $\#out$ are also higher in BREJ, therefore the higher recall. The BREJ system in the different similarity configurations outperforms the baseline BREE and BRET in terms of $F1$ score. On an average for the four relations, BREJ in configurations config$_9$ and config$_{10}$ results in $F1$ that is 0.11 (0.85 vs 0.74) and 0.13 (0.87 vs 0.74) better than the baseline BREE.

We discover that sim$_{cc}$ improves $\#out$ and *recall* over sim$_{\text{match}}$ correspondingly in all three systems. Observe that sim$_{cc}$ performs better with BRET than BREE due to *non-noisy* extractors in BRET. The results suggest an alternative to the weighting scheme in sim$_{\text{match}}$ and therefore, the state-of-the-art (sim$_{cc}$) performance with the 3 parameters (w_{-1}, w_0 and w_1) ignored in bootstrap-

τ	k_{it}	$\#out$	P	R	$F1$
0.6	1	691	0.99	0.21	0.35
	2	11288	0.85	**0.79**	0.81
0.7	1	610	1.0	0.19	0.32
	2	7948	**0.93**	0.75	**0.83**
0.8	1	522	1.0	0.17	0.29
	2	2969	0.90	0.51	0.65

Table 6: Iterations (k_{it}) Vs Scores with thresholds (τ) for relation *acquired* in BREJ. τ refers to τ_{sim} and τ_{cnf}

	τ	$\#out$	P	R	$F1$	τ	$\#out$	P	R	$F1$
BREE	.60	1785	.91	.39	.55	.70	1222	.94	.31	.47
	.80	868	.95	.25	.39	.90	626	.96	.19	.32
BRET	.60	2995	.89	.51	.65	.70	1859	.90	.40	.55
	.80	1312	.91	.32	.47	.90	752	.94	.22	.35
BREJ	.60	18271	.81	.85	.83	.70	14900	.84	.83	.83
	.80	8896	.88	.75	.81	.90	5158	.93	.65	.77

Table 7: Comparative analysis using different thresholds τ to evaluate the extracted instances for *acquired*

BREX	*acquired*			*founder-of*			*headquartered*			*affiliation*		
	E	**T**	**J**	**E**	**T**	**J**	**E**	**T**	**J**	**E**	**T**	**J**
$\#hit$	71	682	743	135	956	1042	715	3447	4023	603	14888	15052

Table 8: Disjunctive matching of Instances. $\#hit$: the count of instances matched to positive seeds in $k_{it} = 1$

Attributes		$\|\Lambda\|$	*AIE*	*AES*	*ANE*	*ANNE*	*ANNLC*	*AP*	*AN*	*ANP*
acquired	BREE	167	12.7	0.51	0.84	0.16	0.14	37.7	93.1	2.46
	BRET	17	305.2	1.00	0.11	0.89	0.00	671.8	0.12	0.00
	BREJ	555	**41.6**	**0.74**	**0.71**	**0.29**	**0.03**	**313.2**	**44.8**	**0.14**
founder-of	BREE	8	13.3	0.46	0.75	0.25	0.12	44.9	600.5	13.37
	BRET	5	179.0	1.00	0.00	1.00	0.00	372.2	0.0	0.00
	BREJ	492	**109.1**	**0.90**	0.94	0.06	**0.00**	**451.8**	**79.5**	**0.18**
headquartered	BREE	655	18.4	0.60	0.97	0.03	0.02	46.3	82.7	1.78
	BRET	7	365.7	1.00	0.00	1.00	0.00	848.6	0.0	0.00
	BREJ	1311	**45.5**	**0.80**	0.98	0.02	**0.00**	**324.1**	**77.5**	**0.24**
affiliation	BREE	198	99.7	0.55	0.25	0.75	0.34	240.5	152.2	0.63
	BRET	19	846.9	1.00	0.00	1.00	0.00	2137.0	0.0	0.00
	BREJ	470	**130.2**	**0.72**	**0.21**	**0.79**	**0.06**	**567.6**	**122.7**	**0.22**

Table 9: Analyzing the attributes of extractors Λ learned for each relationship. Attributes are: number of extractors ($|\Lambda|$), avg number of instances in Λ (AIE), avg Λ score (AES), avg number of noisy Λ (ANE), avg number of non-noisy Λ (ANNE), avg number of Λ_{NNLC} below confidence 0.5 (ANNLC), avg number of positives (AP) and negatives (AN), ratio of AN to AP (ANP). The **bold** indicates comparison of BREE and BREJ with sim_{match}. avg: average

ping. Observe that sim_{cc}^{asym} gives higher recall than the two symmetric similarity measures.

Table 6 shows the performance of BREJ in different iterations trained with different similarity τ_{sim} and confidence τ_{cnf} thresholds. Table 7 shows a comparative analysis of the three systems, where we consider and evaluate the extracted relationship instances at different confidence scores.

3.3 Disjunctive Seed Matching of Instances

As discussed in section 2.3, BREJ facilitates disjunctive matching of instances (line 05 Figure 3) with seed entity pairs and templates. Table 8 shows $\#hit$ in the three systems, where the higher values of $\#hit$ in BREJ conform to the desired property. Observe that some instances in BREJ are found to be matched in both the seed types.

3.4 Deep Dive into Attributes of Extractors

We analyze the extractors Λ generated in BREE, BRET and BREJ for the 4 relations to demonstrate the impact of joint bootstrapping. Table 9 shows the attributes of Λ. We manually annotate the extractors as *noisy* and *non-noisy*. We compute $ANNLC$ and the lower values in BREJ compared to BREE suggest fewer non-noisy extractors with lower confidence in BREJ due to the scaled confi-

	Relationships	$\#out$	P	R	$F1$
BREE	acquired	387	0.99	0.13	0.23
	founder-of	28	0.96	0.09	0.17
	headquartered	672	0.95	0.21	0.34
	affiliation	17516	0.99	0.68	0.80
	avg	4651	0.97	0.28	0.39
BRET	acquired	4031	1.00	0.61	0.76
	founder-of	920	0.97	0.77	0.86
	headquartered	3522	0.98	0.59	0.73
	affiliation	22062	0.99	0.74	0.85
	avg	7634	0.99	0.68	0.80
BREJ	acquired	12278	0.87	0.81	**0.84**
	founder-of	23727	0.80	0.99	**0.89**
	headquartered	38737	0.61	0.91	**0.73**
	affiliation	33203	0.98	0.81	**0.89**
	avg	26986	0.82	0.88	**0.84**

Table 10: BREX+$\text{sim}_{\text{match}}$:Scores when w_n ignored

dence scores. *ANNE* (higher), *ANNLC* (lower), *AP* (higher) and *AN* (lower) collectively indicate that BRET mostly generates *NNHC* extractors. *AP* and *AN* indicate an average of $N_+(\lambda, \mathcal{G}_l)$ (line " (i)" Figure 3) for positive and negative seeds, respectively for $\lambda \in \Lambda$ in the three systems. Observe the impact of scaling positive extractions (*AP*) in BREJ that shrink $\frac{N_+(\lambda,\mathcal{G}_n)}{N_+(\lambda,\mathcal{G}_p)}$ i.e., *ANP*. It facilitates λ_{NNLC} to boost its confidence, i.e., λ_{NNHC} in BREJ suggested by *AES* that results in higher $\#out$ and recall (Table 5, BREJ).

3.5 Weighting Negatives Vs Scaling Positives

As discussed, Table 5 shows the performance of BREE, BRET and BREJ with the parameter $w_n = 0.5$ in computing extractors' confidence $\text{cnf}(\lambda, \mathcal{G})$(Eq. 11). In other words, config_9 (Table 5) is combination of both weighted negative and scaled positive extractions. However, we also investigate ignoring $w_n(= 1.0)$ in order to demonstrate the capability of BREJ with only scaling positives and without weighting negatives. In Table 10, observe that BREJ outperformed both BREE and BRET for all the relationships due to higher $\#out$ and recall. In addition, BREJ scores are comparable to config_9 (Table 5) suggesting that the scaling in BREJ is capable enough to remove the parameter w_n. However, the combination of both weighting negatives and scaling positives results in the state-of-the-art performance.

3.6 Qualitative Inspection of Extractors

Table 11 lists some of the non-noisy extractors (simplified) learned in different configurations to illustrate boosting extractor confidence $\text{cnf}(\lambda, \mathcal{G})$. Since, an extractor λ is a cluster of instances, therefore to simplify, we show one in-

config$_1$: BREE + sim$_{\text{match}}$	cnf$(\lambda, \mathbf{G})$	config$_5$: BRET + sim$_{\text{match}}$	cnf$(\lambda, \mathbf{G})$	config$_9$: BREJ + sim$_{\text{match}}$	cnf$(\lambda, \mathbf{G})$	config$_{10}$: BREJ + sim$_{cc}^{\text{asym}}$	cnf$(\lambda, \mathbf{G})$
acquired							
[X] acquired [Y]	0.98	[X] acquired [Y]	1.00	[X] acquired [Y]	1.00	acquired by [X] , [Y] †	0.93
[X] takeover of [Y]	0.89	[X] takeover of [Y]	1.00	[X] takeover of [Y]	0.98	takeover of [X] would boost [Y] 's earnings †	0.90
[X] 's planned acquisition of [Y]	0.87	[X] 's planned acquisition of[Y]	1.00	[X] 's planned acquisition of [Y]	0.98	acquisition of [X] by [Y] †	0.95
[X] acquiring [Y]	0.75	[X] acquiring [Y]	1.00	[X] acquiring [Y]	0.95	[X] acquiring [Y]	0.95
[X] has owned part of [Y]	0.67	[X] has owned part of [Y]	1.00	[X] has owned part of [Y]	0.88	owned by [X] 's parent [Y]	0.90
[X] took control of [Y]	0.49	[X] 's ownership of [Y]	1.00	[X] took control of [Y]	0.91	[X] takes control of [Y]	1.00
[X] 's acquisition of [Y]	0.35	[X] 's acquisition of [Y]	1.00	[X] 's acquisition of [Y]	0.95	acquisition of [X] would reduce [Y] 's share †	0.90
[X] 's merger with [Y]	0.35	[X] 's merger with[Y]	1.00	[X] 's merger with [Y]	0.94	[X] - [Y] merger between †	0.84
[X] 's bid for [Y]	0.35	[X] 's bid for [Y]	1.00	[X] 's bid for [Y]	0.97	part of [X] which [Y] acquired †	0.83
founder-of							
[X] founder [Y]	0.68	[X] founder [Y]	1.00	[X] founder [Y]	0.99	founder of [X] , [Y] †	0.97
[X] CEO and founder [Y]	0.15	[X] CEO and founder [Y]	1.00	[X] CEO and founder [Y]	0.99	co-founder of [X] 's millennial center , [Y] †	0.94
[X] 's co-founder [Y]	0.09	[X] owner [Y]	1.00	[X] owner [Y]	1.00	owned by [X] cofounder [Y]	0.95
		[X] cofounder [Y]	1.00	[X] cofounder [Y]	1.00	Gates co-founded [X] with school friend [Y] †	0.99
		[X] started by [Y]	1.00	[X] started by [Y]	1.00	who co-founded [X] with [Y] †	0.95
		[X] was founded by [Y]	1.00	[X] was founded by [Y]	0.99	to co-found [X] with partner [Y] †	0.68
		[X] begun by [Y]	1.00	[X] begun by [Y]	1.00	[X] was started by [Y] , cofounder	0.98
		[X] has established [Y]	1.00	[X] has established [Y]	0.99	set up [X] with childhood friend [Y] †	0.96
		[X] chief executive and founder [Y]	1.00	[X] co-founder and billionaire [Y] *	0.99	[X] co-founder and billionaire [Y]	0.97
headquartered							
[X] headquarters in [Y]	0.95	[X] headquarters in [Y]	1.00	[X] headquarters in [Y]	0.98	[X] headquarters in [Y]	0.98
[X] relocated its headquarters from [Y]	0.94	[X] relocated its headquarters from [Y]	1.00	[X] relocated its headquarters from [Y]	0.98	based at [X] 's suburban [Y] headquarters †	0.98
[X] head office in [Y]	0.84	[X] head office in [Y]	1.00	[X] head office in [Y]	0.87	head of [X] 's operations in [Y] †	0.65
[X] based in [Y]	0.75	[X] based in [Y]	1.00	[X] based in [Y]	0.98	branch of [X] company based in [Y]	0.98
[X] headquarters building in [Y]	0.67	[X] headquarters building in [Y]	1.00	[X] headquarters building in [Y]	0.94	[X] main campus in [Y]	0.99
[X] headquarters in downtown [Y]	0.64	[X] headquarters in downtown [Y]	1.00	[X] headquarters in downtown [Y]	0.94	[X] headquarters in downtown [Y]	0.96
[X] branch offices in [Y]	0.54	[X] branch offices in [Y]	1.00	[X] branch offices in [Y]	0.98	[X] 's [Y] headquarters represented †	0.98
[X] 's corporate campus in [Y]	0.51	[X] 's corporate campus in [Y]	1.00	[X] 's corporate campus in [Y]	0.99	[X] main campus in [Y]	0.99
[X] 's corporate office in [Y]	0.51	[X] 's corporate office in [Y]	1.00	[X] 's corporate office in [Y]	0.89	[X] , [Y] 's corporate †	0.94
affiliation							
[X] chief executive [Y]	0.92	[X] chief executive [Y]	1.00	[X] chief executive [Y]	0.97	[X] chief executive [Y] resigned monday	0.94
[X] secretary [Y]	0.88	[X] secretary [Y]	1.00	[X] secretary [Y]	0.94	worked with [X] manager [Y]	0.85
[X] president [Y]	0.87	[X] president [Y]	1.00	[X] president [Y]	0.96	[X] voted to retain [Y] as CEO †	0.98
[X] leader [Y]	0.72	[X] leader [Y]	1.00	[X] leader [Y]	0.85	head of [X] , [Y] †	0.99
[X] party leader [Y]	0.67	[X] party leader [Y]	1.00	[X] party leader [Y]	0.87	working with [X] , [Y] suggested †	1.00
[X] has appointed [Y]	0.63	[X] executive editor [Y]	1.00	[X] has appointed [Y]	0.81	[X] president [Y] was fired	0.90
[X] player [Y]	0.38	[X] player [Y]	1.00	[X] player [Y]	0.89	[X] 's [Y] was fired †	0.43
[X] 's secretary-general [Y]	0.36	[X] 's secretary-general [Y]	1.00	[X] 's secretary-general [Y]	0.93	Chairman of [X] , [Y] †	0.88
[X] hired [Y]	0.21	[X] director [Y]	1.00	[X] hired [Y]	0.56	[X] hired [Y] as manager †	0.85

Table 11: Subset of the non-noisy extractors (simplified) with their confidence scores cnf$(\lambda, \mathcal{G})$ learned in different configurations for each relation. $*$ denotes that the extractor was never learned in config$_1$ and config$_5$. † indicates that the extractor was never learned in config$_1$, config$_5$ and config$_9$. [X] and [Y] indicate placeholders for entities.

stance (mostly populated) from every λ. Each cell in Table 11 represents either a simplified representation of λ or its confidence. We demonstrate how the confidence score of a non-noisy extractor in BREE (config$_1$) is increased in BREJ (config$_9$ and config$_{10}$). For instance, for the relation *acquired*, an extractor $\{$*[X] acquiring [Y]*$\}$ is generated by BREE, BRET and BREJ; however, its confidence is boosted from 0.75 in BREE (config$_1$) to 0.95 in BREJ (config$_9$). Observe that BRET generates high confidence extractors. We also show extractors (marked by †) learned by BREJ with sim$_{cc}$ (config$_{10}$) but not by config$_1$, config$_5$ and config$_9$.

3.7 Entity Pairs: Ordered Vs Bi-Set

In Table 5, we use ordered pairs of typed entities. Additionally, we also investigate using entity sets and observe improved recall due to higher $\#out$ in both BREE and BREJ, comparing correspondingly Table 12 and 5 (*baseline* and config$_9$).

4 Conclusion

We have proposed a Joint Bootstrapping Machine for relation extraction (BREJ) that takes advantage of both entity-pair-centered and template-centered approaches. We have demonstrated that the joint approach scales up positive instances that boosts the confidence of NNLC extractors and improves recall. The experiments showed that the cross-context similarity measures improved recall and suggest removing in total four parameters.

Relationships	**BREE** + sim$_{\text{match}}$				**BREJ** + sim$_{\text{match}}$			
	$\#out$	P	R	$F1$	$\#out$	P	R	$F1$
acquired	2786	.90	.50	.64	21733	.80	.87	.83
founder-of	543	1.0	.67	.80	31890	.80	.99	.89
headquartered	16832	.62	.81	.70	52286	.64	.94	.76
affiliation	21812	.99	.74	.85	42601	.96	.85	.90
avg	10493	.88	.68	.75	37127	.80	.91	.85

Table 12: BREX+sim$_{\text{match}}$:Scores with entity *bisets*

Acknowledgments

We thank our colleagues Bernt Andrassy, Mark Buckley, Stefan Langer, Ulli Waltinger and Usama Yaseen, and anonymous reviewers for their review comments. This research was supported by Bundeswirtschaftsministerium (bmwi.de), grant 01MD15010A (Smart Data Web) at Siemens AG-CT Machine Intelligence, Munich Germany.

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the 15th ACM conference on Digital libraries*. Association for Computing Machinery, Washington, DC USA, pages 85–94.

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, Beijing, China, volume 1, pages 344–354.

Hannah Bast, Florian Bäurle, Björn Buchhold, and Elmar Haußmann. 2014. Easy access to the freebase dataset. In *Proceedings of the 23rd International Conference on World Wide Web*. Association for Computing Machinery, Seoul, Republic of Korea, pages 95–98.

David S. Batista, Bruno Martins, and Mário J. Silva. 2015. Semi-supervised bootstrapping of relationship extractors with distributional semantics. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 499–504.

Sergey Brin. 1998. Extracting patterns and relations from the world wide web. In *International Workshop on The World Wide Web and Databases*. Springer, Valencia, Spain, pages 172–183.

Mirko Bronzi, Zhaochen Guo, Filipe Mesquita, Denilson Barbosa, and Paolo Merialdo. 2012. Automatic evaluation of relation extraction systems on large-scale. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX)*. Association for Computational Linguistics, Montrèal, Canada, pages 19–24.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010. Toward an architecture for never-ending language learning. In *Proceedings of the 24th National Conference on Artificial Intelligence (AAAI)*. Atlanta, Georgia USA, volume 5, page 3.

Laura Chiticariu, Yunyao Li, and Frederick R. Reiss. 2013. Rule-based information extraction is dead! long live rule-based information extraction systems! In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington USA, pages 827–832.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Edinburgh, Scotland UK, pages 1535–1545.

Pankaj Gupta, Thomas Runkler, Heike Adel, Bernt Andrassy, Hans-Georg Zimmermann, and Hinrich Schütze. 2015. Deep learning methods for the extraction of relations in natural language text. Technical report, Technical University of Munich, Germany.

Pankaj Gupta, Hinrich Schütze, and Bernt Andrassy. 2016. Table filling multi-task recurrent neural network for joint entity and relation extraction. In *Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan, pages 2537–2547.

Sonal Gupta, Diana L. MacLean, Jeffrey Heer, and Christopher D. Manning. 2014. Induced lexico-syntactic patterns improve information extraction from online medical forums. *Journal of the American Medical Informatics Association* 21(5):902–909.

Sonal Gupta and Christopher Manning. 2014. Improved pattern learning for bootstrapped entity extraction. In *Proceedings of the 18th Conference on Computational Natural Language Learning (CoNLL)*. Association for Computational Linguistics, Baltimore, Maryland USA, pages 98–108.

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 15th International Conference on Computational Linguistics*. Nantes, France, pages 539–545.

Winston Lin, Roman Yangarber, and Ralph Grishman. 2003. Bootstrapped learning of semantic classes from positive and negative examples. In *Proceedings of ICML 2003 Workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*. Washington, DC USA, page 21.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, Jeju Island, Korea, pages 523–534.

Filipe Mesquita, Jordan Schmidek, and Denilson Barbosa. 2013. Effectiveness and efficiency of open relation extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington USA, pages 447–457.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of the Workshop at the International Conference on Learning Representations*. ICLR, Scottsdale, Arizona USA.

Thien Huu Nguyen and Ralph Grishman. 2015. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*. Association for Computational Linguistics, Denver, Colorado USA, pages 39–48.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English gigaword. *Linguistic Data Consortium* .

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Doha, Qatar, pages 1532–1543.

Ellen Riloff. 1996. Automatically generating extraction patterns from untagged text. In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI)*. Portland, Oregon USA, pages 1044–1049.

Peter D. Turney. 2001. Mining the web for synonyms: Pmi-ir versus lsa on toefl. In *Proceedings of the 12th European Conference on Machine Learning*. Springer, Freiburg, Germany, pages 491–502.

Ngoc Thang Vu, Heike Adel, Pankaj Gupta, and Hinrich Schütze. 2016a. Combining recurrent and convolutional neural networks for relation classification. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. Association for Computational Linguistics, San Diego, California USA, pages 534–539.

Ngoc Thang Vu, Pankaj Gupta, Heike Adel, and Hinrich Schütze. 2016b. Bi-directional recurrent neural network with ranking loss for spoken language understanding. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, Shanghai, China, pages 6060–6064.

A Deep Generative Model of Vowel Formant Typology

Ryan Cotterell and **Jason Eisner**

Department of Computer Science
Johns Hopkins University, Baltimore MD, 21218
{ryan.cotterell,eisner}@jhu.edu

Abstract

What makes some types of languages more probable than others? For instance, we know that almost all spoken languages contain the vowel phoneme /i/; why should that be? The field of linguistic typology seeks to answer these questions and, thereby, divine the mechanisms that underlie human language. In our work, we tackle the problem of vowel system typology, i.e., we propose a generative probability model of which vowels a language contains. In contrast to previous work, we work directly with the acoustic information—the first two formant values—rather than modeling discrete sets of phonemic symbols (IPA). We develop a novel generative probability model and report results based on a corpus of 233 languages.

1 Introduction

Human languages are far from arbitrary; cross-linguistically, they exhibit surprising similarity in many respects and many properties appear to be universally true. The field of linguistic typology seeks to investigate, describe and quantify the axes along which languages vary. One facet of language that has been the subject of heavy investigation is the nature of vowel inventories, i.e., which vowels a language contains. It is a cross-linguistic universal that all spoken languages have vowels (Gordon, 2016), and the underlying principles guiding vowel selection are understood: vowels must be both easily recognizable and well-dispersed (Schwartz et al., 2005). In this work, we offer a more formal treatment of the subject, deriving a generative probability model of vowel inventory typology. Our work builds on (Cotterell and Eisner, 2017) by investigating not just discrete IPA inventories but the cross-linguistic variation in acoustic formants.

The philosophy behind our approach is that linguistic typology should be treated probabilistically and its goal should be the construction of a universal prior over potential languages. A probabilistic approach does not rule out linguistic systems completely (as long as one's theoretical formalism can describe them at all), but it can position phenomena on a scale from very common to very improbable. Probabilistic modeling also provides a discipline for drawing conclusions from sparse data. While we know of over 7000 human languages, we have some sort of linguistic analysis for only 2300 of them (Comrie et al., 2013), and the dataset used in this paper (Becker-Kristal, 2010) provides simple vowel data for fewer than 250 languages.

Formants are the resonant frequencies of the human vocal tract during the production of speech sounds. We propose a Bayesian generative model of vowel inventories, where each language's inventory is a finite subset of acoustic vowels represented as points $(F_1, F_2) \in \mathbb{R}^2$. We deploy tools from the neural-network and point-process literatures and experiment on a dataset with 233 distinct languages. We show that our most complicated model outperforms simpler models.

2 Acoustic Phonetics and Formants

Much of human communication takes place through speech: one conversant emits a sound wave to be comprehended by a second. In this work, we consider the nature of the portions of such sound waves that correspond to vowels. We briefly review the relevant bits of acoustic phonetics so as to give an overview of the data we are actually modeling and develop our notation.

The anatomy of a sound wave. The sound wave that carries spoken language is a function from time to amplitude, describing sound pressure variation in the air. To distinguish vowels, it is helpful to transform this function into a **spectrogram** (Fig. 1) by using a short-time Fourier transform

Proceedings of NAACL-HLT 2018, pages 37–46
New Orleans, Louisiana, June 1 - 6, 2018.

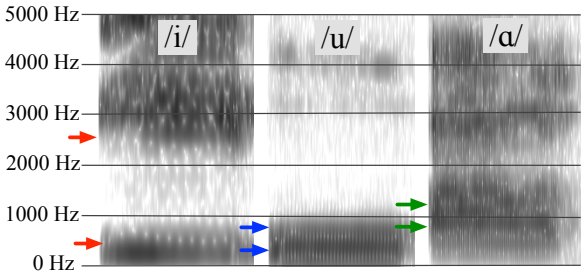


Figure 1: Example spectrogram of the three English vowels: /i/, /u/ and /ɑ/. The x-axis is time and y-axis is frequency. The first two formants F_1 and F_2 are marked in with arrows for each vowel. The figure was made with Praat (Boersma et al., 2002).

(Deng and O'Shaughnessy, 2003, Chapter 1) to decompose each short interval of the wave function into a weighted sum of sinusoidal waves of different frequencies (measured in Hz). At each interval, the variable darkness of the spectrogram indicates the weights of the different frequencies. In phonetic analysis, a common quantity to consider is a **formant**—a local maximum of the (smoothed) frequency spectrum. The fundamental frequency F_0 determines the pitch of the sound. The formants F_1 and F_2 determine the quality of the vowel.

Two is all you need (and what we left out). In terms of vowel recognition, it is widely speculated that humans rely almost exclusively on the first two formants of the sound wave (Ladefoged, 2001, Chapter 5). The two-formant assumption breaks down in edge cases: e.g., the third formant F_3 helps to distinguish the roundness of the vowel (Ladefoged, 2001, Chapter 5). Other non-formant features may also play a role. For example, in tonal languages, the same vowel may be realized with different tones (which are signaled using F_0): Mandarin Chinese makes a distinction between mǎ (*horse*) and má (*hemp*) without modifying the quality of the vowel /a/. Other features, such as creaky voice, can play a role in distinguishing phonemes. We do not explicitly model any of these aspects of vowel space, limiting ourselves to (F_1, F_2) as in previous work (Liljencrants and Lindblom, 1972). However, it would be easy to extend all the models we will propose here to incorporate such information, given appropriate datasets.

3 The Phonology of Vowel Systems

The vowel inventories of the world's languages display clear structure and appear to obey several underlying principles. The most prevalent of these principles are **focalization** and **dispersion**.

Focalization. The notion of focalization grew out of quantal vowel theory (Stevens, 1989). Quantal vowels are those that are phonetically "better" than others. They tend to display certain properties, e.g., the formants tend to be closer together (Stevens, 1987). Cross-linguistically, quantal vowels are the most frequently attested vowels, e.g., the cross-linguistically common vowel /i/ is considered quantal, but less common /y/ is not.

Dispersion. The second core principle of vowel system organization is known as dispersion. As the name would imply, the principle states that the vowels in "good" vowel systems tend to be spread out. The motivation for such a principle is clear—a well-dispersed set of vowels reduces a listener's potential confusion over which vowel is being pronounced. See Schwartz et al. (1997) for a review of dispersion in vowel system typology and its interaction with focalization, which has led to the joint dispersion-focalization theory.

Notation. We will denote the universal set of international phonetic alphabet (IPA) symbols as $\mathcal{V}$. The observed vowel inventory for language ℓ has size n^ℓ and is denoted $V^\ell = \{(v_1^\ell, \mathbf{v}_1^\ell), \ldots, (v_{n^\ell}^\ell, \mathbf{v}_{n^\ell}^\ell)\} \subseteq \mathcal{V} \times \mathbb{R}^d$, where for each $k \in [1, n^\ell]$, $v_k^\ell \in \mathcal{V}$ is an IPA symbol assigned by a linguist and $\mathbf{v}_k^\ell \in \mathbb{R}^d$ is a vector of d measurable phonetic quantities. In short, the IPA symbol v_k^ℓ was assigned as a label for a phoneme with pronunciation $\mathbf{v}_k^\ell$. The ordering of the elements within V^ℓ is arbitrary.

Goals. This framework recognizes that the same IPA symbol v (such as /u/) may represent a slightly different sound $\mathbf{v}$ in one language than in another, although they are transcribed identically. We are specifically interested in how the vowels in a language influence one another's fine-grained pronunciation in $\mathbb{R}^d$. In general, there is no reason to suspect that speakers of two languages, whose phonological systems contain the same IPA symbol, should produce that vowel with identical formants.

Data. For the remainder of the paper, we will take $d = 2$ so that each $\mathbf{v} = (F_1, F_2) \in \mathbb{R}^2$, the vector consisting of the first two formant values, as compiled from the field literature by Becker-Kristal (2006). This dataset provides inventories V^ℓ in the form above. Thus, we do not consider further variation of the vowel pronunciation that

may occur within the language (between speakers, between tokens of the vowel, or between earlier and later intervals within a token).

4 Phonemes versus Phones

Previous work (Cotterell and Eisner, 2017) has placed a distribution over discrete phonemes, ignoring the variation across languages in the *pronunciation* of each phoneme. In this paper, we crack open the phoneme abstraction, moving to a learned set of finer-grained phones.

Cotterell and Eisner (2017) proposed (among other options) using a *determinantal point process* (DPP) over a universal inventory $\mathcal{V}$ of 53 symbolic (IPA) vowels. A draw from such a DPP is a language-specific inventory of vowel *phonemes*, $V \subseteq \mathcal{V}$. In this paper, we say that a language instead draws its inventory from a larger set $\bar{\mathcal{V}}$, again using a DPP. In both cases, the reason to use a DPP is that it prefers relatively diverse inventories whose individual elements are relatively quantal.

While we could in principle identify $\bar{\mathcal{V}}$ with $\mathbb{R}^d$, for convenience we still take it to be a (large) discrete finite set $\bar{\mathcal{V}} = \{\bar{v}_1, \ldots, \bar{v}_N\}$, whose elements we call *phones*. $\bar{\mathcal{V}}$ is a learned cross-linguistic parameter of our model; thus, its elements—the "universal phones"—may or may not correspond to phonetic categories traditionally used by linguists.

We presume that language ℓ draws from the DPP a subset $\bar{V}^\ell \subseteq \bar{\mathcal{V}}$, whose size we call n^ℓ. For each universal phone $\bar{v}_i$ that appears in this inventory $\bar{V}^\ell$, the language then draws an observable language-specific pronunciation $\mathbf{v}_i^\ell \sim \mathcal{N}\left(\boldsymbol{\mu}_i, \sigma^2 I\right)$ from a distribution associated cross-linguistically with the universal phone $\bar{v}_i$. We now have an inventory of pronunciations.

As a final step in generating the vowel inventory, we could model IPA labels. For each $\bar{v}_i \in \bar{V}^\ell$, a field linguist presumably draws the IPA label v_i^ℓ conditioned on all the pronunciations $\{\mathbf{v}_i^\ell \in \mathbb{R}^d : \bar{v}_i \in \bar{V}^\ell\}$ in the inventory (and perhaps also on their underlying phones $\bar{v}_i \in \bar{V}^\ell$). This labeling process may be complex. While each pronunciation in $\mathbb{R}^d$ (or each underlying phone in $\bar{\mathcal{V}}$) may have a preference for certain IPA labels in $\mathcal{V}$, the n^ℓ labels must be drawn jointly because the linguist will take care not to use the same label for two phones, and also because the linguist may like to describe the inventory using a small number of distinct IPA features, which will tend to favor factorial grids of symbols. The linguist's use of IPA features may also be informed by phonological and phonetic processes in the language. We leave modeling of this step to future work; so our current likelihood term ignores the evidence contributed by the IPA labels in the dataset, considering only the pronunciations in $\mathbb{R}^d$.

The overall idea is that human languages ℓ draw their inventories from some universal prior, which we are attempting to reconstruct. A caveat is that we will train our method by maximum-likelihood, which does not quantify our uncertainty about the reconstructed parameters. An additional caveat is that some languages in our dataset are related to one another, which belies the idea that they were drawn independently. Ideally, one ought to capture these relationships using hierarchical or evolutionary modeling techniques.

5 Determinantal Point Processes

Before delving into our generative model, we briefly review technical background used by Cotterell and Eisner (2017). A DPP is a probability distribution over the subsets of a *fixed ground set* of size N—in our case, the set of phones $\bar{\mathcal{V}}$. The DPP is usually given as an L-ensemble (Borodin and Rains, 2005), meaning that it is parameterized by a positive semi-definite matrix $L \in \mathbb{R}^{N \times N}$. Given a discrete base set $\bar{\mathcal{V}}$ of phones, the probability of a subset $\bar{V} \subseteq \bar{\mathcal{V}}$ is given by

$$p(\bar{V}) \propto \det\left(L_{\bar{V}}\right), \tag{1}$$

where $L_{\bar{V}}$ is the submatrix of L corresponding to the rows and columns associated with the subset $\bar{V} \subseteq \bar{\mathcal{V}}$. The entry L_{ij}, where $i \neq j$, has the effect of describing the similarity between the elements $\bar{v}_i$ and $\bar{v}_j$ (both in $\bar{\mathcal{V}}$)—an ingredient needed to model dispersion. And, the entry L_{ii} describes the quality—focalization—of the vowel $\bar{v}_i$, i.e., how much the model wants to have $\bar{v}_i$ in a sampled set independent of the other members.

5.1 Probability Kernel

In this work, each phone $\bar{v}_i \in \bar{\mathcal{V}}$ is associated with a probability density over the space of possible pronunciations $\mathbb{R}^2$. Our measure of phone similarity will consider the "overlap" between the densities associated with two phones. This works as follows: Given two densities $f(x, y)$ and $f'(x, y)$ over $\mathbb{R}^2$, we define the kernel (Jebara et al., 2004) as

$$\mathcal{K}(f, f'; \rho) = \int_x \int_y f(x, y)^\rho f'(x, y)^\rho dx\, dy, \tag{3}$$

$$\prod_{\ell=1}^{M} \Big[p(\mathbf{v}^{\ell,1}, \ldots, \mathbf{v}^{\ell,n^\ell} \mid \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_N, N)\Big] p(\boldsymbol{\mu}_1, \ldots \boldsymbol{\mu}_N \mid N)\, p(N) \tag{2}$$

$$= \prod_{\ell=1}^{M} \left[\sum_{\mathbf{a}^\ell \in A(n^\ell, N)} \left(\prod_{k=1}^{n^\ell} \underbrace{p(\mathbf{v}^{\ell,k} \mid \boldsymbol{\mu}_{a_k^\ell})}_{④} \right) \underbrace{p(\bar{V}(\mathbf{a}^\ell) \mid \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_N, N)}_{③} \right] \underbrace{p(\boldsymbol{\mu}_1, \ldots \boldsymbol{\mu}_N \mid N)}_{②} \underbrace{p(N)}_{①}$$

Figure 2: Joint likelihood of M vowel systems under our deep generative probability model for continuous-space vowel inventories. Here language ℓ has an observed inventory of pronunciations $\{\mathbf{v}^{\ell,k} : 1 \leq k \leq n^\ell\}$, and $a_k^\ell \in [1, N]$ denotes a phone that might be responsible for the pronunciation $\mathbf{v}^{\ell,k}$. Thus, $\mathbf{a}^\ell$ denotes some way to jointly label all n^ℓ pronunciations with distinct phones. We must sum over all $\binom{N}{n^\ell}$ such labelings $\mathbf{a}^\ell \in A(n^\ell, N)$ since the true labeling is not observed. In other words, we sum over all ways $\mathbf{a}^\ell$ of completing the data for language ℓ. Within each summand, the product of factors 3 and 4 is the probability of the completed data, i.e., the joint probability of generating the inventory $\bar{V}(\mathbf{a}^\ell)$ of phones used in the labeling and their associated pronunciations. Factor 3 considers the prior probability of $\bar{V}(\mathbf{a}^\ell)$ under the DPP, and factor 4 is a likelihood term that considers the probability of the associated pronunciations.

with inverse temperature parameter ρ.

In our setting, f, f' will both be Gaussian distributions with means $\boldsymbol{\mu}$ and $\boldsymbol{\mu}'$ that share a fixed spherical covariance matrix $\sigma^2 I$. Then eq. (3) and indeed its generalization to any $\mathbb{R}^d$ has a closed-form solution (Jebara et al., 2004, §3.1):

$$\mathcal{K}(f, f'; \rho) = \tag{4}$$
$$(2\rho)^{\frac{d}{2}} \left(2\pi\sigma^2\right)^{\frac{(1-2\rho)d}{2}} \exp\left(-\frac{\rho||\boldsymbol{\mu} - \boldsymbol{\mu}'||^2}{4\sigma^2}\right).$$

Notice that making ρ small (i.e., high temperature) has an effect on (4) similar to scaling the variance σ^2 by the temperature, but it also results in changing the scale of $\mathcal{K}$, which affects the balance between dispersion and focalization in (6) below.

5.2 Focalization Score

The probability kernel given in eq. (3) naturally handles the linguistic notion of dispersion. What about focalization? We say that a phone is focal to the extent that it has a high score

$$F(\boldsymbol{\mu}) = \exp\left(U_2 \tanh(U_1 \boldsymbol{\mu} + \mathbf{b}_1) + \mathbf{b}_2\right) > 0 \tag{5}$$

where $\boldsymbol{\mu}$ is the mean of its density. To learn the parameters of this neural network from data is to learn which phones are focal. We use a neural network since the focal regions of $\mathbb{R}^2$ are distributed in a complex way.

5.3 The L Matrix

If $f_i = \mathcal{N}(\boldsymbol{\mu}_i, \sigma^2 I)$ is the density associated with the phone $\bar{v}_i$, we may populate an $N \times N$ real matrix L where

$$L_{ij} = \begin{cases} \mathcal{K}(f_i, f_j; \rho) & \text{if } i \neq j \\ \mathcal{K}(f_i, f_j; \rho) + F(\boldsymbol{\mu}_i) & \text{if } i = j \end{cases} \tag{6}$$

Since L is the sum of two positive definite matrices (the first specializes a known kernel and the second is diagonal and positive), it is also positive definite. As a result, it can be used to parameterize a DPP over $\bar{\mathcal{V}}$. Indeed, since L is positive definite and not merely positive semidefinite, it will assign positive probability to *any* subset of $\bar{\mathcal{V}}$.

As previously noted, this DPP does not define a distribution over an infinite set, e.g., the powerset of $\mathbb{R}^2$, as does recent work on continuous DPPs (Affandi et al., 2013). Rather, it defines a distribution over the powerset of a *set of densities with finite cardinality*. Once we have sampled a subset of densities, a real-valued quantity may be additionally sampled from each sampled density.

Algorithm 1 Generative Process

```
N ~ Poisson(λ) (∈ ℕ)                                   ①
for i = 1 to N :
    μ_i ~ 𝒩(0, I) (∈ ℝ^2)                              ②
define L ∈ ℝ^{N×N} via (6)
for ℓ = 1 to M :
    V̄^ℓ ~ DPP(L) (⊆ [1, N]); let n^ℓ = |V̄^ℓ|           ③
    for i ∈ V̄^ℓ :
        ṽ_i^ℓ ~ 𝒩(μ_i, σ^2 I)                          ④
        v_i^ℓ = ν_θ(ṽ_i^ℓ)                             ④
```

6 A Deep Generative Model

We are now in a position to expound our generative model of continuous-space vowel typology. We

generate a set of formant pairs for M languages in a four step process. Note that throughout this exposition, language-specific quantities with be superscripted with an integral language marker ℓ, whereas universal quantities are left unsuperscripted. The generative process is written in algorithmic form in Alg. 1. Note that each step is numbered and color-coded for ease of comparison with the full joint likelihood in Fig. 2.

Step ①: $p(N)$. We sample the size N of the universal phone inventory $\bar{\mathcal{V}}$ from a Poisson distribution with a rate parameter λ, i.e.,

$$N \sim \text{Poisson}\,(\lambda)\,. \tag{7}$$

That is, we do not presuppose a certain number of phones in the model.

Step ②: $p(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_N)$. Next, we sample the means $\boldsymbol{\mu}_i$ of the Gaussian phones. In the model presented here, we assume that each phone is generated independently, so $p(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_N) = \prod_{i=1}^{N} p(\boldsymbol{\mu}_i)$. Also, we assume a standard Gaussian prior over the means, $\boldsymbol{\mu}_i \sim \mathcal{N}(\mathbf{0}, I)$.

The sampled means define our N Gaussian phones $\mathcal{N}\left(\boldsymbol{\mu}_i, \sigma^2 I\right)$: we are assuming for simplicity that all phones share a *single* spherical covariance matrix, defined by the hyperparameter σ^2. The dispersion and focalization of these phones define the matrix L according to equations (4)–(6), where ρ in (4) and the weights of the focalization neural net (5) are also hyperparameters.

Step ③: $p(\bar{V}^\ell \mid \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_N)$. Next, for each language $\ell \in [1, \ldots, M]$, we sample a diverse subset of the N phones, via a single draw from a DPP parameterized by matrix L:

$$\bar{V}^\ell \sim \text{DPP}(L), \tag{8}$$

where $\bar{V}^\ell \subseteq [1, N]$. Thus, $i \in \bar{V}^\ell$ means that language ℓ contains phone $\bar{v}_i$. Note that even the size of the inventory, $n^\ell = |\bar{V}^\ell|$, was chosen by the DPP. In general, we have $n^\ell \ll N$.

Step ④: $\prod_{i \in \bar{V}^\ell} p(\mathbf{v}_i^\ell \mid \boldsymbol{\mu}_i)$ The final step in our generative process is that the phones $\bar{v}_i$ in language ℓ must generate the pronunciations $\mathbf{v}_i^\ell \in \mathbb{R}^2$ (formant vectors) that are actually observed in language ℓ. Each vector takes two steps. For each $i \in \bar{V}^\ell$, we generate an underlying $\tilde{\mathbf{v}}_i \in \mathbb{R}^2$ from the corresponding Gaussian phone. Then, we run this vector through a feed-forward neural network $\nu_{\boldsymbol{\theta}}$ with parameters $\boldsymbol{\theta}$. In short:

$$\tilde{\mathbf{v}}_i^\ell \sim \mathcal{N}(\boldsymbol{\mu}_i, \sigma^2 I) \tag{9}$$

$$\mathbf{v}_i^\ell = \nu_{\boldsymbol{\theta}}(\tilde{\mathbf{v}}_i^\ell), \tag{10}$$

where the second step is deterministic. We can fuse these two steps into a single step $p(\mathbf{v}_i \mid \boldsymbol{\mu}_i)$, whose closed-form density is given in eq. (12) below. In effect, step 4 takes a Gaussian phone as input and produces the observed formant vector with an *underlying* formant vector in the middle.

This completes our generative process. We do not observe all the steps, but only the final collection of pronunciations $\mathbf{v}_i^\ell$ for each language, where the subscripts i that indicate phone identity have been lost. The probability of this incomplete dataset involves summing over possible phones for each pronunciation, and is presented in Fig. 2.

6.1 A Neural Transformation of a Gaussian

A crucial bit of our model is running a sample from a Gaussian through a neural network. Under certain restrictions, we can find a closed form for the resulting density; we discuss these below. Let $\nu_{\boldsymbol{\theta}}$ be a depth-2 multi-layer perceptron

$$\nu_{\boldsymbol{\theta}}(\tilde{\mathbf{v}}_i) = W_2 \tanh\left(W_1 \tilde{\mathbf{v}}_i + \mathbf{b}_1\right) + \mathbf{b}_2. \tag{11}$$

In order to find a closed-form solution, we require that (5) be a diffeomorphism, i.e., an invertible mapping from $\mathbb{R}^2 \to \mathbb{R}^2$ where both $\nu_{\boldsymbol{\theta}}$ and its inverse $\nu_{\boldsymbol{\theta}}^{-1}$ are differentiable. This will be true as long as $W_1, W_2 \in \mathbb{R}^{2\times 2}$ are square matrices of full-rank and we choose a smooth, invertible activation function, such as $\tanh$. Under those conditions, we may apply the standard theorem for transforming a random variable (see Stark and Woods, 2011):

$$\begin{aligned} p(\mathbf{v}_i \mid \boldsymbol{\mu}_i) &= p(\nu_{\boldsymbol{\theta}}^{-1}(\mathbf{v}_i) \mid \boldsymbol{\mu}_i)\ \det J_{\nu_{\boldsymbol{\theta}}^{-1}(\mathbf{v}_i)} \\ &= p(\tilde{\mathbf{v}}_i \mid \boldsymbol{\mu}_i)\ \det J_{\nu_{\boldsymbol{\theta}}^{-1}(\mathbf{v}_i)} \end{aligned} \tag{12}$$

where $J_{\nu_{\boldsymbol{\theta}}^{-1}(\mathbf{x})}$ is the Jacobian of the inverse of the neural network at the point $\mathbf{x}$. Recall that $p(\tilde{\mathbf{v}}_i \mid \boldsymbol{\mu}_i)$ is Gaussian-distributed.

7 Modeling Assumptions

Imbued in our generative story are a number of assumptions about the linguistic processes behind vowel inventories. We briefly draw connections between our theory and the linguistics literature.

Why underlying phones? A technical assumption of our model is the existence of a universal set of underlying phones. Each phone is equipped with a probability distribution over reported acoustic measurements (pronunciations), to allow for a single phone to account for multiple slightly different pronunciations in different languages (though never in the same language). This distribution can capture both actual interlingual variation and also random noise in the measurement process.

While our universal phones may seem to resemble the universal IPA symbols used in phonological transcription, they lack the rich featural specifications of such phonemes. A phone in our model has no features other than its mean position, which wholly determines its behavior. Our universal phones are not a substantive linguistic hypothesis, but are essentially just a way of partitioning $\mathbb{R}^2$ into finitely many small regions whose similarity and focalization can be precomputed. This technical trick allows us to use a discrete rather than a continuous DPP over the $\mathbb{R}^2$ space.[1]

Why a neural network? Our phones are Gaussians of spherical variance σ^2, presumed to be scattered with variance 1 about a two-dimensional *latent* vowel space. Distances in this latent space are used to compute the dissimilarity of phones for modeling dispersion, and also to describe the phone's ability to vary across languages. That is, two phones that are *distant* in the latent space can appear in the same inventory—presumably they are easy to discriminate in both perception and articulation—and it is easy to choose which one better explains an acoustic measurement, thereby affecting the other measurements that may appear in the inventory.

We relate this *latent* space to measurable acoustic space by a learned diffeomorphism $\nu_{\boldsymbol{\theta}}$ (Cotterell and Eisner, 2017). $\nu_{\boldsymbol{\theta}}^{-1}$ can be regarded as warping the acoustic distances into perceptual/articulatory distances. In some "high-resolution" regions of acoustic space, phones with fairly similar (F_1, F_2) values might yet be far apart in the latent space. Conversely, in other regions, relatively large acoustic changes in some direction might not prevent two phones from acting as similar or two pronunciations from being attributed to the same phone. In general, a unit circle of radius σ in latent space may be mapped by $\nu_{\boldsymbol{\theta}}$ to an oddly shaped connected region in acoustic space, and a Gaussian in latent space may be mapped to a multimodal distribution.

[1] Indeed, we could have simply taken our universal phone set to be a huge set of tiny, regularly spaced overlapping Gaussians that "covered" (say) the unit circle. As a computational matter, we instead opted to use a smaller set of Gaussians, giving the learner the freedom to infer their positions and tune their variance σ^2. Because of this freedom, this set should not be too large, or a MAP learner may overfit the training data with zero-variance Gaussians and be unable to explain the test languages—similar to overfitting a Gaussian mixture model.

8 Inference and Learning

We fit our model via MAP-EM (Dempster et al., 1977). The E-step involves deciding which phones each language has. To achieve this, we fashion a Gibbs sampler (Geman and Geman, 1984), yielding a Markov-Chain Monte Carlo E-step (Levine and Casella, 2001).

8.1 Inference: MCMC E-Step

Inference in our model is intractable even when the phones $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_N$ are fixed. Given a language with n vowels, we have to determine which subset of the N phones best explains those vowels. As discussed above, the alignment $\mathbf{a}$ between the n vowels and n of the N phones represents a latent variable. Marginalizing it out is #P-hard, as we can see that it is equivalent to summing over all bipartite matchings in a weighted graph, which, in turn, is as costly as computing the permanent of a matrix (Valiant, 1979). Our sampler[2] is an approximation algorithm for the task. We are interested in sampling $\mathbf{a}$, the labeling of observed vowels with universal phones. Note that this implicitly samples the language's phone inventory $\bar{V}(\mathbf{a})$, which is fully determined by $\mathbf{a}$.

Specifically, we employ an MCMC method closely related to Gibbs sampling. At each step of the sampler, we update our vowel-phone alignment $\mathbf{a}^\ell$ as follows. Choose a language ℓ and a vowel index $k \in [1, n^\ell]$, and let $i = a_k^\ell$ (that is, pronunciation $\mathbf{v}^{\ell,k}$ is currently labeled with universal phone $\bar{v}_i$). We will consider changing a_k^ℓ to j, where j is drawn from the $(N - n^\ell)$ phones that do *not* appear in $\bar{V}(\mathbf{a}^\ell)$, heuristically choosing j in proportion to the likelihood $p(\mathbf{v}^{\ell,k} \mid \boldsymbol{\mu}_j)$. We then stochastically decide whether to keep $a_k^\ell = i$ or set $a_k^\ell = j$ in proportion to the resulting values of the product (4) · (3) in eq. (2).

For a single E-step, the Gibbs sampler "warm-starts" with the labeling from the end of the previous iteration's E-step. It sweeps $S = 5$ times

[2] Taken from Volkovs and Zemel (2012, 3.1).

through all vowels for all languages, and returns S sampled labelings, one from the end of each sweep.

We are also interested in automatically choosing the number of phones N, for which we take the Poisson's rate parameter $\lambda = 100$. To this end, we employ reversible-jump MCMC (Green, 1995), resampling N at the start of every E-step.

8.2 Learning: M-Step

Given the set of sampled alignments provided by the E-step, our M-step consists of optimizing the log-likelihood of the now-complete training data using the inferred latent variables. We achieved this through SGD training of the diffeomorphism parameters $\boldsymbol{\theta}$, the means $\boldsymbol{\mu}_i$ of the Gaussian phones, and the parameters of the focalization kernel $\mathcal{F}$.

9 Experiments

9.1 Data

Our data is taken from the Becker-Kristal corpus (Becker-Kristal, 2006), which is a compilation of various phonetic studies and forms the largest multilingual phonetic database. Each entry in the corpus corresponds to a linguist's phonetic description of a language's vowel system: an inventory consisting of IPA symbols where each symbol is associated with two or more formant values. The corpus contains data from 233 distinct languages. When multiple inventories were available for the same language (due to various studies in the literature), we selected one at random and discarded the others.

9.2 Baselines

Baseline #1: Removing dispersion. The key technical innovation in our work lies in the incorporation of a DPP into a generative model of vowel formants—a continuous-valued quantity. The role of the DPP was to model the linguistic principle of dispersion—we may cripple this portion of our model, e.g., by forcing $\mathcal{K}$ to be a diagonal kernel, i.e., $K_{ij} = 0$ for $i \neq j$. In this case the DPP becomes a Bernoulli Point Process (BPP)—a special case of the DPP. Since dispersion is widely accepted to be an important principle governing naturally occurring vowel systems, we expect a system trained without such knowledge to perform worse.

Baseline #2: Removing the neural network $\nu_{\boldsymbol{\theta}}$. Another question we may ask of our formulation is whether we actually need a fancy neural mapping $\nu_{\boldsymbol{\theta}}$ to model our typological data well. The human perceptual system is known to perform a non-linear transformation on acoustic signals, starting with the non-linear cochlear transform that is physically performed in the ear. While $\nu_{\boldsymbol{\theta}}^{-1}$ is intended as loosely analogous, we determine its benefit by removing eq. (10) from our generative story, i.e., we take the observed formants $\mathbf{v}_k$ to arise directly from the Gaussian phones.

Baseline #3: Supervised phones and alignments. A final baseline we consider is *supervised* phones. Linguists standardly employ a finite set of phones—symbols from the international phonetic alphabet (IPA). In phonetic annotation, it is common to map each sound in a language back to this universal discrete alphabet. Under such an annotation scheme, it is easy to discern, cross-linguistically, which vowels originate from the same phoneme: an /ɪ/ in German may be roughly equated with an /ɪ/ in English. However, it is not clear how consistent this annotation truly is. There are several reasons to expect high-variance in the cross-linguistic acoustic signal. First, IPA symbols are primarily useful for interlinked phonological distinctions, i.e., one applies the symbol /ɪ/ to distinguish it from /i/ in the given language, rather than to associate it with the sound bearing the same symbol in a second language. Second, field linguists often resort to the closest common IPA symbol, rather than an exact match: if a language makes no distinction between /i/ and /ɪ/, it is more common to denote the sound with a /i/. Thus, IPA may not be as universal as hoped. Our dataset contains 50 IPA symbols so this baseline is only reported for $N = 50$.

9.3 Evaluation

Evaluation in our setting is tricky. The scientific goal of our work is to place a bit of linguistic theory on a firm probabilistic footing, rather than a downstream engineering-task, whose performance we could measure. We consider three metrics.

Cross-Entropy. Our first evaluation metric is cross-entropy: the average negative log-probability of the vowel systems in held-out test data, given the universal inventory of N phones that we trained through EM. We find this to be the cleanest method for scientific evaluation—it is the metric of optimization and has a clear interpretation: how surprised was the model to see the vowel systems of held-out, but attested, languages?

The cross-entropy is the negative log of the $\prod \left[\cdots \right]$ expression in eq. (2), with ℓ now rang-

N	metric	DPP+$\nu_{\boldsymbol{\theta}}$	BPP+$\nu_{\boldsymbol{\theta}}$	DPP−$\nu_{\boldsymbol{\theta}}$	Sup.
15	x-ent	540.02	540.05	600.34	✗
	cloze1	5.76	5.76	6.53	✗
	cloze12	4.89	4.89	5.24	✗
25	x-ent	280.47	275.36	335.36	✗
	cloze1	5.04	5.25	6.23	✗
	cloze12	4.76	4.97	5.43	✗
50	x-ent	222.85	231.70	320.05	1610.37
	cloze1	3.38	3.16	4.02	4.96
	cloze12	2.73	2.93	3.04	6.95
57	x-ent	212.14	220.42	380.31	✗
	cloze1	2.21	3.08	3.25	✗
	cloze12	2.01	3.05	3.41	✗
100	x-ent	271.95	301.45	380.02	✗
	cloze1	2.26	2.42	3.03	✗
	cloze12	1.96	2.01	2.51	✗

Table 1: Cross-entropy in nats per language (lower is better) and expected Euclidean-distance error of the cloze prediction (lower is better). The overall best value for each task is bold-faced. The case $N = 50$ is compared against our supervised baseline. The $N = 57$ row is the case where we allowed N to fluctuate during inference using reversible-jump MCMC; this was the N value selected at the final EM iteration.

ing over held-out languages.[3] Wallach et al. (2009) give several methods for estimating the intractable sum in language ℓ. We use the simple harmonic mean estimator, based on 50 samples of $\mathbf{a}^\ell$ drawn with our Gibbs sampler (warm-started from the final E-step of training).

Cloze Evaluation. In addition, following Cotterell and Eisner (2017), we evaluate our trained model's ability to perform a cloze task (Taylor, 1953). Given $n^\ell - 1$ or $n^\ell - 2$ of the vowels in held-out language ℓ, can we predict the pronunciations $\mathbf{v}_k$ of the remaining 1 or 2? We predict $\mathbf{v}_k$ to be $\nu_{\boldsymbol{\theta}}(\boldsymbol{\mu}_i)$ where $i = a_k^\ell$ is the phone inferred by the sampler. Note that the sampler's inference here is based only on the observed vowels (the likelihood) and the focalization-dispersion preferences of the DPP (the prior). We report the expected error of such a prediction—where error is quantified by Euclidean distance in (F_1, F_2) formant space—over the same 50 samples of $\mathbf{a}^\ell$.

For instance, consider a previously unseen vowel system with formant values $\{(499, 2199), (861, 1420), (571, 1079)\}$. A "cloze1" evaluation would aim to predict $\{(499, 2199)\}$ as the missing vowel, given $\{(861, 1420), (571, 1079)\}$, and the fact that $n^\ell = 3$. A "cloze12" evaluation would aim to predict two missing vowels.

[3] Since that expression is the product of both probability distributions and probability densities, our "cross-entropy" metric is actually the sum of both entropy terms and (potentially negative) differential entropy terms. Thus, a value of 0 has no special significance.

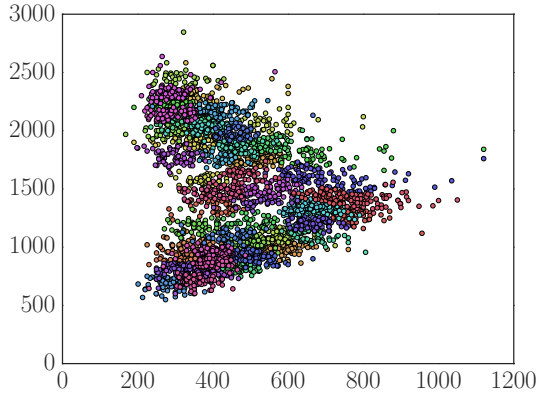


Figure 3: A graph of $\mathbf{v} = (F_1, F_2)$ in the union of all the training languages' inventories, color-coded by inferred phone ($N = 50$).

9.4 Experimental Details

Here, we report experimental details and the hyperparameters that we use to achieve the results reported. We consider a neural network $\nu_{\boldsymbol{\theta}}$ with $k \in [1, 4]$ layers and find $k = 1$ the best performer on development data. Recall that our *diffeomorphism* constraint requires that each layer have exactly two hidden units, the same as the number of observed formants. We consider $N \in \{15, 25, 50, 100\}$ phones as well as letting N fluctuate with reversible-jump MCMC (see footnote 1). We train for 100 iterations of EM, taking $S = 5$ samples at each E-step. At each M-step, we run 50 iterations of SGD for the focalization NN and also for the diffeomorphism NN. For each N, we selected (σ^2, ρ) by minimizing cross-entropy on a held-out development set. We considered $(\sigma^2, \rho) \in \{10^k\}_{k=1}^5 \times \{\rho^k\}_{k=1}^5$.

9.5 Results and Error Analysis

We report results in Tab. 1. We find that our DPP model improves over the baselines. The results support two claims: (i) dispersion plays an important role in the structure of vowel systems and (ii) learning a non-linear transformation of a Gaussian improves our ability to model sets of formant-pairs. Also, we observe that as we increase the number of phones, the role of the DPP becomes more important. We visualize a sample of the trained alignment in Fig. 3.

Frequency Encodes Dispersion. Why does dispersion not always help? The models with fewer phones do not reap the benefits that the models with more phones do. The reason lies in the fact that the most common vowel formants are *already* dispersed. This indicates that we still have not quite modeled the mechanisms that select for good vowel formants, despite our work at the phonetic level; further research is needed. We would prefer a model that explains the *evolutionary motivation* of sound systems as communication systems.

Number of Induced Phones. What is most salient in the number of induced phones is that it is close to the number of IPA phonemes in the data. However, the performance of the phoneme-supervised system is much worse, indicating that, perhaps, while the linguists have the right idea about the *number* of universal symbols, they did not specify the correct IPA symbol in all cases. Our data analysis indicates that this is often due to pragmatic concerns in linguistic field analysis. For example, even if /ɪ/ is the proper IPA symbol for the sound, if there is no other sound in the vicinity the annotator may prefer to use more common /i/.

10 Related Work

Most closely related to our work is the classic study of Liljencrants and Lindblom (1972), who provide a simulation-based account of vowel systems. They argued that minima of a certain objective that encodes dispersion should correspond to canonical vowel systems of a given size n. Our tack is different in that we construct a generative probability model, whose parameters we learn from data. However, the essence of modeling is the same in that we explain *formant* values, rather than discrete IPA symbols. By extension, our work is also closely related to extensions of this theory (Schwartz et al., 1997; Roark, 2001) that focused on incorporating the notion of focalization into the experiments.

Our present paper can also be regarded as a continuation of Cotterell and Eisner (2017), in which we used DPPs to model vowel inventories as sets of discrete IPA symbols. That paper pretended that each IPA symbol had a single cross-linguistic (F_1, F_2) pair, an idealization that we remove in this paper by discarding the IPA symbols and modeling formant values directly.

11 Conclusion

Our model combines existing techniques of probabilistic modeling and inference to attempt to fit the actual distribution of the world's vowel systems. We presented a generative probability model of sets of measured (F_1, F_2) pairs. We view this as a necessary step in the development of generative probability models that can explain the distribution of the world's languages. Previous work on generating vowel inventories has focused on how those inventories were transcribed into IPA by field linguists, whereas we focus on the field linguists' acoustic measurements of how the vowels are actually pronounced.

Acknowledgments

We would like to acknowledge Tim Vieira, Katharina Kann, Sebastian Mielke and Chu-Cheng Lin for reading many early drafts. The first author would like to acknowledge an NDSEG grant and a Facebook PhD fellowship. This material is also based upon work supported by the National Science Foundation under Grant No. 1718846 to the last author.

References

Raja Hafiz Affandi, Emily Fox, and Ben Taskar. 2013. Approximate inference in continuous determinantal processes. In *Advances in Neural Information Processing Systems*, pages 1430–1438.

Roy Becker-Kristal. 2006. Predicting vowel inventories: The dispersion-focalization theory revisited. *The Journal of the Acoustical Society of America*, 120(5):3248–3248.

Roy Becker-Kristal. 2010. *Acoustic Typology of Vowel Inventories and Dispersion Theory: Insights from a Large Cross-Linguistic Corpus*. Ph.D. thesis, UCLA.

Paulus Petrus Gerardus Boersma et al. 2002. Praat, a system for doing phonetics by computer. *Glot International*, 5.

Alexei Borodin and Eric M. Rains. 2005. Eynard-Mehta theorem, Schur process, and their Pfaffian analogs. *Journal of Statistical Physics*, 121(3-4):291–317.

Bernard Comrie, Matthew S. Dryer, David Gil, and Martin Haspelmath. 2013. Introduction. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Ryan Cotterell and Jason Eisner. 2017. Probabilistic typology: Deep generative models of vowel inventories. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, Vancouver, Canada.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Rtatistical Society, Series B (Statistical Methodology)*, pages 1–38.

Li Deng and Douglas O'Shaughnessy. 2003. *Speech Processing: A Dynamic and Optimization-Oriented Approach*. CRC Press.

Stuart Geman and Donald Geman. 1984. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):721–741.

Matthew K. Gordon. 2016. *Phonological Typology*. Oxford.

Peter J. Green. 1995. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711–732.

Tony Jebara, Risi Kondor, and Andrew Howard. 2004. Probability product kernels. *Journal of Machine Learning Research*, 5:819–844.

Peter Ladefoged. 2001. *Vowels and Consonants: An Introduction to the Sounds of Languages*. Wiley-Blackwell.

Richard A. Levine and George Casella. 2001. Implementations of the Monte Carlo EM algorithm. *Journal of Computational and Graphical Statistics*, 10(3):422–439.

Johan Liljencrants and Björn Lindblom. 1972. Numerical simulation of vowel quality systems: The role of perceptual contrast. *Language*, pages 839–862.

Brian Roark. 2001. Explaining vowel inventory tendencies via simulation: Finding a role for quantal locations and formant normalization. In *North East Linguistic Society*, volume 31, pages 419–434.

Jean-Luc Schwartz, Christian Abry, Louis-Jean Boë, Nathalie Vallée, and Lucie Ménard. 2005. The dispersion-focalization theory of sound systems. *The Journal of the Acoustical Society of America*, 117(4):2422–2422.

Jean-Luc Schwartz, Louis-Jean Boë, Nathalie Vallée, and Christian Abry. 1997. The dispersion-focalization theory of vowel systems. *Journal of Phonetics*, 25(3):255–286.

Henry Stark and John Woods. 2011. *Probability, Statistics, and Random Processes for Engineers*. Pearson.

Kenneth N. Stevens. 1987. Relational properties as perceptual correlates of phonetic features. In *International Conference of Phonetic Sciences*, pages 352–355.

Kenneth N. Stevens. 1989. On the quantal nature of speech. *Journal of Phonetics*, 17:3–45.

Wilson L. Taylor. 1953. Cloze procedure: a new tool for measuring readability. *Journalism and Mass Communication Quarterly*, 30(4):415.

Leslie G. Valiant. 1979. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201.

Maksims Volkovs and Richard S. Zemel. 2012. Efficient sampling for bipartite matching problems. In *Advances in Neural Information Processing Systems*, pages 1313–1321.

Hanna Wallach, Ian Murray, Ruslan Salakhutdinov, and David Mimno. 2009. Evaluation methods for topic models. In *International Conference on Machine Learning (ICML)*, pages 1105–1112.

Fortification of Neural Morphological Segmentation Models for Polysynthetic Minimal-Resource Languages

Katharina Kann*
Center for Information and Language Processing
LMU Munich, Germany
kann@cis.lmu.de

Manuel Mager*
Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas
Universidad Nacional Autónoma de México
mmager@turing.iimas.unam.mx

Ivan Meza-Ruiz
Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas
Universidad Nacional Autónoma de México
ivanvladimir@turing.iimas.unam.mx

Hinrich Schütze
Center for Information and Language Processing
LMU Munich, Germany
inquiries@cislmu.org

Abstract

Morphological segmentation for polysynthetic languages is challenging, because a word may consist of many individual morphemes and training data can be extremely scarce. Since neural sequence-to-sequence (seq2seq) models define the state of the art for morphological segmentation in high-resource settings and for (mostly) European languages, we first show that they also obtain competitive performance for Mexican polysynthetic languages in minimal-resource settings. We then propose two novel multi-task training approaches—one with, one without need for external unlabeled resources—, and two corresponding data augmentation methods, improving over the neural baseline for all languages. Finally, we explore cross-lingual transfer as a third way to fortify our neural model and show that we can train one single multi-lingual model for related languages while maintaining comparable or even improved performance, thus reducing the amount of parameters by close to 75%. We provide our morphological segmentation datasets for Mexicanero, Nahuatl, Wixarika and Yorem Nokki for future research.

1 Introduction

Due to the advent of computing technologies to indigenous communities all over the world, natural language processing (NLP) applications for languages with limited computer-readable textual data are getting increasingly important. This contrasts with current research, which focuses strongly on approaches which require large amounts of training data, e.g., deep neural networks. Those are not trivially applicable to *minimal-resource* settings with less than $1,000$ available training examples. We aim at closing this gap for morphological surface segmentation, the task of splitting a word into the surface forms of its smallest meaning-bearing units, its *morphemes*.

Recovering morphemes provides information about unknown words and is thus especially important for polysynthetic languages with a high morpheme-to-word ratio and a consequently large overall number of words. To illustrate how segmentation helps understanding unknown multiple-morpheme words, consider an example in this paper's language of writing: even if the word *unconditionally* did not appear in a given training corpus, its meaning could still be derived from a combination of its morphs *un*, *condition*, *al* and *ly*.

Due to its importance for down-stream tasks (Creutz et al., 2007; Dyer et al., 2008), segmentation has been tackled in many different ways, considering unsupervised (Creutz and Lagus, 2002), supervised (Ruokolainen et al., 2013) and semi-supervised settings (Ruokolainen et al., 2014). Here, we add three new questions to this line of research: (i) Are data-hungry neural network models

*The first two authors contributed equally.

Proceedings of NAACL-HLT 2018, pages 47–57
New Orleans, Louisiana, June 1 - 6, 2018.

applicable to segmentation of polysynthetic languages in minimal-resource settings? (ii) How can the performance of neural networks for surface segmentation be improved if we have only unlabeled or no external data at hand? (iii) Is cross-lingual transfer for this task possible between related languages? The last two questions are crucial: While for many languages it is difficult to obtain the number of annotated examples used in earlier work on (semi-)supervised methods, a limited amount might still be obtainable.

We experiment on four polysynthetic Mexican languages: Mexicanero, Nahuatl, Wixarika and Yorem Nokki (details in §2). The datasets we use are, as far as we know, the first computer-readable datasets annotated for morphological segmentation in those languages.

Our experiments show that neural seq2seq models perform on par with or better than other strong baselines for our polysynthetic languages in a minimal-resource setting. However, we further propose two novel multi-task approaches and two new data augmentation methods. Combining them with our neural model yields up to 5.05% absolute accuracy or 3.40% F1 improvements over our strongest baseline.

Finally, following earlier work on cross-lingual knowledge transfer for seq2seq tasks (Johnson et al., 2017; Kann et al., 2017), we investigate training one single model for all languages, while sharing parameters. The resulting model performs comparably to or better than the individual models, but requires only roughly as many parameters as one single model.

Contributions. To sum up, we make the following contributions: (i) we confirm the applicability of neural seq2seq models to morphological segmentation of polysynthetic languages in minimal-resource settings; (ii) we propose two novel multi-task training approaches and two novel data augmentation methods for neural segmentation models; (iii) we investigate the effectiveness of cross-lingual transfer between related languages; and (iv) we provide morphological segmentation datasets for Mexicanero, Nahuatl, Wixarika and Yorem Nokki.

2 Polysynthetic Languages

Polysynthetic languages are morphologically rich languages which are highly synthetic, i.e., single words can be composed of many individual morphemes. In extreme cases, entire sentences consist of only one single token, whereupon "every argument of a predicate must be expressed by morphology on the word that contains that assigner" (Baker, 2006). This property makes surface segmentation of polysynthetic languages at the same time complex and particularly relevant for further linguistic analysis.

Mexicanero		Nahuatl		Wixarika		Yorem N.	
frq.	m.	frq.	m.	frq.	m.	frq.	m.
136	ni	155	o	327	p+	102	k
128	ki	99	ni	230	ne	88	m
114	ti	84	ti	173	p	87	ne
105	u	81	k	169	ti	83	ka
70	s	61	tl	167	ka	79	ta
44	mo	59	mo	98	u	54	po
42	ka	55	s	97	ta	50	e'
39	a	52	ki	95	a	36	ye
31	nich	48	i	92	pe	36	su
31	$i	43	tla	91	e	36	ri
24	ta	39	'ke	80	r	34	a
24	l	34	nech	74	wa	31	me
22	tahtanili	31	no	69	me	30	wa
21	no	27	ya	68	ni	30	re
17	ya	27	tli	68	ke	27	na
17	t	24	x	66	eu	24	wi
17	ke	23	tlanilia	58	ye	24	a
17	ita	23	e	57	ri	23	te
16	piya	21	tika	52	tsi	20	si
15	an	21	n	52	te	16	'wi

Table 1: The most frequent morphs (*m.*) together with their frequencies (*frq.*) in our datasets.

In this paper, we experiment on four polysynthetic languages of the Yuto-Aztecan family (Baker, 1997), with the goal of improving the performance of neural seq2seq models. The languages will be described in the rest of this section.

Mexicanero is a Western Peripheral Nahuatl variant, spoken in the Mexican state of Durango by approximately one thousand people. This dialect is isolated from the rest of the other branches and has a strong process of Spanish stem incorporation, while also having borrowed some suffixes from that language (Vanhove et al., 2012). It is common to see Spanish words mixed with Nahuatl agglutinations. In the following example we can see an intrasentencial mixing of Spanish (*in uppercases*) and Mexicanero:

u|ni|ye MALO – *I was sick*

Nahuatl is a large subgroup of the Yuto-Aztecan language family, and, including all of its variants, the most spoken native language in Mexico. Its almost two million native speakers live mainly in Puebla, Guerrero, Hidalgo, Veracruz, and San Luis Potosi, but also in Oaxaca, Durango, Modelos, Mexico City, Tlaxcala, Michoacan, Nayarit and the State of Mexico. Three dialectical groups are known: Central Nahuatl, Occidental Nahuatl and Oriental Nahuatl. The data collected for this work belongs to the Oriental branch spoken by 70 thousand people in Northern Puebla.

Like all languages of the Yuto-Aztecan family, Nahuatl is agglutinative and one word can consist of a combination of many different morphemes. Usually, the verb functions as the stem and gets extended by morphemes specifying, e.g., subject, patient, object or indirect object. The most common syntax sequence for Nahuatl is SOV. An example word is:

o|ne|mo|kokowa|ya – *I was sick*

Wixarika is a language spoken in the states of Jalisco, Nayarit, Durango and Zacatecas in Central West Mexico by approximately fifty thousand people. It belongs to the Coracholan group of languages within the Yuto-Aztecan family. Wixarika has five vowels {a,e,i,+[1],u} with long and short variants. An example for a word in the language is:

ne|p+|ti|kuye|kai – *I was sick*

Like Nahuatl, it has an SOV syntax, with heavy agglutination on the verb. Wixarika is morphologically more complex than other languages from the same family, because it incorporates more information into the verb (Leza and López, 2006). This leads to a higher number of morphemes per word as can also be seen in Table 3.

Yorem Nokki is part of Taracachita subgroup of the Yuto-Aztecan language family. Its Southern dialect is spoken by close to forty thousand people in the Mexican states of Sinaloa and Sonora, while its Northern dialect has about twenty thousand speakers. In this work, we consider the Southern dialect. The nominal morphology of Yorem

[1]While linguists often use a dashed i (ɨ) to denote this vowel, in practice almost all native speakers use a plus symbol (+). In this work, we choose to use the latter.

	Mexicanero	Nahuatl	Wixarika	Yorem N.
train	427	540	665	511
dev	106	134	176	127
test	355	449	553	425
total	888	1123	1394	1063

Table 2: Number of examples in the final data splits for all languages.

Nokki is rather simple, but, like in the other Yuto-Aztecan languages, the verb is highly complex. Its alphabet consists of 28 characters and contains 8 different vowels. An example verb is:

ko'kore|ye|ne – *I was sick*

3 Morphological Segmentation Datasets

To create our datasets, we make use of both segmentable (i.e., consisting of multiple morphemes) and non-segmentable (i.e., consisting of one single morpheme) words described in books of the collection *Archive of Indigenous Languages* in Mexicanero (Canger, 2001), Nahuatl (Lastra de Suárez, 1980), Wixarika (Gómez and López, 1999), and Yorem Nokki (Freeze, 1989). Statistics about the data in the four languages are displayed in Tables 1, 2 and 3. We include segmentable as well as non-segmentable words into our datasets in order to ensure that our methods can correctly decide against splitting up single morphemes. The phrases in all languages are mostly parallel, such that the corpora are roughly equivalent. Therefore, we can compare the morphology of translated words (cf. Table 3), noticing that the language with most agglutination is Wixarika, with an average rate of 3.25 morphemes per word; the other languages have an average of close to 2.2 morphemes per word. This higher morphological complexity naturally produces data sparsity at the token level. Also, we can notice that Wixarika has more unique words than the rest of our studied languages. However, Nahuatl has with 810 the highest number of unique morphemes.

Final splits. In order to make follow-up work on minimal-resource settings for morphological segmentation easily comparable, we provide predefined splits of our datasets[2]. 40% of the data constitute the test sets. Of the remaining data, we

[2]Our datasets can be found together with the code of our models at `http://turing.iimas.unam.mx/wix/MexSeg`.

	Mex.	Nahuatl	Wixarika	Yorem N.
Words	888	1123	1385	1063
SegWords	539	746	1131	774
Morphs	1889	2467	4502	2266
UniMorphs	602	810	653	662
Seg/W	0.606	0.664	0.816	0.728
Morphs/W	2.127	2.196	3.250	2.131
MaxMorphs	7	6	10	10

Table 3: Number of words, segmentable words (SegWords), total morphs (Morphs), and unique morphs (UniMorphs) in our datasets. Seg/W: proportion of words consisting or more than one morpheme; Morphs/W: morphemes per word; MaxMorphs: maximum number of morphemes found in one word.

use 20% for development and the rest for training. The final numbers of words per dataset and language are shown in Table 2.

4 Neural Seq2seq Models for Segmentation

In the beginning of this section, we will introduce our neural architecture for segmentation. Subsequently, we will first describe our two proposed multi-task training approaches and second our data augmentation methods. Finally, we will elaborate on expected differences between the two.

4.1 Character-Based Encoder-Decoder RNN

Following work on segmentation by Kann et al. (2016) for high-resource settings, our approach is based on the neural seq2seq model introduced by Bahdanau et al. (2015) for machine translation.

Encoder. The first part of our model is a bidirectional recurrent neural network (RNN) which encodes the input sequence, i.e., the sequence of characters of a given word $w = w_1, w_2, \ldots, w_{T_v}$, represented by the corresponding embedding vectors $v_{w_1}, ..., v_{w_{T_v}}$. In particular, our encoder consists of one gated recurrent neural network (GRU) which processes the input in forward direction and a second GRU which processes the input from the opposite side.

Encoding with this bidirectional GRU yields the forward hidden state $\overrightarrow{h}_i = f\left(\overrightarrow{h}_{i-1}, v_i\right)$ and the backward hidden state $\overleftarrow{h}_i = f\left(\overleftarrow{h}_{i+1}, v_i\right)$, for a non-linear activation function f. Their concatenation $h_i = \left[\overrightarrow{h_i}; \overleftarrow{h_i}\right]$ is passed on to the decoder.

Decoder. The second part of our network, the decoder, is a single GRU, defining a probability distribution over strings in $(\Sigma \cup \mathtt{S})^*$, for an alphabet Σ and a separation symbol $\mathtt{S}$:

$$p_{\text{ED}}(c \mid w) = \prod_{t=1}^{T_c} p(c_t \mid c_1, \ldots, c_{t-1}, w). \quad (1)$$

where $p(c_t \mid c_1, \ldots, c_{t-1}, w)$ is computed using an attention mechanism and an output softmax layer over $\Sigma \cup \mathtt{S}$.

A more detailed description of the general attention-based encoder-decoder architecture can be found in the original paper by Bahdanau et al. (2015).

5 Improving Neural Models for Segmentation

5.1 Multi-Task Training

In order to leverage unlabeled data or even random strings during training, we define an autoencoding auxiliary task, which consists of encoding the input and decoding an output which is identical to the original string.

Then, our multi-task training objective is to maximize the joint log-likelihood of this auxiliary task and our segmentation main task:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{(w,c)\in\mathcal{T}} \log p_{\boldsymbol{\theta}}\left(c \mid e(w)\right) + \sum_{a\in\mathcal{A}} \log p_{\boldsymbol{\theta}}(a \mid e(a)) \quad (2)$$

$\mathcal{T}$ denotes the segmentation training data with examples consisting of a word w and its segmentation c. $\mathcal{A}$ denotes either a set of words in the language of the system or a set of random strings. The function e describes the encoder and depends on the model parameters $\boldsymbol{\theta}$, which are shared across the two tasks. For training, we use data from both sets at the same time and mark each example with an additional, task-specific input symbol.

We treat the size of $\mathcal{A}$ as a hyperparameter which we optimize on the development set separately for each language. Values we experiment with are m times the amount of instances in the original training set, with $m \in \{1, 2, 4, 8\}$.[3]

[3] An exception is Yorem Nokki, for which we do not have enough unlabeled data available, such that we experiment only with $m \in \{1, 2\}$.

There are multiple reasons why we expect multi-task training to improve the performance of the final model. First, multi-task training should act as a regularizer. Second, for our models, the segmentation task consists in large parts of learning to copy the input character sequence to the output. This, however, can be learned from any string and does not require annotated segmentation boundaries. Third, in the case of unlabeled data (i.e., not for random strings), we expect the character language model in the decoder to improve, since it is trained on additional data.

We denote models trained with multi-task training using unlabeled corpus data as **MTT-U** and models trained with multi-task training using random strings as **MTT-R**.

5.2 Data Augmentation

A second option to make use of unlabeled data or random strings is to extend the available training data with new examples made from those. The main question to answer here is how to include the new data into the existing datasets. We do this by building new training examples in a fashion similar to the multi-task setup. All newly created instances are of the form

$$w \mapsto w \tag{3}$$

where either $w \in V$ with V being the observed vocabulary of the language, e.g., words in a given unlabeled corpus, or $w \in R$ with R being a set of sequences of random characters from the alphabet Σ of the language.

Again, we treat the amount of additional training examples as a hyperparameter which we optimize on the development set separately for each language. We explore m times the amount of instances in the original training set, with $m \in \{1, 2, 4, 8\}$.

The reasons why we expect our data augmentation methods to lead to better segmentation models are similar to those for multi-task training.

We call models trained on datasets augmented with unlabeled corpus data or random strings **DA-U** or **DA-R**, respectively.

5.3 Differences Between Multi-task Training and Data Augmentation

The difference between MTT-U (resp. MTT-R) and DA-U (resp. MTT-U) is a single element in the input sequence (the one representing the task). However, this information enables the model to handle each given instance correctly at inference time. As a result, it gets more robust against noisy data, which seems crucial for our way of using unlabeled corpora. Consider, for example, the Nahuatl word $onemokokowaya$. Training on

$$onemokokowaya \mapsto onemokokowaya$$

will make the model learn *not* to segment words which consist of the morphemes $o, ne, mo, kokowa, ya$, which should ultimately hurt performance. The multi-task approach, in contrast, mitigates this problem.

As a conclusion, we expect the data augmentation approach with *unlabeled data* to not obtain outstanding performance, but rather consider it an important and informative baseline for the corresponding multi-task approach. Using *random strings*, the difference between the multi-task and the data augmentation approaches is less obvious: Real morphemes should appear rarely enough in the created random character sequences to avoid the negative effect which we expect for corpus words. We thus assume that the performances of MTT-R and DA-R should be similar.

6 Experiments

6.1 Data

We apply our models to the datasets described in §3. For the multi-task training and data augmentation using unlabeled data, we use (unsegmented) words from a parallel corpus collected by Gutierrez-Vasques et al. (2016) for Nahuatl and the closely related Mexicanero. For Wixarika we use data from Mager et al. (2018) and for Yorem Nokki we use text from Maldonado Martínez et al. (2010).

6.2 Baselines

Now, we will describe the baselines we use to evaluate the overall performance of our approaches.

Supervised seq2seq RNN (S2S). As a first baseline, we employ a fully supervised neural model without data augmentation or multi-task training, i.e., an attention-based encoder-decoder RNN (Bahdanau et al., 2015) which has been trained only on the available annotated data.

Semi-supervised MORFESSOR (MORF). We further compare to the semi-supervised version

of MORFESSOR (Kohonen et al., 2010), a well-known morphological segmentation system. During training, we tune the hyperparameters for each language on the respective development set. The best performing model is applied to the test set.

FlatCat (FC). Our next baseline is FlatCat (Grönroos et al., 2014), a variant of MORFESSOR. It consists of a hidden Markov model for segmentation. The states of the model correspond either to a word boundary and one of the four morph categories stem, prefix, suffix, and non-morpheme. It can work in an unsupervised way, but, similar to the previous baseline, can make effective use of small amounts of labeled data.

CRF. We further compare to a conditional random fields (CRF) (Lafferty et al., 2001) model, in particular a strong discriminative model for segmentation by Ruokolainen et al. (2014). It reduces the task to a classification problem with four classes: beginning of a morph, middle of a morph, end of a morph and single character morph. Training is again semi-supervised and the model was previously reported to obtain good results for small amounts of unlabeled data (Ruokolainen et al., 2014), which makes it very suitable for our minimal-resource setting.

6.3 Hyperparameters

Neural network parameters. All GRUs in both the encoder and the decoder have 100-dimensional hidden states. All embeddings are 300-dimensional.

For training, we use ADADELTA (Zeiler, 2012) with a minibatch size of 20. We initialize all weights to the identity matrix and biases to zero (Le et al., 2015). All models are trained for a maximum of 200 epochs, but we evaluate after every 5 epochs and apply the best performing model at test time. Our final reported results are averaged accuracies over 5 single training runs.

Optimizing the amount of auxiliary task data. The performance of our neural segmentation model in dependence of the amount of auxiliary task training data can be seen in Figure 1. As a general tendency across all languages, adding more data seems better, particularly for the autoencoding task with random strings. The only exception is Wixarika.

The final configurations we choose for m (cf. §5.1) in the case of multi-task training with the auxiliary task of autoencoding corpus data are $m = 4$ for Mexicanero, Nahuatl and Wixarika and $m = 1$ for Yorem Nokki. For multi-task training with autoencoding of random strings we select $m = 8$ for Mexicanero, Nahuatl and Yorem Nokki and $m = 4$ for Wixarika.

Optimizing the amount of artificial training data for data augmentation. Figure 2 shows the performance of the encoder-decoder depending on the amount of added artificial training data. In the case of random strings, again, adding more training data seems to help more. However, using corpus data seems to hurt performance and the more such examples we use, the worse accuracy we obtain. Thus, we conclude that (as expected) data augmentation with corpus data is not a good way to improve the model's performance. We will discuss this in more detail in §6.5.

Even though the final conclusion should be to not add much corpus data, we apply what gives best results on the development set. The final configurations we thus choose for DA-U are $m = 1$ for Mexicanero, Wixarika and Yorem Nokki and $m = 2$ for Nahuatl. For DA-R, we select $m = 4$ for Mexicanero, Wixarika and Yorem Nokki and $m = 8$ for Nahuatl.

6.4 Evaluation Metrics

Accuracy. First, we evaluate using accuracy on the token level. Thus, an example counts as correct if and only if the output of the system matches the reference solution exactly, i.e., if all output symbols are predicted correctly.

F1. Our second evaluation metric is border F1, which measures how many segment boundaries are predicted correctly by the model. While we use this metric because it is common for segmentation tasks, it is not ideal for our models since those are not guaranteed to preserve the input character sequence. We handle this problem as follows: In order to compare borders, we identify them by the position of their preceding letter, i.e., if in both the model's guess and the gold solution a segment border appears after the second character, it counts as correct. Wrong characters are ignored. Note that this comes with the disadvantage of erroneously inserted characters leading to all subsequent segment borders being counted as incorrect.

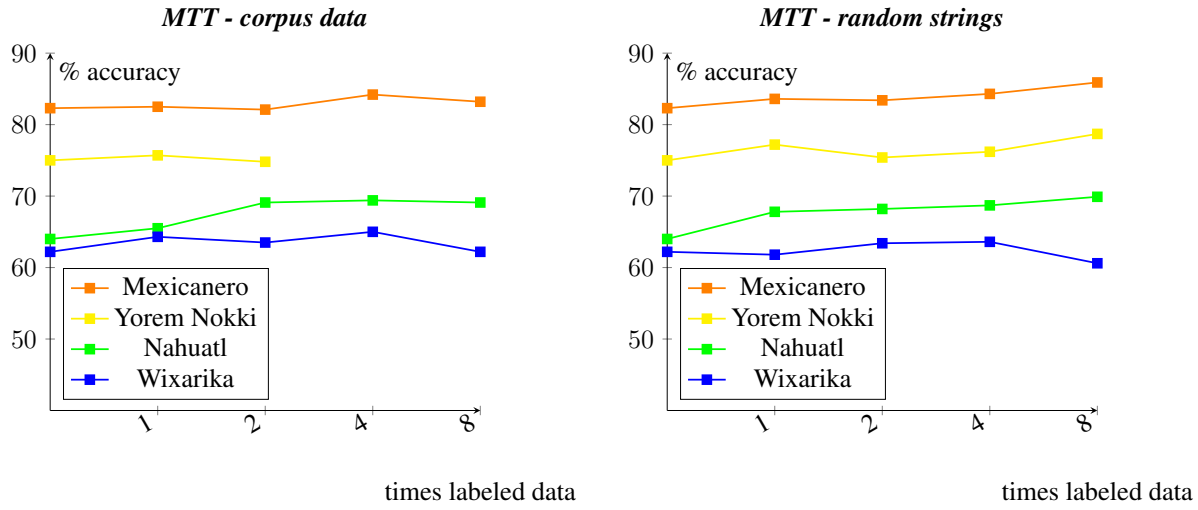


Figure 1: Accuracy on the development set in dependence of the amount of auxiliary task training data for multi-task learning.

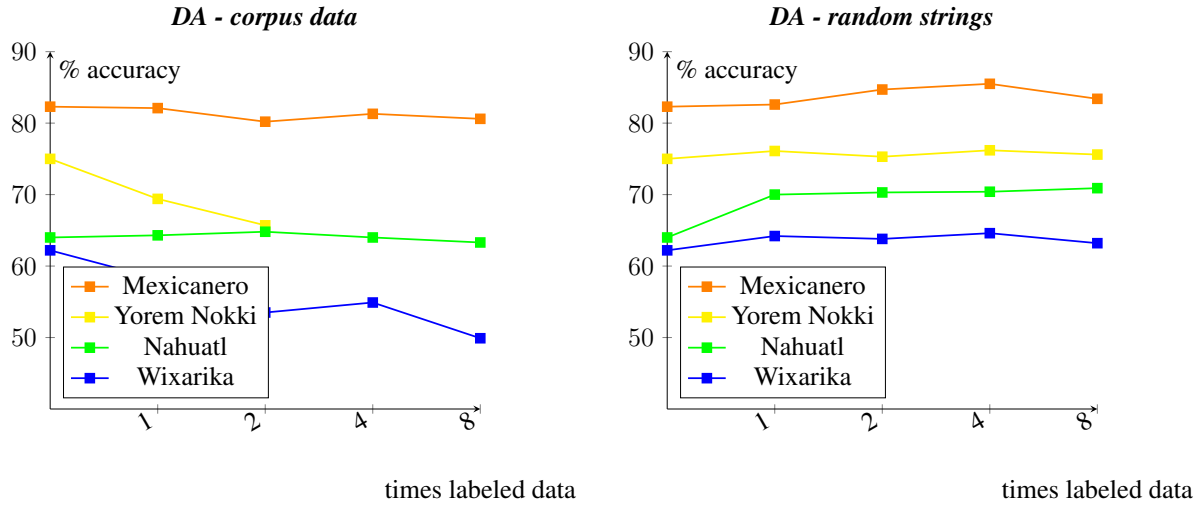


Figure 2: Accuracy on the development set in dependence of the amount of additional training data.

6.5 Test Results and Discussion

Table 4 shows that accuracy and F1 seem to be highly correlated for our task. The test results also give an answer to our first research question: The neural model S2S performs on par with CRF, the strongest baseline, for all languages but Nahuatl. Further, S2S and CRF both outperform MORF and FC by a wide margin. We may thus conclude that neural models are indeed applicable to segmentation of polysynthetic languages in a low-resource setting.

Second, we can see that all our proposed methods except for DA-U improve over S2S, the neural baseline: The accuracy of MTT-U is between 0.0141 (Wixarika) and 0.0547 (Mexicanero) higher than S2S's. MTT-R improves between 0.0380 (Wixarika) and 0.0532 (Yorem Nokki). Finally, DA-R outperforms S2S by 0.0367 to 0.0479 accuracy for Yorem Nokki and Mexicanero, respectively. The overall picture when considering F1 looks similar. Comparing our approaches to each other, there is no clear winner. This might be due to differences in the unlabeled data we use: the corpus we use for Mexicanero and Nahuatl is from dialects different from both respective test sets. Assuming that the effect of training a language model using unlabeled data and erroneously learning to not segment words are working against each other for MTT-U, this might explain why MTT-U is best for Mexicanero and the gap between MTT-U and MTT-R is smaller for Nahuatl than for Yorem Nokki and Wixarika.

As mentioned before (cf. §5.3), a simple data augmentation method using unlabeled data should

	Accuracy								F1							
	MTT-U	MTT-R	DA-U	DA-R	S2S	MORF	CRF	FC	MTT-U	MTT-R	DA-U	DA-R	S2S	MORF	CRF	FC
Mex.	**.8051**	.7955	.7611	.7983	.7504	.3364	.7837	.5420	**.8786**	.8694	.6715	.8683	.8618	.5121	.8639	.5621
Nahuatl	.6004	.6027	.5541	.6018	.5585	.4044	**.6444**	.4888	.7388	.7367	.6865	.7328	.7266	.4154	**.7487**	.5185
Wixarika	.5895	.6134	.5425	**.6188**	.5754	.3989	.5866	.4523	.7949	.8024	.7109	**.8161**	.7961	.4426	.7932	.5568
Yorem N.	.6856	**.7101**	.6212	.6936	.6569	.4812	.6596	.5781	.7887	**.8076**	.7133	.7923	.7730	.3528	.7736	.6139

Table 4: Performances of our multi-task and data augmentation approaches compared to all baselines described in the text. The reported results for neural models are averages over 5 training runs. Best results per language and metric are in bold.

hurt performance. This is indeed the result of our experiments: DA-U performs worse than S2S for all languages except for Mexicanero, where the unlabeled corpus is from another language: the closely related Nahuatl. We thus conclude that multi-task training (instead of simple data augmentation) is crucial for the use of unlabeled data.

Finally, our methods compare favorably to all baselines, with the exception of CRF for Nahuatl. While CRF is overall the strongest baseline for our considered languages, our methods outperform it by up to 0.0214 accuracy or 0.0147 F1 for Mexicanero, 0.0322 accuracy or 0.0229 F1 for Wixarika and 0.0505 accuracy or 0.0340 F1 for Yorem Nokki. This shows the effectiveness of our fortified neural models for minimal-resource morphological segmentation.

7 Cross-Lingual Transfer Learning

We now want to investigate the performance of *one single model* trained on all languages at once. This is done in analogy to the multi-task training described in §5.1. We treat segmentation in each language as a separate task and train an attention-based encoder-decoder model on maximizing the joint log-likelihood:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{L_i \in L} \sum_{(w,c) \in \mathcal{T}_{L_i}} \log p_{\boldsymbol{\theta}}\left(c \mid e(w)\right) \tag{4}$$

$\mathcal{T}_{L_i}$ denotes the segmentation training data in language L_i and L is the set of our languages. As before, each training example consists of a word w and its segmentation c.

7.1 Experimental Setup

We keep all model parameters and the training regime as described in §6.3. However, our training data now consists of a combination of all available training data for all 4 languages. In order to enable the model to differentiate between the tasks, we prepend one language-specific input symbol to each instance. This corresponds to having one embedding in the input which marks the task. An example training instance for Yorem Nokki is

$$\textit{L=YN}\ ko'koreyene \mapsto ko'kore|ye|ne,$$

where *L=YN* indicates the language.

Due to the previous high correlation between accuracy and F1 we only use accuracy on the word level as the evaluation metric for this experiment.

	M-Lang	S-Lang	BestMTT	BestDA
Mex.	.6858	.7504	**.8051**	.7983
Nahuatl	.5955	.5585	**.6027**	.6018
Wixarika	.6021	.5754	.6134	**.6188**
Yorem N.	.6223	.6569	**.7101**	.6936

Table 5: Accuracies of our model trained on all languages (M-Lang) and the models trained on single languages (S-Lang). The highest multi-task and data augmentation accuracies are repeated for an easy comparison.

7.2 Results and Discussion

In Table 5, we show the results of the multi-lingual model, which was trained on all languages, compared to all individual models, as well as each respective best multi-task approach and data augmentation method. The results differ among languages: Most remarkably, for both Wixarika and Nahuatl, the accuracy of the multi-lingual model is higher than the one of the single-language model. This might be related to them being the languages with most training data available (cf. Table 3).

Note, however, that even for the remaining two languages—Mexicanero and Yorem Nokki—we hardly lose accuracy when comparing the multi-lingual to the individual models. Since we only use one model (instead of four), without increasing its size significantly, we thus reduce the amount of parameters by nearly 75%.

8 Related Work

Work on morphological segmentation was started more than 6 decades ago (Harris, 1951). Since then, many approaches have been developed: In the realm of unsupervised methods, two important systems are LINGUISTICS (Goldsmith, 2001) and MORFESSOR (Creutz and Lagus, 2002). The latter was later extended to a semi-supervised version (Kohonen et al., 2010) in order to make use of the abundance of unlabeled data which is available for many languages.

Ruokolainen et al. (2013) focused explicitly on low-resource scenarios and applied CRFs to morphological segmentation in several languages. They reported better results than earlier work, including semi-supervised approaches. In the following year, they extended their approach to be able to use unlabeled data as well, further improving performance (Ruokolainen et al., 2014).

Cotterell et al. (2015) trained a semi-Markov CRF (semi-CRF) (Sarawagi and Cohen, 2005) jointly on morphological segmentation, stemming and tagging. For the similar problem of Chinese word segmentation, Zhang and Clark (2008) trained a model jointly on part-of-speech tagging. However, we are not aware of any prior work on multi-task training or data augmentation for neural segmentation models.

In fact, the two only neural seq2seq approaches for morphological segmentation we know of focused on *canonical segmentation* (Cotterell et al., 2016) which differs from the *surface segmentation* task considered here in that it restores changes to the surface form of morphemes which occurred during word formation. Kann et al. (2016) also used an encoder-decoder RNN and combined it with a neural reranker. While our model architecture was inspired by them, their model was purely supervised. Additionally, they did not investigate the applicability of their neural seq2seq model in low-resource settings or for polysynthetic languages. Ruzsics and Samardzic (2017) extended the standard encoder-decoder architecture for canonical segmentation to contain a language model over segments and improved results. However, a big difference to our work is that they still used more than ten times as much training data as we have available for the indigenous Mexican languages we are working on here.

Another neural approach—this time for surface segmentation—was presented by Wang et al. (2016). The authors, instead of using seq2seq models, treat the task as a sequence labeling problem and use LSTMs to classify every character either as the beginning, middle or end of a morpheme, or as a single-character morpheme.

Cross-lingual knowledge transfer via language tags was proposed for neural seq2seq models before, both for tasks that handle sequences of words (Johnson et al., 2017) and tasks that work on sequences of characters (Kann et al., 2017). However, to the best of our knowledge, we are the first to try such an approach for a morphological segmentation task. In many other areas of NLP, cross-lingual transfer has been applied successfully, e.g., in entity recognition (Wang and Manning, 2014), language modeling (Tsvetkov et al., 2016), or parsing (Cohen et al., 2011; Søgaard, 2011; Ammar et al., 2016).

9 Conclusion and Future Work

We first investigated the applicability of neural seq2seq models to morphological surface segmentation for polysynthetic languages in minimal-resource settings, i.e., for considerably less than $1,000$ training instances. Although they are generally thought to require large amounts of training data, neural networks obtained an accuracy comparable to or higher than several strong baselines.

Subsequently, we proposed two novel multi-task training approaches and two novel data augmentation methods to further increase the performance of our neural models. Adding those, we improved over the neural baseline for all languages, and for Mexicanero, Wixarika and Yorem Nokki our final models outperformed all baselines by up to 5.05% absolute accuracy or 3.40% F1. Furthermore, we explored cross-lingual transfer between our languages and reduced the amount of necessary model parameters by about 75%, while improving performance for some of the languages.

We publically release our datasets for morphological surface segmentation of the polysynthetic minimal-resource languages Mexicanero, Nahuatl, Wixarika and Norem Yokki.

Acknowledgments

We would like to thank Paulina Grnarova, Rodrigo Nogueira and Ximena Gutierrez-Vasques for their helpful feedback.

References

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

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

Mark C Baker. 1997. Complex predicates and agreement in polysynthetic languages. *Complex predicates,* pages 247–288.

Mark C Baker. 2006. On zero agreement and polysynthesis. In *Arguments and Agreement*, pages 289–320.

Una Canger. 2001. *Mexicanero de la sierra madre occidental*. El Colegio de México.

Shay B Cohen, Dipanjan Das, and Noah A Smith. 2011. Unsupervised structure prediction with non-parallel multilingual guidance. In *EMNLP*.

Ryan Cotterell, Thomas Müller, Alexander Fraser, and Hinrich Schütze. 2015. Labeled morphological segmentation with semi-markov models. In *CoNLL*.

Ryan Cotterell, Tim Vieira, and Hinrich Schütze. 2016. A joint model of orthography and morphological segmentation. In *NAACL-HLT*.

Mathias Creutz, Teemu Hirsimäki, Mikko Kurimo, Antti Puurula, Janne Pylkkönen, Vesa Siivola, Matti Varjokallio, Ebru Arisoy, Murat Saraçlar, and Andreas Stolcke. 2007. Morph-based speech recognition and modeling of out-of-vocabulary words across languages. *TSLP* 5(1):3:1–3:29.

Mathias Creutz and Krista Lagus. 2002. Unsupervised discovery of morphemes. In *Workshop on Morphological and Phonological Learning*.

Christopher Dyer, Smaranda Muresan, and Philip Resnik. 2008. Generalizing word lattice translation. In *ACL-HLT*.

Ray A Freeze. 1989. *Mayo de Los Capomos, Sinaloa*. El Colegio de México.

John Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics* 27(2):153–198.

Paula Gómez and Paula Gómez López. 1999. *Huichol de San Andrés Cohamiata, Jalisco*. El Colegio de México.

Stig-Arne Grönroos, Sami Virpioja, Peter Smit, and Mikko Kurimo. 2014. Morfessor FlatCat: An HMM-based method for unsupervised and semi-supervised learning of morphology. In *COLING*.

Ximena Gutierrez-Vasques, Gerardo Sierra, and Isaac Hernandez Pompa. 2016. Axolotl: A web accessible parallel corpus for Spanish-Nahuatl. In *LREC*.

Zellig Sabbettai Harris. 1951. *Methods in structural linguistics*. Chicago University Press.

Melvin Johnson, Mike Schuster, Quoc Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *TACL* 5:339–351.

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2016. Neural morphological analysis: Encoding-decoding canonical segments. In *EMNLP*.

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2017. One-shot neural cross-lingual transfer for paradigm completion. In *ACL*.

Oskar Kohonen, Sami Virpioja, and Krista Lagus. 2010. Semi-supervised learning of concatenative morphology. In *SIGMORPHON*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Yolanda Lastra de Suárez. 1980. *Náhuatl de Acaxochitlán (Hidalgo)*. Colegio de México.

José Luis Iturrioz Leza and Paula Gómez López. 2006. *Gramática wixarika*. Lincom Europa.

Manuel Mager, Carrillo Dionico, and Meza Ivan. 2018. Probabilistic finite-state morphological segmenter for the Wixarika (Huichol) language. *Journal of Intelligent & Fuzzy Systems* (Special Issue).

Juan Pedro Maldonado Martínez, Crescencio Buitimea Valenzuela, and Ángel Macochini Alonzo. 2010. *Vivimos en un pueblo yaqui*. SEP.

Teemu Ruokolainen, Oskar Kohonen, Sami Virpioja, and Mikko Kurimo. 2013. Supervised morphological segmentation in a low-resource learning setting using conditional random fields. In *CoNLL*.

Teemu Ruokolainen, Oskar Kohonen, Sami Virpioja, and Mikko Kurimo. 2014. Painless semi-supervised morphological segmentation using conditional random fields. In *EACL*.

Tatyana Ruzsics and Tanja Samardzic. 2017. Neural sequence-to-sequence learning of internal word structure. In *CoNLL*.

Sunita Sarawagi and William W Cohen. 2005. Semi-Markov conditional random fields for information extraction. In *NIPS*.

Anders Søgaard. 2011. Data point selection for cross-language adaptation of dependency parsers. In *ACL-HLT*.

Yulia Tsvetkov, Sunayana Sitaram, Manaal Faruqui, Guillaume Lample, Patrick Littell, David Mortensen, Alan W Black, Lori Levin, and Chris Dyer. 2016. Polyglot neural language models: A case study in cross-lingual phonetic representation learning. In *NAACL-HLT*.

Martine Vanhove, Thomas Stolz, Aina Urdze, and Hitomi Otsuka. 2012. *Morphologies in contact*. Walter de Gruyter.

Linlin Wang, Zhu Cao, Yu Xia, and Gerard de Melo. 2016. Morphological segmentation with window LSTM neural networks. In *AAAI*.

Mengqiu Wang and Christopher D Manning. 2014. Cross-lingual pseudo-projected expectation regularization for weakly supervised learning. *TACL* 2:55–66.

Yue Zhang and Stephen Clark. 2008. Joint word segmentation and POS tagging using a single perceptron. In *ACL*.

Improving Character-based Decoding Using Target-Side Morphological Information for Neural Machine Translation

Peyman Passban, Qun Liu, Andy Way
ADAPT Centre
School of Computing
Dublin City University, Ireland
firstname.lastname@adaptcentre.ie

Abstract

Recently, neural machine translation (NMT) has emerged as a powerful alternative to conventional statistical approaches. However, its performance drops considerably in the presence of morphologically rich languages (MRLs). Neural engines usually fail to tackle the large vocabulary and high out-of-vocabulary (OOV) word rate of MRLs. Therefore, it is not suitable to exploit existing word-based models to translate this set of languages. In this paper, we propose an extension to the state-of-the-art model of Chung et al. (2016), which works at the character level and boosts the decoder with target-side morphological information. In our architecture, an additional morphology table is plugged into the model. Each time the decoder samples from a target vocabulary, the table sends auxiliary signals from the most relevant affixes in order to enrich the decoder's current state and constrain it to provide better predictions. We evaluated our model to translate English into German, Russian, and Turkish as three MRLs and observed significant improvements.

1 Introduction

Morphologically complex words (MCWs) are multi-layer structures which consist of different subunits, each of which carries semantic information and has a specific syntactic role. Table 1 gives a Turkish example to show this type of complexity. This example is a clear indication that word-based models are not suitable to process such complex languages. Accordingly, when translating MRLs, it might not be a good idea to treat words as atomic units as it demands a large vocabulary that imposes extra overhead. Since MCWs can appear in various forms we require a very large vocabulary to i) cover as many morphological forms and words as we can, and ii) reduce the number of OOVs. Neural models by their nature are complex, and we do not want to make them more complicated by working with large vocabularies. Furthermore, even if we have quite a large vocabulary set, clearly some words would remain uncovered by that. This means that a large vocabulary not only complicates the entire process, but also does not necessarily mitigate the OOV problem. For these reasons we propose an NMT engine which works at the character level.

Word	Translation
terbiye	good manners
terbiye.siz	rude
terbiye.siz.lik	rudeness
terbiye.siz.lik.leri	their rudeness
terbiye.siz.lik.leri.nden	from their rudeness

Table 1: Illustrating subword units in MCWs. The boldfaced part indicates the stem.

In this paper, we focus on translating into MRLs and issues associated with word formation on the target side. To provide a better translation we do not necessarily need a large target lexicon, as an MCW can be gradually formed during decoding by means of its subunits, similar to the solution proposed in character-based decoding models (Chung et al., 2016). Generating a complex word character-by-character is a better approach compared to word-level sampling, but it has other disadvantages.

One character can co-occur with another with almost no constraint, but a particular word or morpheme can only collocate with a very limited number of other constituents. Unlike words, characters are not meaning-bearing units and do not preserve syntactic information, so (in the extreme case) the

Proceedings of NAACL-HLT 2018, pages 58–68
New Orleans, Louisiana, June 1 - 6, 2018.

chance of sampling each character by the decoder is almost equal to the others, but this situation is less likely for words. The only constraint that prioritize which character should be sampled is information stored in the decoder, which we believe is insufficient to cope with all ambiguities. Furthermore, when everything is segmented into characters the target sentence with a limited number of words is changed to a very long sequence of characters, which clearly makes it harder for the decoder to remember such a long history. Accordingly, character-based information flows in the decoder may not be as informative as word- or morpheme-based information.

In the character-based NMT model everything is almost the same as its word-based counterpart except the target vocabulary whose size is considerably reduced from thousands of words to just hundreds of characters. If we consider the decoder as a classifier, it should in principle be able to perform much better over hundreds of classes (characters) rather than thousands (words), but the performance of character-based models is almost the same as or slightly better than their word-based versions. This underlines the fact that the character-based decoder is perhaps not fed with sufficient information to provide improved performance compared to word-based models.

Character-level decoding limits the search space by dramatically reducing the size of the target vocabulary, but at the same time widens the search space by working with characters whose sampling seems to be harder than words. The freedom in selection and sampling of characters can mislead the decoder, which prevents us from taking the maximum advantages of character-level decoding. If we can control the selection process with other constraints, we may obtain further benefit from restricting the vocabulary set, which is the main goal followed in this paper.

In order to address the aforementioned problems we redesign the neural decoder in three different scenarios. In the first scenario we equip the decoder with an additional morphology table including target-side affixes. We place an attention module on top of the table which is controlled by the decoder. At each step, as the decoder samples a character, it searches the table to find the most relevant information which can enrich its state. Signals sent from the table can be interpreted as additional constraints. In the second scenario we share the decoder between two output channels. The first one samples the target character and the other one predicts the morphological annotation of the character. This multi-tasking approach forces the decoder to send morphology-aware information to the final layer which results in better predictions. In the third scenario we combine these two models. Section 3 provides more details on our models.

Together with different findings that will be discussed in the next sections, there are two main contributions in this paper. We redesigned and tuned the NMT framework for translating into MRLs. It is quite challenging to show the impact of external knowledge such as morphological information in neural models especially in the presence of large parallel corpora. However, our models are able to incorporate morphological information into decoding and boost its quality. We inject the decoder with morphological properties of the target language. Furthermore, the novel architecture proposed here is not limited to morphological information alone and is flexible enough to provide other types of information for the decoder.

2 NMT for MRLs

There are several models for NMT of MRLs which are designed to deal with morphological complexities. García-Martínez et al. (2016) and Sennrich and Haddow (2016) adapted the factored machine translation approach to neural models. Morphological annotations can be treated as extra factors in such models. Jean et al. (2015) proposed a model to handle very large vocabularies. Luong et al. (2015) addressed the problem of rare words and OOVs with the help of a post-translation phase to exchange unknown tokens with their potential translations. Sennrich et al. (2016) used subword units for NMT. The model relies on frequent subword units instead of words. Costa-jussà and Fonollosa (2016) designed a model for translating from MRLs. The model encodes source words with a convolutional module proposed by Kim et al. (2016). Each word is represented by a convolutional combination of its characters.

Luong and Manning (2016) used a hybrid model for representing words. In their model, unseen and complex words are encoded with a character-based representation, with other words encoded via the usual surface-form embeddings. Vylomova et al. (2016) compared differ-

ent representation models (word-, morpheme, and character-level models) which try to capture complexities on the source side, for the task of translating from MRLs.

Chung et al. (2016) proposed an architecture which benefits from different segmentation schemes. On the encoder side, words are segmented into subunits with the byte-pair segmentation model (*bpe*) (Sennrich et al., 2016), and on the decoder side, one target character is produced at each time step. Accordingly, the target sequence is treated as a long chain of characters without explicit segmentation. Grönroos et al. (2017) focused on translating from English into Finnish and implicitly incorporated morphological information into NMT through multi-task learning. Passban (2018) comprehensively studied the problem of translating MRLs and addressed potential challenges in the field.

Among all the models reviewed in this section, the network proposed by Chung et al. (2016) could be seen as the best alternative for translating into MRLs as it works at the character level on the decoder side and it was evaluated in different settings on different languages. Consequently, we consider it as a baseline model in our experiments.

3 Proposed Architecture

We propose a compatible neural architecture for translating into MRLs. The model benefits from subword- and character-level information and improves upon the state-of-the-art model of Chung et al. (2016). We manipulated the model to incorporate morphological information and developed three new extensions, which are discussed in Sections 3.1, 3.2, and 3.3.

3.1 The Embedded Morphology Table

In the first extension an additional table containing the morphological information of the target language is plugged into the decoder to assist with word formation. Each time the decoder samples from the target vocabulary, it searches the morphology table to find the most relevant affixes given its current state. Items selected from the table act as guiding signals to help the decoder sample a better character.

Our base model is an encoder-decoder model with attention (Bahdanau et al., 2014), implemented using gated recurrent units (GRUs) (Cho et al., 2014). We use a four-layer model in our experiments. Similar to Chung et al. (2016) and Wu et al. (2016), we use bidirectional units to encode the source sequence. Bidirectional GRUs are placed only at the input layer. The forward GRU reads the input sequence in its original order and the backward GRU reads the input in the reverse order. Each hidden state of the encoder in one time step is a concatenation of the forward and backward states at the same time step. This type of bidirectional processing provides a richer representation of the input sequence.

On the decoder side, one target character is sampled from a target vocabulary at each time step. In the original encoder-decoder model, the probability of predicting the next token y_i is estimated based on i) the current hidden state of the decoder, ii) the last predicted token, and iii) the context vector. This process can be formulated as $p(y_i|y_1, ..., y_{i-1}, \mathbf{x}) = g(h_i, y_{i-1}, \mathbf{c}_i)$, where $g(.)$ is a softmax function, y_i is the target token (to be predicted), $\mathbf{x}$ is the representation of the input sequence, h_i is the decoder's hidden state at the i-th time step, and $\mathbf{c}_i$ indicates the context vector which is a weighted summary of the input sequence generated by the attention module. $\mathbf{c}_i$ is generated via the procedure shown in (1):

$$\mathbf{c}_i = \sum_{j=1}^{n} \alpha_{ij} s_j$$
$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}; \quad e_{ij} = a(s_j, h_{i-1}) \tag{1}$$

where α_{ij} denotes the weight of the j-th hidden state of the encoder (s_j) when the decoder predicts the i-th target token, and $a()$ shows a combinatorial function which can be modeled through a simple feed-forward connection. n is the length of the input sequence.

In our first extension, the prediction probability is conditioned on one more constraint in addition to those three existing ones, as in $p(y_i|y_1, ..., y_{i-1}, \mathbf{x}) = g(h_i, y_{i-1}, \mathbf{c}_i, \mathbf{c}_i^m)$, where $\mathbf{c}_i^m$ is the morphological context vector and carries information from those useful affixes which can enrich the decoder's information. $\mathbf{c}_i^m$ is generated via an attention module over the morphology table which works in a similar manner to word-based attention model. The attention procedure for

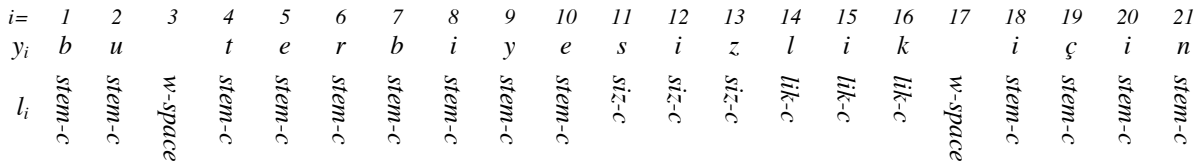


Figure 1: The target label that each output channel is supposed to predict when generating the Turkish sequence 'bu_1 $terbiyesizlik_2$ $için_3$' meaning '$because_3$ of_3 $this_1$ $rudeness_2$'.

generating $\mathbf{c}_i^m$ is formulated as in (2):

$$\mathbf{c}_i^m = \sum_{u=1}^{|\mathcal{A}|} \beta_{iu} f_u$$
$$\beta_{iu} = \frac{\exp\left(e_{iu}^m\right)}{\sum_{v=1}^{|\mathcal{A}|} \exp\left(e_{iv}\right)}; \quad e_{iu}^m = a^m(f_u, h_{i-1}) \tag{2}$$

where f_u represents the embedding of the u-th affix (u-th column) in the morphology/affix table $\mathcal{A}$, β_{iu} is the weight assigned to f_u when predicting the i-th target token, and a^m is a feed-forward connection between the morphology table and the decoder.

The attention module in general can be considered as a search mechanism, e.g. in the original encoder-decoder architecture the basic attention module finds the most relevant input words to make the prediction. In multi-modal NMT (Huang et al., 2016; Calixto et al., 2017) an extra attention module is added to the basic one in order to search the image input to find the most relevant image segments. In our case we have a similar additional attention module which searches the morphology table.

In this scenario, the morphology table including the target language's affixes can be considered as an external knowledge repository that sends auxiliary signals which accompany the main input sequence at all time steps. Such a table certainly includes useful information for the decoder. As we are not sure which affix preserves those pieces of useful information, we use an attention module to search for the best match. The attention module over the table works as a filter which excludes irrelevant affixes and amplifies the impact of relevant ones by assigning different weights (β values).

3.2 The Auxiliary Output Channel

In the first scenario, we embedded a morphology table into the decoder in the hope that it can enrich sampling information. Mathematically speaking, such an architecture establishes an extra constraint for sampling and can control the decoder's predictions. However, this is not the only way of constraining the decoder. In the second scenario, we define extra supervision to the network via another predictor (output channel). The first channel is responsible for generating translations and predicts one character at each time step, and the other one tries to understand the morphological status of the decoder by predicting the morphological annotation (l_i) of the target character.

The approach in the second scenario proposes a multi-task learning architecture, by which in one task we learn translations and in the other one morphological annotations. Therefore, all network modules –especially the last hidden layer just before the predictors– should provide information which is useful enough to make correct predictions in both channels, i.e. the decoder should preserve translation as well as morphological knowledge. Since we are translating into MRLs this type of mixed information (morphology+translation) can be quite useful.

In our setting, the morphological annotation l_i predicted via the second channel shows to which part of the word or morpheme the target character belongs, i.e. the label for the character is the morpheme that includes it. We clarify the prediction procedure via an example from our training set (see Section 4). When the Turkish word '*terbiyesizlik*' is generated, the first channel is supposed to predict *t*, *e*, *r*, up to *k*, one after another. For the same word, the second channel is supposed to predict *stem-C* for the fist 7 steps as the first 7 characters '*terbiye*' belong to the stem of the word. The *C* sign indicates that *stem-C* is a class label. The second channel should also predict *siz-C* when the first channel predicts *s* (eighth character), *i* (ninth character), and *z* (tenth character), and *lik-C* when the first channel samples the last three characters. Clearly, the second channel is a classifier which works over the {*stem-C*, *siz-C*, *lik-C*, ...} classes. Figure 1 illustrates a segment of a sentence including this Turkish word and explains which class

tags should be predicted by each channel.

To implement the second scenario we require a single-source double-target training corpus: [source sentence] → [sequence of target characters & sequence of morphological annotations] (see Section 4). The objective function should also be manipulated accordingly. Given a training set $\{\mathbf{x}_t, \mathbf{y}_t, \mathbf{m}_t\}_{t=1}^{T}$ the goal is to maximize the joint loss function shown in (3):

$$\lambda \sum_{t=1}^{T} \log P(\mathbf{y}_t|\mathbf{x}_t;\theta) + (1-\lambda) \sum_{t=1}^{T} \log P(\mathbf{m}_t|\mathbf{x}_t;\theta) \quad (3)$$

where $\mathbf{x}_t$ is the t-th input sentence whose translation is a sequence of target characters shown by $\mathbf{y}_t$. $\mathbf{m}_t$ is the sequence of morphological annotations and T is the size of the training set. θ is the set of network parameters and λ is a scalar to balance the contribution of each cost function. λ is adjusted on the development set during training.

3.3 Combining the Extended Output Layer and the Embedded Morphology Table

In the first scenario, we aim to provide the decoder with useful information about morphological properties of the target language, but we are not sure whether signals sent from the table are what we really need. They might be helpful or even harmful, so there should be a mechanism to control their quality. In the second scenario we also have a similar problem as the last layer requires some information to predict the correct morphological class through the second channel, but there is no guarantee to ensure that information in the decoder is sufficient for this sort of prediction. In order to address these problems, in the third extension we combine both scenarios as they are complementary and can potentially help each other.

The morphology table acts as an additional useful source of knowledge as it already consists of affixes, but its content should be adapted according to the decoder and its actual needs. Accordingly, we need a trainer to update the table properly. The extra prediction channel plays this role for us as it forces the network to predict the target language's affixes at the output layer. The error computed in the second channel is back-propagated to the network including the morphology table and updates its affix information into what the decoder actually needs for its prediction. Therefore, the second output channel helps us train better affix embeddings.

The morphology table also helps the second predictor. Without considering the table, the last layer only includes information about the input sequence and previously predicted outputs, which is not directly related to morphological information. The second attention module retrieves useful affixes from the morphology table and concatenates to the last layer, which means the decoder is explicitly fed with morphological information. Therefore, these two modules mutually help each other. The external channel helps update the morphology table with high-quality affixes (backward pass) and the table sends its high-quality signals to the prediction layer (forward pass). The relation between these modules and the NMT architecture is illustrated in Figure 2.

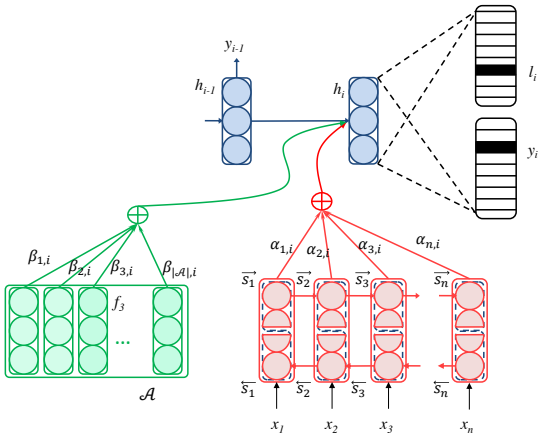


Figure 2: The architecture of the NMT model with an auxiliary prediction channel and an extra morphology table. This network includes only one decoder layer and one encoder layer. ⊕ shows the attention modules.

4 Experimental Study

As previously reviewed, different models try to capture complexities on the encoder side, but to the best of our knowledge the only model which proposes a technique to deal with complex constituents on the decoder side is that of Chung et al. (2016), which should be an appropriate baseline for our comparisons. Moreover, it outperforms other existing NMT models, so we prefer to compare our network to the best existing model. This model is referred to as CDNMT in our experiments. In the next sections first we explain our experimental setting, corpora, and how we build the morphology table (Section 4.1), and then report experimental results (Section 4.2).

4.1 Experimental Setting

In order to make our work comparable we try to follow the same experimental setting used in CDNMT, where the GRU size is 1024, the affix and word embedding size is 512, and the beam width is 20. Our models are trained using stochastic gradient descent with Adam (Kingma and Ba, 2015). Chung et al. (2016) and Sennrich et al. (2016) demonstrated that *bpe* boosts NMT, so similar to CDNMT we also preprocess the source side of our corpora using *bpe*. We use `WMT-15` corpora[1] to train the models, `newstest-2013` for tuning and `newstest-2015` as the test sets. For English–Turkish (En–Tr) we use the `OpenSubtitle2016` collection (Lison and Tiedemann, 2016). The training side of the English–German (En–De), English–Russian (En–Ru), and En–Tr corpora include 4.5, 2.1, and 4 million parallel sentences, respectively. We randomly select 3K sentences for each of the development and test sets for En–Tr. For all language pairs we keep the 400 most frequent characters as the target-side character set and replace the remainder (infrequent characters) with a specific character.

One of the key modules in our architecture is the morphology table. In order to implement it we use a look-up table whose columns include embeddings for the target language's affixes (each column represents one affix) which are updated during training. As previously mentioned, the table is intended to provide useful, morphological information so it should be initialized properly, for which we use a morphology-aware embedding-learning model. To this end, we use the neural language model of Botha and Blunsom (2014) in which each word is represented via a linear combination of the embeddings of its surface form and subunits, e.g. $\overrightarrow{terbiyesizlik} = \overrightarrow{terbiyesizlik} + \overrightarrow{terbiye} + \overrightarrow{siz} + \overrightarrow{lik}$. Given a sequence of words, the neural language model tries to predict the next word, so it learns sentence-level dependencies as well as intra-word relations. The model trains surface form and subword-level embeddings which provides us with high-quality affix embeddings.

Our neural language model is a recurrent network with a single 1000-dimensional GRU layer, which is trained on the target sides of our parallel corpora. The embedding size is 512 and we use a batch size of 100 to train the model. Before training the neural language model, we need to manipulate the training corpus to decompose words into morphemes for which we use Morfessor (Smit et al., 2014), an unsupervised morphological analyzer. Using Morfessor each word is segmented into different subunits where we consider the longest part as the stem of each word; what appears before the stem is taken as a member of the set of prefixes (there might be one or more prefixes) and what follows the stem is considered as a member of the set of suffixes.

Since Morfessor is an unsupervised analyzer, in order to minimize segmentation errors and avoid noisy results we filter its output and exclude subunits which occur fewer than 500 times.[2] After decomposing, filtering, and separating stems from affixes, we extracted several affixes which are reported in Table 2. We emphasize that there might be wrong segmentations in Morfessor's output, e.g. Turkish is a suffix-based language, so there are no prefixes in this language, but based on what Morfessor generated we extracted 11 different types of prefixes. We do not post-process Morfessor's outputs.

Language	Prefix	Suffix
German	75	160
Russian	110	260
Turkish	11	293

Table 2: The number of affixes extracted for each language.

Using the neural language model we train word, stem, and affix embeddings, and initialize the look-up table (but not other parts) of the decoder using those affixes. The look-up table includes high-quality affixes trained on the target side of the parallel corpus by which we train the translation model. Clearly, such an affix table is an additional knowledge source for the decoder. It preserves information which is very close to what the decoder actually needs. However, there might be some missing pieces of information or some incompatibility between the decoder and the table, so we do not freeze the morphology table during training, but let the decoder update it with respect to its needs in the forward and backward passes.

[1] `http://www.statmt.org/wmt15/`

[2] The number may seem a little high, but for a corpus with more than 115M words this is not a strict threshold in practice.

4.2 Experimental Results

Table 3 summarizes our experimental results. We report results for the *bpe→char* setting, which means the source token is a *bpe* unit and the decoder samples a character at each time step. CDNMT is the baseline model. Table 3 includes scores reported from the original CDNMT model (Chung et al., 2016) as well as the scores from our reimplementation. To make our work comparable and show the impact of the new architecture, we tried to replicate CDNMT's results in our experimental setting, we kept everything (parameters, iterations, epochs etc.) unchanged and evaluated the extended model in the same setting. Table 3 reports BLEU scores (Papineni et al., 2002) of our NMT models.

Model	En→De	En→Ru	En→Tr
CDNMT	21.33	26.00	-
CDNMT*	21.01	26.23	18.01
CDNMT*$_m$	**21.27**	**26.78**	**18.44**
CDNMT*$_o$	**21.39**	26.39	**18.59**
CDNMT*$_{mo}$	**21.48**	**26.84**	**18.70**

Table 3: CDNMT* is our implementation of CDNMT. m and o indicates that the base model is extended with the morphology table and the additional output channel, respectively. mo is the combination of both the extensions. The improvement provided by the boldfaced number compared to CDNMT* is statistically significant according to paired bootstrap re-sampling (Koehn, 2004) with $p = 0.05$.

Table 3 can be interpreted from different perspectives but the main findings are summarized as follows:

- The morphology table yields significant improvements for all languages and settings.
- The morphology table boosts the En–Tr engine more than others and we think this is because of the nature of the language. Turkish is an agglutinative language in which morphemes are clearly separable from each other, but in German and Russian morphological transformations rely more on fusional operations rather than agglutination.
- It seems that there is a direct relation between the size of the morphology table and the gain provided for the decoder, because Russian and Turkish have bigger tables and benefit from the table more than German which has fewer affixes.
- The auxiliary output channel is even more useful than the morphology table for all settings but En–Ru, and we think this is because of the morpheme-per-word ratio in Russian. The number of morphemes attached to a Russian word is usually more than those of German and Turkish words in our corpora, and it makes the prediction harder for the classifier (the more the number of suffixes attached to a word, the harder the classification task).
- The combination of the morphology table and the extra output channel provides the best result for all languages.

Figure 3 depicts the impact of the morphology table and the extra output channel for each language.

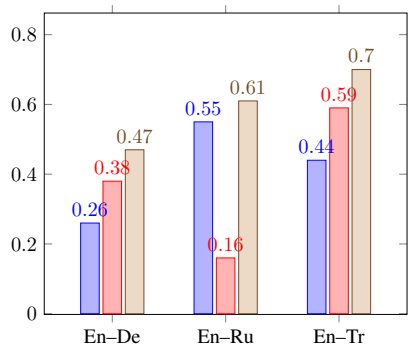


Figure 3: The y axis shows the difference between the BLEU score of CDNMT* and the extended model. The first, second, and third bars show the m, o, and mo extensions, respectively.

To further study our models' behaviour and ensure that our extensions do not generate random improvements we visualized some attention weights when generating '*terbiyesizlik*'. In Figure 4, the upper figure shows attention weights for all Turkish affixes, where the y axis shows different time steps and the x axis includes attention weights of all affixes (304 columns) for those time steps, e.g. the first row and the first column represents the attention weight assigned to the first Turkish affix when sampling ***t*** in '***t***erbiyesizlik'. While at the first glance the figure may appear to be somewhat confusing, but it provides some interesting insights which we elaborate next.

In addition to the whole attention matrix we also visualized a subset of weights to show how the morphology table provides useful information. In the second figure we study the behaviour of the morphology table for the first ($\boldsymbol{t}_1$), fifth ($\boldsymbol{i}_5$), ninth

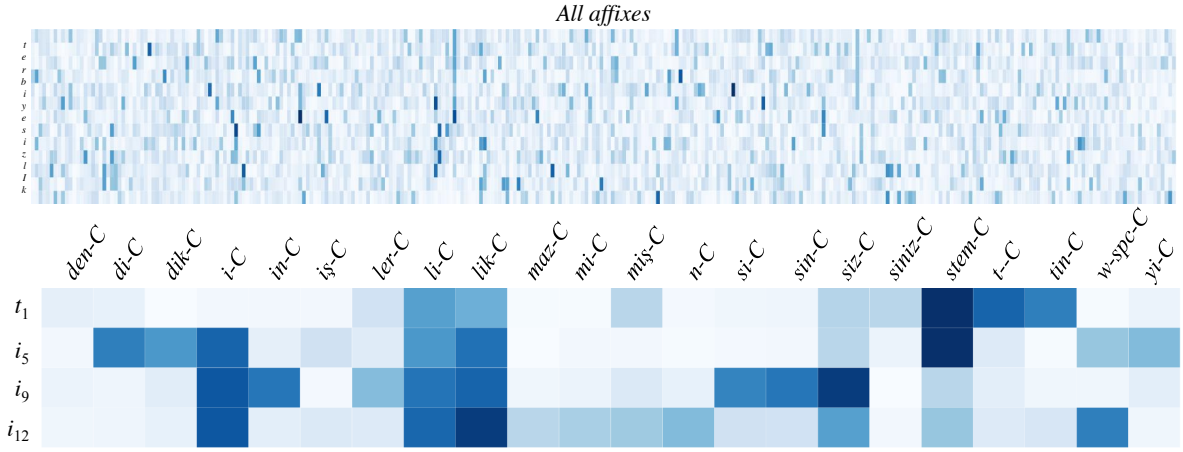


Figure 4: Visualizing the attention weights between the morphology table and the decoder when generating '*terbiyesizlik*.

($\boldsymbol{i}_9$), and twelfth ($\boldsymbol{i}_{12}$) time steps when generating the same Turkish word '$\boldsymbol{t}_1$*erbi*$\boldsymbol{i}_5$*yesi*$\boldsymbol{i}_9$*zli*$\boldsymbol{i}_{12}$*k*'. $\boldsymbol{t}_1$ is the first character of the word. We also have three $\boldsymbol{i}$ characters from different morphemes, where the first one is part of the stem, the second one belongs to the suffix '*siz*', and the third one to '*lik*'. It is interesting to see how the table reacts to the same character from different parts. For each time step we selected the top-10 affixes which have the highest attention weights. The set of top-10 affixes can be different for each step, so we made a union of those sets which gives us 22 affixes. The bottom part of Figure 4 shows the attention weights for those 22 affixes at each time step.

After analyzing the weights we observed interesting properties about the morphology table and the auxiliary attention module.[3] The main findings about the behaviour of the table are as follows:

- The model assigns high attention weights to *stem-C* for almost all time steps. However, the weights assigned to this class for $\boldsymbol{t}_1$ and $\boldsymbol{i}_5$ are much higher than those of affix characters (as they are part of the stem). The vertical lines in both figures approve this feature (bad behaviour).

- For some unknown reasons there are some affixes which have no direct relation to that particulate time step but they receive a high attention, such as *maz* in $\boldsymbol{t}_{12}$ (bad behaviour).

- For almost all time steps the highest attention weight belongs to the class which is expected to be selected, e.g. weights for ($\boldsymbol{i}_5$,*stem-C*) or ($\boldsymbol{i}_9$,*siz-C*) (good behaviour).

- The morphology table may send bad or good signals but it is consistent for similar or co-occurring characters, e.g. for the last three time steps $\boldsymbol{l}_{11}$, $\boldsymbol{i}_{12}$, and $\boldsymbol{k}_{13}$, almost the same set of affixes receives the highest attention weights. This consistency is exactly what we are looking for, as it can define a reliable external constraint for the decoder to guide it. Vertical lines on the figure also confirm this fact. They show that for a set of consecutive characters which belong to the same morpheme the attention module sends a signal from a particular affix (good behaviour).

- There are some affixes which might not be directly related to that time step but receive high attention weights. This is because those affixes either include the same character which the decoder tries to predict (e.g. *i-C* for $\boldsymbol{i}_4$ or *t-C* and *tin-C* for $\boldsymbol{t}_1$), or frequently appear with that part of the word which includes the target character (e.g. *mi-C* has a high weight when predicting $\boldsymbol{t}_1$ because $\boldsymbol{t}_1$ belongs to *terbiye* which frequently collocates with *mi-C*: *terbiye*+*mi*) (good behaviour).

Finally, in order to complete our evaluation study we feed the English-to-German NMT model with the sentence '*Terms and conditions for sending contributions to the BBC*', to show how the model behaves differently and generates a better target sentence. Translations generated by our models are illustrated in Table 4.

[3] Our observations are not based on this example alone as we studied other random examples, and the table shows consistent behaviour for all examples.

Reference:	*Geschäftsbedingungen für das Senden von Beiträgen an die BBC*
CDNMT*	*allgemeinen geschaftsbedingungen fur die versendung von Beiträgen an die BBC*
CDNMT$^*_{mo}$	*Geschäft s bedingungen für die versendung von Beiträgen zum BBC*

Table 4: Comparing translation results for the CDNMT* (baseline) and CDNMT$^*_{mo}$ (improved) models when the input sentence is '*Terms and conditions for sending contributions to the BBC*'.

The table demonstrates that our architecture is able to control the decoder and limit its selections, e.g. the word *'allgemeinen'* generated by the baseline model is redundant. There is no constraint to inform the baseline model that this word should not be generated, whereas our proposed architecture controls the decoder in such situations. After analyzing our model, we realized that there are strong attention weights assigned to the *w-space* (indicating white space characters) and *BOS* (beginning of the sequence) columns of the affix table while sampling the first character of the word *'Geschäft'*, which shows that the decoder is informed about the start point of the sequence. Similar to the baseline model's decoder, our decoder can sample any character including *'a'* of *'allgemeinen'* or *'G'* of *'Geschäft'*. Translation information stored in the baseline decoder is not sufficient for selecting the right character *'G'*, so the decoder wrongly starts with *'i'* and continues along a wrong path up to generating the whole word. However, our decoder's information is accompanied with signals from the affix table which force it to start with a better initial character, whose sampling leads to generating the correct target word.

Another interesting feature about the table is the new structure '*Geschäft s bedingungen'* generated by the improved model. As the reference translation shows, in the correct form these two structures should be glued together via *'s'*, which can be considered as an infix. As our model is supposed to detect this sort of intra-word relation, it treats the whole structure as two compounds which are connected to one another via an infix. Although this is not a correct translation and it would be trivial to post-edit into the correct output form, it is interesting to see how our mechanism forces the decoder to pay attention to intra-word relations.

Apart from these two interesting findings, the number of wrong character selections in the baseline model is considerably reduced in the improved model because of our enhanced architecture.

5 Conclusion and Future Work

In this paper we proposed a new architecture to incorporate morphological information into the NMT pipeline. We extended the state-of-the-art NMT model (Chung et al., 2016) with a morphology table. The table could be considered as an external knowledge source which is helpful as it increases the capacity of the model by increasing the number of network parameters. We tried to benefit from this advantage. Moreover, we managed to fill the table with morphological information to further boost the NMT model when translating into MRLs. Apart from the table we also designed an additional output channel which forces the decoder to predict morphological annotations. The error signals coming from the second channel during training inform the decoder with morphological properties of the target language. Experimental results show that our techniques were useful for NMT of MRLs.

As our future work we follow three main ideas. i) We try to find more efficient ways to supply morphological information for both the encoder and decoder. ii) We plan to benefit from other types of information such as syntactic and semantic annotations to boost the decoder, as the table is not limited to morphological information alone and can preserve other sorts of information. iii) Finally, we target sequence generation for fusional languages. Although our model showed significant improvements for both German and Russian, the proposed model is more suitable for generating sequences in agglutinative languages.

Acknowledgments

We thank our anonymous reviewers for their valuable feedback, as well as the Irish centre for high-end computing (www.ichec.ie) for providing computational infrastructures. This work has been supported by the ADAPT Centre for Digital Content Technology which is funded under the SFI Research Centres Programme (Grant 13/RC/2106) and is co-funded under the European Regional Development Fund.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*. Banff, Canada.

Jan A Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *The 3st International Conference on Machine Learning (ICML)*. Beijing, China, pages 1899–1907.

Iacer Calixto, Qun Liu, and Nick Campbell. 2017. Doubly-attentive decoder for multi-modal neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada, pages 1913–1924.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar, pages 1724–1734.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany, pages 1693–1703.

Marta R. Costa-jussà and José A. R. Fonollosa. 2016. Character-based neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Berlin, Germany, pages 357–361.

Mercedes García-Martínez, Loïc Barrault, and Fethi Bougares. 2016. Factored neural machine translation. *arXiv preprint arXiv:1609.04621* .

Stig-Arne Grönroos, Sami Virpioja, and Mikko Kurimo. 2017. Extending hybrid word-character neural machine translation with multi-task learning of morphological analysis. In *Proceedings of the Second Conference on Machine Translation*. Copenhagen, Denmark, pages 296–302.

Po-Yao Huang, Frederick Liu, Sz-Rung Shiang, Jean Oh, and Chris Dyer. 2016. Attention-based multimodal neural machine translation. In *Proceedings of the first Conference on Machine Translation*. pages 639–645.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China, pages 1–10. http://www.aclweb.org/anthology/P15-1001.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*. Phoenix, Arizona, USA, pages 2741–2749.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*. San Diego, USA.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Barcelona, Spain, pages 388–395.

Pierre Lison and Jörg Tiedemann. 2016. Opensubtitles2016: Extracting large parallel corpora from movie and TV subtitles. In *Proceedings of the 10th International Conference on Language Resources and Evaluation*. Portoroz, Slovenia, pages 923–929.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany, pages 1054–1063.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China, pages 11–19.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of Association for Computational Linguistics*. Pennsylvania, PA., USA, pages 311–318.

Peyman Passban. 2018. *Machine Translation of Morphologically Rich Languages Using Deep Neural Networks*. Ph.D. thesis, School of Computing, Dublin City University, Ireland.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation*. Berlin, Germany, pages 83–91.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany, pages 1715–1725.

Peter Smit, Sami Virpioja, Stig-Arne Grönroos, and Mikko Kurimo. 2014. Morfessor 2.0: Toolkit for statistical morphological segmentation. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*. Gothenburg, Sweden, pages 21–24.

Ekaterina Vylomova, Trevor Cohn, Xuanli He, and Gholamreza Haffari. 2016. Word representation models for morphologically rich languages in neural machine translation. *CoRR* abs/1606.04217. http://arxiv.org/abs/1606.04217.

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

Parsing Speech: A Neural Approach to Integrating Lexical and Acoustic-Prosodic Information

Trang Tran[*1], Shubham Toshniwal[*2], Mohit Bansal[3], Kevin Gimpel[2], Karen Livescu[2], Mari Ostendorf[1]

[1]Electrical Engineering, University of Washington
[2]Toyota Technological Institute at Chicago
[3]Department of Computer Science, UNC Chapel Hill

{ttmt001, ostendor}@uw.edu, mbansal@cs.unc.edu,
{shtoshni, kgimpel, klivescu}@ttic.edu

Abstract

In conversational speech, the acoustic signal provides cues that help listeners disambiguate difficult parses. For automatically parsing spoken utterances, we introduce a model that integrates transcribed text and acoustic-prosodic features using a convolutional neural network over energy and pitch trajectories coupled with an attention-based recurrent neural network that accepts text and prosodic features. We find that different types of acoustic-prosodic features are individually helpful, and together give statistically significant improvements in parse and disfluency detection F1 scores over a strong text-only baseline. For this study with known sentence boundaries, error analyses show that the main benefit of acoustic-prosodic features is in sentences with disfluencies, attachment decisions are most improved, and transcription errors obscure gains from prosody.

1 Introduction

While parsing has become a relatively mature technology for written text, parser performance on conversational speech lags behind. Speech poses challenges for parsing: transcripts may contain errors and lack punctuation; even perfect transcripts can be difficult to handle because of disfluencies (restarts, repetitions, and self-corrections), filled pauses ("um", "uh"), interjections ("like"), parentheticals ("you know", "I mean"), and sentence fragments. Some of these phenomena can be handled in standard grammars, but disfluencies typically require extensions of the model. Different approaches have been explored in both constituency parsing (Charniak and Johnson, 2001; Johnson and Charniak, 2004) and dependency parsing (Rasooli and Tetreault, 2013; Honnibal and Johnson, 2014).

Despite these challenges, speech carries helpful extra information – beyond the words – associated with the prosodic structure of an utterance and encoded via variation in timing and intonation. Speakers pause in locations that are correlated with syntactic structure (Grosjean et al., 1979), and listeners use prosodic structure in resolving syntactic ambiguities (Price et al., 1991). Prosodic cues also signal disfluencies by marking the interruption point (Shriberg, 1994). However, most speech parsing systems in practice take little advantage of these cues. Our study focuses on this last challenge, aiming to incorporate prosodic cues in a neural parser, handling disfluencies as constituents via a neural attention mechanism.

A challenge of incorporating prosody in parsing is that multiple acoustic cues interact to signal prosodic structure, including pauses, lengthening, fundamental frequency modulation, and spectral shape. These cues also vary with the phonetic segment, emphasis, emotion and speaker, so feature extraction typically involves multiple time windows and normalization techniques. The most successful constituent parsers have mapped these features to prosodic boundary posteriors by using labeled training data (Kahn et al., 2005; Hale et al., 2006; Dreyer and Shafran, 2007). The approach proposed here takes advantage of advances in neural networks to automatically learn a good feature representation without the need to explicitly represent prosodic constituents. To narrow the scope of this work and facilitate error analysis, our experiments use known transcripts and sentence segmentation.

Our work offers the following contributions. We introduce a framework for directly integrating acoustic-prosodic features with text in a neural encoder-decoder parser that does not require hand-annotated prosodic structure. We demonstrate improvements in constituent parsing of conversational

[*]Equal Contribution.

Proceedings of NAACL-HLT 2018, pages 69–81
New Orleans, Louisiana, June 1 - 6, 2018.

speech over a high-quality text-only parser and provide analyses showing where prosodic features help and that assessment of their utility is affected by human transcription errors.

2 Task and Model Description

Our model maps a sequence of word-level input features to a linearized parse output sequence. The word-level input feature vector consists of the concatenation of (learnable) word embeddings $\boldsymbol{e_i}$ and several types of acoustic-prosodic features, described in Section 2.3.

2.1 Task Setup

We assume the availability of a training treebank of conversational speech (in our case, Switchboard-NXT (Calhoun et al., 2010)) and corresponding constituent parses. The transcriptions are preprocessed by removing punctuation and lower-casing all text to better mimic the speech recognition setting. Following Vinyals et al. (2015), the parse trees are linearized, and pre-terminals are normalized as "XX" (see Appendix A.1).

2.2 Encoder-Decoder Parser

Our attention-based encoder-decoder model is similar to the one used by Vinyals et al. (2015). The *encoder* is a deep long short-term memory recurrent neural network (LSTM-RNN) (Hochreiter and Schmidhuber, 1997) that reads in a word-level inputs,[1] represented as a sequence of vectors $\boldsymbol{x} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{T_s})$, and outputs high-level features $\boldsymbol{h} = (\boldsymbol{h}_1, \cdots, \boldsymbol{h}_{T_s})$ where $\boldsymbol{h}_i = \text{LSTM}(\boldsymbol{x}_i, \boldsymbol{h}_{i-1})$.[2]

The *parse decoder* is also a deep LSTM-RNN that predicts the linearized parse sequence $\boldsymbol{y} = (y_1, \cdots, y_{T_o})$ as follows:

$$P(\boldsymbol{y}|\boldsymbol{x}) = \prod_{t=1}^{T_o} P(y_t|\boldsymbol{h}, \boldsymbol{y_{<t}})$$

In attention-based models, the posterior distribution of the output y_t at time step t is given by:

$$P(y_t|\boldsymbol{h}, \boldsymbol{y_{<t}}) = \text{softmax}(\boldsymbol{W}_s[\boldsymbol{c}_t; \boldsymbol{d}_t] + \boldsymbol{b}_s),$$

where vector $\boldsymbol{b}_s$ and matrix $\boldsymbol{W}_s$ are learnable parameters; $\boldsymbol{c}_t$ is referred to as a *context vector* that summarizes the encoder's output $\boldsymbol{h}$; and $\boldsymbol{d}_t$ is the decoder hidden state at time step t, which captures the previous output sequence context $\boldsymbol{y_{<t}}$.

$$u_{it} = \boldsymbol{v}^\top \tanh(\boldsymbol{W}_1\boldsymbol{h}_i + \boldsymbol{W}_2\boldsymbol{d}_t + \boldsymbol{b}_a)$$

$$\boldsymbol{\alpha}_t = \text{softmax}(\boldsymbol{u}_t) \qquad \boldsymbol{c}_t = \sum_{i=1}^{T_s} \alpha_{ti}\boldsymbol{h}_i$$

where vectors $\boldsymbol{v}$, $\boldsymbol{b}_a$ and matrices $\boldsymbol{W}_1$, $\boldsymbol{W}_2$ are learnable parameters; $\boldsymbol{u}_t$ and $\boldsymbol{\alpha}_t$ are the attention score and attention weight vector, respectively, for decoder time step t.

The above attention mechanism is only *content*-based, i.e., it is only dependent on $\boldsymbol{h}_i$, $\boldsymbol{d}_t$. It is not *location*-aware, i.e., it does not consider the "location" of the previous attention vector. For parsing conversational text, location awareness is beneficial since disfluent structures can have duplicate words/phrases that may "confuse" the attention mechanism.

In order to make the model location-aware, the attention mechanism takes into account the previous attention weight vector $\boldsymbol{\alpha}_{t-1}$. In particular, we use the attention mechanism proposed by Chorowski et al. (2015), in which $\boldsymbol{\alpha}_{t-1}$ is represented via a feature vector $\boldsymbol{f}_t = \boldsymbol{F} * \boldsymbol{\alpha}_{t-1}$, where $\boldsymbol{F} \in \mathcal{R}^{k \times r}$ represents k learnable convolution filters of width r. The filters are used for performing 1-D convolution over $\boldsymbol{\alpha}_{t-1}$ to extract k features $\boldsymbol{f}_{ti}$ for each time step i of the input sequence. The extracted features are then incorporated in the alignment score calculation as:

$$u_{it} = \boldsymbol{v}^\top \tanh(\boldsymbol{W}_1\boldsymbol{h}_i + \boldsymbol{W}_2\boldsymbol{d}_t + \boldsymbol{W}_f\boldsymbol{f}_{ti} + \boldsymbol{b}_a)$$

where $\boldsymbol{W}_f$ is another learnable parameter matrix. Finally, the decoder state $\boldsymbol{d}_t$ is computed as $\boldsymbol{d}_t = \text{LSTM}([\tilde{\boldsymbol{y}}_{t-1}; \boldsymbol{c}_{t-1}], \boldsymbol{d}_{t-1})$, where $\tilde{\boldsymbol{y}}_{t-1}$ is the embedding vector corresponding to the previous output symbol y_{t-1}. As we will see in Sec. 4.1, the location-aware attention mechanism is especially useful for handling disfluencies.

2.3 Acoustic-Prosodic Features

In previous work using encoder-decoder models for parsing (Vinyals et al., 2015; Luong et al., 2016), vector $\boldsymbol{x}_i$ is simply the word embedding $\boldsymbol{e}_i$ of the word at position i of the input sentence. For parsing conversational speech, we can incorporate acoustic-prosodic features. Here we explore four types of features widely used in computational models of prosody: pauses, duration lengthening, fundamental frequency, and energy. Since prosodic cues are

[1] As in Vinyals et al. (2015) the input sequence is processed in reverse order, as shown in Figure 1.

[2] For brevity we omit the LSTM equations. The details can be found, e.g., in Zaremba et al. (2014).

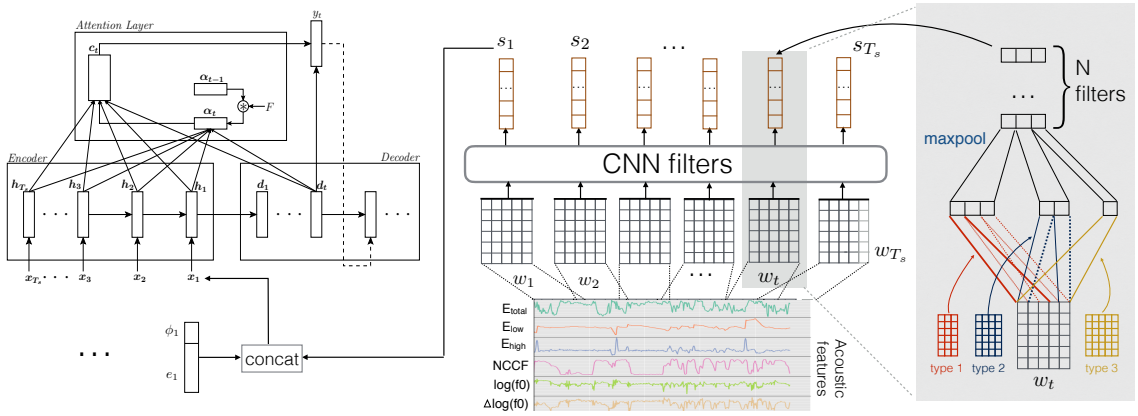


Figure 1: Left – An attention-based encoder-decoder reading the input $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{T_s}$, where $\boldsymbol{x}_i = [\boldsymbol{e}_i \ \boldsymbol{\phi}_i \ \boldsymbol{s}_i]$ is composed of word embeddings $\boldsymbol{e}_i$, prosodic features $\boldsymbol{\phi}_i$, and learned (CNN-based) features $\boldsymbol{s}_i$. The encoder reads the input in *reverse* order and the decoder outputs the linearized parse $y_1, \cdots, y_t, \cdots$. Right – Detailed illustration of acoustic-prosodic feature learning module. CNN features are computed from input energy and pitch features; here the CNN filter parameters are $m = 3$ and $w = [3, 4, 5]$.

at sub- and multi-word time scales, they are integrated with the encoder-decoder using different mechanisms.

All features are extracted from transcriptions that are time-aligned at the word level.[3] We use time alignments associated with the corpus to be consistent with other studies. In a small number of cases, the time alignment for a particular word boundary is missing. Some cases are due to tokenization. For example, contractions, such as *don't* in the original transcript, are treated as separated words for the parser (*do* and *n't*), and the internal word boundary time is missing. In such cases, these internal times are estimated. In other cases, there are transcription mismatches that lead to missing time alignments, where we cannot estimate times. For the roughly 1% of sentences where time alignments are missing, we simply back off to the text-based parser.

Pause. The pause feature vector $\boldsymbol{p}_i$ for word i is the concatenation of pre-word pause feature $\boldsymbol{p}_{pre,i}$ and post-word pause feature $\boldsymbol{p}_{post,i}$, where each subvector is a learned embedding for 6 pause categories: no pause, missing, $0 < p \leq 0.05$ s, 0.05 s $< p \leq 0.2$ s, $0.2 < p \leq 1$ s, and $p > 1$ s (including turn boundaries). The bins are chosen based on the observed distribution (see Appendix A.1). We did not use (real-valued) pause duration directly, for two main reasons: (1) to handle missing time alignments; and (2) duration of pause does not matter beyond a threshold (e.g. $p > 1$ s).

Word duration. Both word duration and word-final duration lengthening are strong cues to prosodic phrase boundaries (Wightman et al., 1992; Pate and Goldwater, 2013). The word duration feature δ_i is computed as the actual word duration divided by the mean duration of the word, clipped to a maximum value of 5. The sample mean is used for frequent words (count ≥ 15). For infrequent words we estimate the mean as the sum over the sample means for the phonemes in the word's dictionary pronunciation. We refer to the manually defined prosodic feature pair of $\boldsymbol{p}_i$ and δ_i as ϕ_i.

Fundament frequency (f0) and Energy (E) contours (f0/E). We use a CNN to automatically learn the mapping from the time series of f0/E features to a word-level vector. The contour features are extracted from 25-ms frames with 10-ms hops using Kaldi (Povey et al., 2011). Three f0 features are used: warped Normalized Cross Correlation Function (NCCF), log-pitch with Probability of Voicing (POV)-weighted mean subtraction over a 1.5-second window, and the estimated derivative (delta) of the raw log pitch. Three energy features are extracted from the Kaldi 40-mel-frequency filter bank features: E_{total}, the log of total energy normalized by dividing by the speaker side's max total energy; E_{low}, the log of total energy in the lower 20 mel-frequency bands, normalized by total energy, and E_{high}, the log of total energy in the higher 20 mel-frequency bands, normalized by total energy. Multi-band energy features are used as a

[3] The assumption of known word alignments is standard for prosodic feature extraction in many spoken language processing studies. Time alignments can be obtained as a by-product of recognition or from forced alignment.

simple mechanism to capture articulatory strengthening at prosodic constituent onsets (Fourgeron and Keating, 1997).

Figure 1 summarizes the feature learning approach. The f0 and E features are processed at the word level: each sequence of frames corresponding to a time-aligned word (and potentially its surrounding context) is convolved with N filters of m sizes (a total of mN filters). The motivation for the multiple filter sizes is to enable the computation of features that capture information on different time scales. For each filter, we perform a 1-D convolution over the 6-dimensional f0/E features with a stride of 1. Each filter output is max-pooled, resulting in mN-dimensional speech features $\boldsymbol{s}_i$. Our overall acoustic-prosodic feature vector is the concatenation of $\boldsymbol{p}_i$, δ_i, and $\boldsymbol{s}_i$ in various combinations.

3 Experiments

3.1 Dataset

Our core corpus is Switchboard-NXT (Calhoun et al., 2010), a subset of the Switchboard corpus (Godfrey and Holliman, 1993): 2,400 telephone conversations between strangers; 642 of these were hand-annotated with syntactic parses and further augmented with richer layers of annotation facilitated by the NITE XML toolkit (Calhoun et al., 2010). Our sentence segmentations and syntactic trees are based on the annotations from the Treebank set, with a few manual corrections from the NXT release. This core dataset consists of 100K sentences, totaling 830K tokens forming a vocabulary of 13.5K words. We use the time alignments available from NXT, which is based on a corrected word transcript that occasionally differs from the Treebank, leading to some missing time alignments. We follow the sentence boundaries defined by the parsed data available,[4] and the data split (90% train; 5% dev; 5% test) defined by related work done on Switchboard (Charniak and Johnson, 2001; Kahn et al., 2005; Honnibal and Johnson, 2014).

3.2 Evaluation Metrics and Baselines

The standard evaluation metric for constituent parsing is the *parseval* metric which uses bracketing precision, recall, and F1, as in the canonical implementation of EVALB.[5] For written text, punctuation is sometimes represented as part of the sequence and impacts the final score, but for speech the punctuation is not explicitly available so it does not contribute to the score. Another challenge of transcribed speech is the presence of disfluencies. Speech repairs are indicated under "EDITED" nodes in Switchboard parse trees, which include structure under these nodes that is not of interest for simple text clean-up. Therefore, some studies report *flattened-edit parseval* F1 scores ("flat-F1"), which is *parseval* computed on trees where the structure under edit nodes has been eliminated so that all leaves are immediate children. We report both scores for the baseline text-only model showing that the differences are small, then use the standard *parseval* F1 score for most results.[6]

Disfluencies are particularly problematic for statistical parsers, as explained by Charniak and Johnson (2001), and some systems incorporate a separate disfluency detection stage. For this reason, and because it is useful for understanding system performance, most studies also report disfluency detection performance, which is measured in terms of the F1 score for detecting whether a word is in an edit region. Our approach does not involve a separate disfluency detection stage, but identifies disfluencies implicitly via the parse structure. Consequently, the disfluency detection results are not competitive with work that directly optimize for disfluency detection. We report disfluency detection scores primarily as a diagnostic.

Most previous work on integrating prosody and parsing has used the Switchboard corpus, but it is still difficult to compare results because of differences in constraints, objectives and the use of constituent vs. dependency structure, as discussed further in Section 6. The most relevant prior studies (on constituent parsing) that we compare to are a bit old. The text-only result from our neural parser represents a stronger baseline and is important for decoupling the impact of prosody vs. the parsing framework.

3.3 Model Training and Inference

Both the encoder and decoder are 3-layer deep LSTM-RNNs with 256 hidden units in each layer. For the location-aware attention, the convolution operation uses 5 filters of width 40 each. We use 512-dimensional embedding vectors to repre-

[4] Note that these sentence units can be inconsistent with other layers of Switchboard annotations, such as *slash units*.

[5] http://nlp.cs.nyu.edu/evalb/

[6] A variant of the "flat-F1" score is used in (Charniak and Johnson, 2001; Kahn et al., 2005), which uses a relaxed edited node precision and recall but also ignores filled pauses.

sent words and linearized parsing symbols, such as "(S".[7]

A number of configurations are explored for the acoustic-prosodic features, tuning based on dev set parsing performance. Pause embeddings are tuned over $\{4, 16, 32\}$ dimensions. For the CNN, we try different configurations of filter widths $w \in \{[10, 25, 50], [5, 10, 25, 50]\}$ and number of filters $N \in \{16, 32, 64, 128\}$ for each filter width.[8] These filter size combinations are chosen to capture f0 and energy phenomena on various levels: $w = 5, 10$ for sub-word, $w = 25$ for word, and $w = 50$ for word and extended context. Our best model uses 32-dimensional pause embeddings and $N = 32$ filters of widths $w = [5, 10, 25, 50]$, which corresponds to $m = 4$ and 128 filters.

For optimization we use Adam (Kingma and Ba, 2014) with a minibatch size of 64. The initial learning rate is 0.001 which is decayed by a factor of 0.9 whenever training loss, calculated after every 500 updates, degrades relative to the worst of its previous 3 values. All models are trained for up to 50 epochs with early stopping. For regularization, dropout with 0.3 probability is applied on the output of all LSTM layers (Pham et al., 2014).

For inference, we use a greedy decoder to generate the linearized parse. The output token with maximum posterior probability is chosen at every time step and fed as input in the next time step. The decoder stops upon producing the end-of-sentence symbol. We use TensorFlow (Abadi et al., 2015) to implement all models.[9]

4 Results

4.1 Text-only Results

Model	F1	flat-F1	fluent	disf
Berkeley	85.41	85.91	90.52	83.08
C-attn	83.33	83.20	90.86	79.94
CL-attn	**87.85**	**87.68**	**92.07**	**85.95**

Table 1: Scores of text-only models on the dev set: 2044 fluent and 3725 disfluent sentences. C-attn denotes *content*-only attention; CL-attn denotes *content+location* attention.

[7]The number of layers, dimension of hidden units, dimension of embedding, and convolutional attention filter parameters of the text-only parser were explored in earlier experiments on the development set and then fixed as described.

[8]Note that a filter of width 10 has size 6×10, since the features are of dimension 6.

[9]Our code resources can be found in Appendix A.1.

Model	Parse	Disf
Berkeley (text only)	85.41	62.45
CL-attn (text only)	87.85	79.50
CL-attn text and		
+ p	88.37	80.24
+ δ	88.04	77.41
+ p + δ	88.21	80.84
+ f0/E-CNN	88.52	80.81
+ p + f0/E-CNN	88.45	**81.19**
+ δ + f0/E-CNN	88.44	80.09
+ p + δ + f0/E-CNN	**88.59**	80.84

Table 2: Parse and disfluency detection F1 scores on the dev set. Flat-F1 scores were consistently 0.1%-0.3% lower for our models, but 0.2% higher for the Berkeley parser (85.64).

We first show our results on the model using only text (i.e. $\boldsymbol{x}_i = \boldsymbol{e}_i$) to establish a strong baseline, on top of which we can add acoustic-prosodic features. We experiment with the *content*-only attention model used by Vinyals et al. (2015) and the *content+location* attention of Chorowski et al. (2015). For comparison with previous non-neural models, we use a high-quality latent-variable parser, the Berkeley parser (Petrov et al., 2006), retrained on our Switchboard data. Table 1 compares the three text-only models. In terms of F1, the *content+location* attention beats the Berkeley parser by about 2.5% and *content*-only attention by about 4.5%. Flat-F1 scores for both encoder-decoder models is lower than their corresponding F1 scores, suggesting that the encoder-decoder models do well on predicting the internal structure of EDIT nodes while the reverse is true for the Berkeley parser.

To explain the gains of *content+location* attention over *content*-only attention, we compare their scores on fluent (without EDIT nodes) and disfluent sentences, shown in Table 1. It is clear that most of the gains for *content+location* attention are from disfluent sentences. A possible explanation is the presence of duplicate words or phrases in disfluent sentences, which can be problematic for a *content*-only attention model. Since our best model is the *content+location* attention model, we will henceforth refer to it as the "CL-attn" text-only model. All models using acoustic-prosodic features are extensions of this model, which provides a strong text-only baseline.

Model	Parse	Disf
CL-attn	87.79 (0.11)	78.65 (0.46)
best model	88.15 (0.41)	80.48 (0.70)

Table 3: Parse and disfluency detection F1 scores on the dev set: mean (and standard deviation) over 10 runs for the baseline text-only model (CL-attn) and the best model with prosody.

Model	Parse	Disfl
Berkeley	85.87	63.44
CL-attn	87.99	76.69
best model	**88.50**	**77.47**

Table 4: Parse and disfluency detection F1 scores on the test set. The best model has *statistically significant* gains over the text-only baseline with p-value < 0.02.

4.2 Adding Acoustic-Prosodic Features

We extend our CL-attn model with the three kinds of acoustic-prosodic features: pause (p), word duration (δ), and CNN mappings of fundamental frequency (f0) and energy (E) features (f0/E-CNN).

The results of several model configurations on our dev set are presented in Table 2. First, we note that adding any combination of acoustic-prosodic features (individually or in sets) improves performance over the text-only baseline. However, certain combinations of acoustic-prosodic features are not always better than their subsets. The *text* + p + δ + *f0/E-CNN* model that uses all three types of features has the best performance with a gain of 0.7% over the already-strong text-only baseline. We will henceforth refer to the *text* + p + δ + *f0/E-CNN* model as our "best model".

As a robustness check, we report results of averaging 10 runs on the CL-attn text-only and the best model in Table 3. We performed a bootstrap test (Efron and Tibshirani, 1993) that simulates 10^5 random test draws on the models giving median performance on the dev set. These *median* models gave a statistically significant difference between the text-only and best model (p-value < 0.02). Additionally, a simple t-test over the two sets of 10 results also shows statistical significance p-value < 0.03.

Table 4 presents the results on the test set. Again, adding the acoustic-prosodic features improves over the text-only baseline. The gains are statistically significant for the best model with p-value < 0.02, again using a bootstrap test with simulated 10^5 random test draws on the two models.

Model	Parse	Disfl
Text Only		
Kahn et al. (2005)	86.4	78.2
Hale et al. (2006)	71.16	41.7
CL-attn (text only)	87.99	76.7
Text + Prosody		
Kahn et al. (2005)	86.6	78.2
Hale et al. (2006)	71.05	36.2
best model	88.50	77.5

Table 5: Parse and disfluency detection F1 scores on the test set comparing to other reported results.

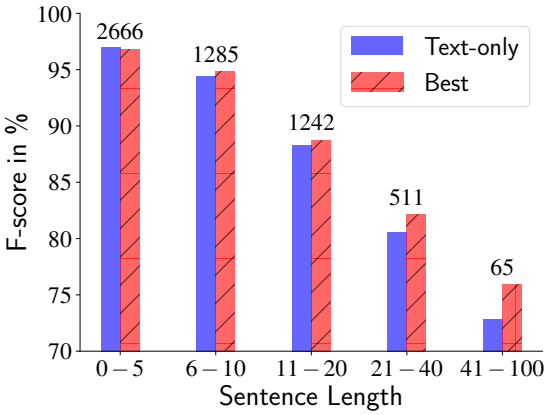


Figure 2: F1 scores of the text-only model and our best model as a function of sentence length.

Table 5 includes results from prior studies that compare systems using text alone with ones that incorporate prosody, given hand transcripts and sentence segmentation. It is difficult to compare systems directly, because of the many differences in the experimental set-up. For example, the original Charniak and Johnson (2001) result (reporting F=85.9 for parsing and F=78.2 for disfluencies) leverages punctuation in the text stream, which is not realistic for speech transcripts and not used in most other work. Our work benefits from more text training material than others, but others benefit from gold part-of-speech tags. Kahn et al. (2005) use a modified sentence segmentation. There are probably minor differences in handling of word fragments and scoring edit regions. Thus, this table primarily shows that our framework leads to more benefits from sentence-internal prosodic cues than others have obtained.

5 Analysis

Effect of sentence length. Figure 2 shows performance differences between our best model and the text-only model for varying sentence lengths.

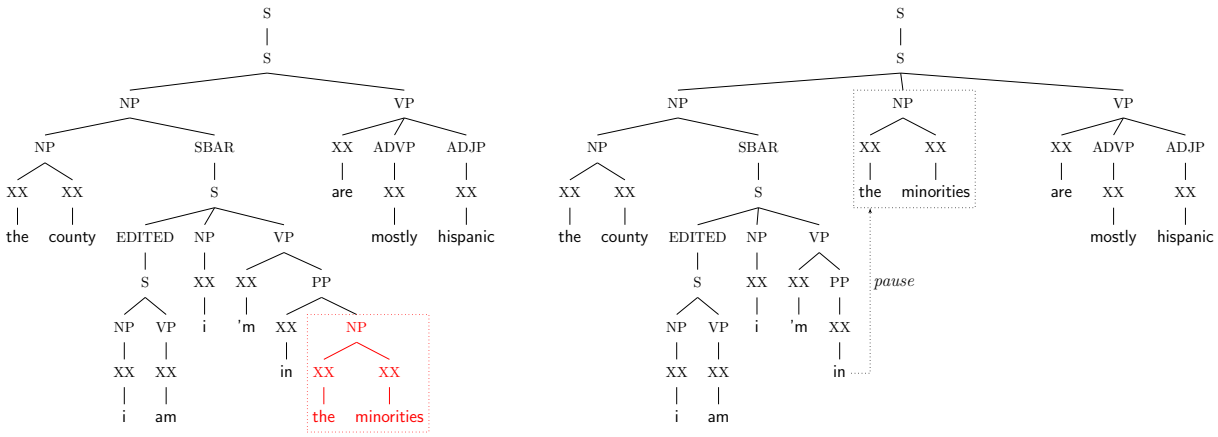


Figure 3: An example sentence from development data – *the county i am i 'm in [pause] the minorities are mostly hispanic*. The text-only parser (on the left) makes a PP Attachment error. The prosody-enhanced parser (on the right) uses the pause indicator to correctly predict a constituent change after the word *in*.

Model	fluent	disfluent
text-only	92.07	85.90
best model	92.03	**87.02**

Table 6: Dev set F1-score of text-only and best model on fluent (2029) vs. disfluent (3689) sentences.[10]

Both models do worse on longer sentences, as expected since the corresponding parse trees tend to be more complex. The performance difference between our best model and the text-only model increases with sentence length. This is likely because longer sentences more often have multiple prosodic phrases and disfluencies.

Effect of disfluencies. Table 6 presents parse scores on the subsets of fluent and disfluent sentences, showing that the performance gain is in the disfluent set (65% of the dev set sentences). Because sentence boundaries are given, and so many fluent sentences in spontaneous speech are short, there is less potential for benefit from prosody in the fluent set.

Types of errors. We use the Berkeley Parser Analyzer (Kummerfeld et al., 2012) to compare the types of errors made by the different parsers.[10] Table 7 presents the relative error reductions over the text-only baseline achieved by the text + p model and our best model for disfluent sentences. The two models differ in the types of error reductions they provide. Including pause information gives largest improvements on PP attachment and Modifier attachment errors. Adding the remaining acoustic-prosodic features helps to correct more types of attachment errors, especially VP and NP attachment. Figure 3 demonstrates one case where the pause feature helps in correcting a PP attachment error made by a text-only parser. Other interesting examples (see Appendix A.2) suggest that the learned f0/E features help reduce NP attachment errors where the audio reveals a prominent word at the constituent boundary, even though there is no pause at that word.

Error Type	Disfluent Sentences	
	text + p	best model
Clause Att.	5.7%	1.3%
Diff. Label	7.6%	4.2%
Modifier Att.	9.7%	19.1%
NP Att.	-2.7%	14.5%
NP Internal	7.8%	7.4%
PP Att.	10.1%	7.8%
1-Word Phrase	6.3%	6.8%
Unary	-1.1%	8.9%
VP Att.	0.0%	12.0%

Table 7: Relative error reduction over the text-only baseline in the disfluent subset (3689 sentences) of the development set. Shown here are the most frequent error types (with count $\geq$ 100 for the text-only model).

Effect of transcription errors. The results and analyses so far have assumed that we have reliable transcripts. In fact, the original transcripts contained errors, and the Treebank annotators used these without reference to audio files. Mississippi State University (MS-State) ran a clean-up project

[10] This analysis omits the 1% of the sentences that did not have timing information.

that produced more accurate word transcripts and time alignments (Deshmukh et al., 1998). The NXT corpus provides reconciliation between Treebank and MS-State transcripts in terms of annotating missed/extra/substituted words, but parses were not re-annotated. The transcript errors mean that the acoustic signal is inconsistent with the "gold" parse tree. Below are some examples of "fluent" sentences (according to the Treebank transcripts) with transcription errors, for which prosodic features "hurt" parsing. Words that transcribers missed are in brackets and those inserted are underlined.

S1: and because <uh> like if your spouse died <all of a sudden you be> all alone it 'd be nice to go someplace with people similar to you to have friends

S2: uh uh <i have had> my wife 's picked up a couple of things saying uh boy if we could refinish that 'd be a beautiful piece of furniture

Multi-syllable errors are especially problematic, leading to serious inconsistencies between the text and the acoustic signal. Further, the missed words lead to an incorrect attachment in the "gold" parse in S1 and a missing restart edit in S2. Indeed, for sentences with consecutive transcript errors, which we expect to impact the prosodic features, there is a statistically significant (p-value < 0.05) negative effect on parsing with prosody. Not included in this analysis are sentence boundary errors, which also change the "gold" parse. Thus, prosody may be more useful than results here indicate.

6 Related Work

Related work on parsing conversational speech has mainly addressed four problems: speech recognition errors, unknown sentence segmentation, disfluencies, and integrating prosodic cues. Our work addresses the last two problems, which involve studies based on hand-transcribed text and known sentence boundaries, as in much speech parsing work. The related studies are thus the focus of this discussion. We describe studies using the Switchboard corpus, since it has dominated work in this area, being the largest source of treebanked English spontaneous speech.

One major challenge of parsing conversational speech is the presence of disfluencies, which are grammatical and prosodic interruptions. Disfluencies include repetitions ('I am + I am'), repairs ('I am + we are'), and restarts ('What I + Today is the...'), where the '+' corresponds to an interruption point. Repairs often involve parallel grammatical constructions, but they can be more complex, involving hedging, clarifications, etc. Charniak and Johnson (Charniak and Johnson, 2001; Johnson and Charniak, 2004) demonstrated that disfluencies are different in character than other constituents and that parsing performance improves from combining a PCFG parser with a separate module for disfluency detection via parse rescoring. Our approach does not use a separate disfluency detection module; we hypothesized that the location-sensitive attention model helps handle these differences based on analysis of the text-only results (Table 1). However, more explicit modeling of disfluency pattern match characteristics in a dependency parser (Honnibal and Johnson, 2014) leads to better disfluency detection performance (F = 84.1 vs. 76.7 for our text only model). Pattern match features also benefit a neural model for disfluency detection alone (F = 87.0) (Zayats et al., 2016), and similar gains are observed by formulating disfluency detection in a transition-based framework (F = 87.5) (Wang et al., 2017). Experiments with oracle disfluencies as features improve the CL-attn text-only parsing performance from 87.85 to 89.38 on the test set, showing that more accurate disfluency modeling is a potential area of improvement.

It is well known that prosodic features play a role in human resolution of syntactic ambiguities, with more than two decades of studies seeking to incorporate prosodic features in parsing. A series of studies looked at constituent parsing informed by the presence (or likelihood) of prosodic breaks at word boundaries (Kahn et al., 2004, 2005; Hale et al., 2006; Dreyer and Shafran, 2007). Our approach improves over performance of these systems using raw acoustic features, without the need for hand-labeling prosodic breaks. The gain is in part due to the improved text-based parser, but the incremental benefit of prosody here is similar to that in these prior studies. (In prior work using acoustic feature directly (Gregory et al., 2004), prosody actually degraded performance.) Our analyses of the impact of prosody also extends prior work.

Prosody is also known to provide useful cues to sentence boundaries (Liu et al., 2006), and automatic sentence segmentation performance has been shown to have a significant impact on parsing performance (Kahn and Ostendorf, 2012). In our study, sentence boundaries are given so as to focus on the role of prosody in resolving sentence-internal parse ambiguity, for which prior work had

obtained smaller gains. Studies have also shown that parsing lattices or confusion networks can improve ASR performance (Kahn and Ostendorf, 2012; Yoshikawa et al., 2016). Our analysis of performance degradation for the system with prosody when the gold transcript and associated parse are in error suggests that prosody may have benefits for parsers operating on alternative ASR hypotheses.

The results we compare to in Section 4 are relatively old. More recent parsing results on spontaneous speech involve dependency parsers using only text (Rasooli and Tetreault, 2013; Honnibal and Johnson, 2014; Yoshikawa et al., 2016), with the exception of a study on unsupervised dependency parsing (Pate and Goldwater, 2013). With the recent success of transition-based neural approaches in dependency parsing, researchers have adapted transition-based ideas to constituent parsing (Zhu et al., 2013; Watanabe and Sumita, 2015; Dyer et al., 2016). These approaches have not yet been used with speech, to our knowledge, but we expect it to be straightforward to extend our prosody integration framework to these systems, both for dependency and constituency parsing.

7 Conclusion

We have presented a framework for directly integrating acoustic-prosodic features with text in a neural encoder-decoder parser that does not require hand-annotated prosodic structure. On conversational sentences, we obtained strong results when including word-level acoustic-prosodic features over using only transcriptions. The acoustic-prosodic features provide the largest gains when sentences are disfluent or long, and analysis of error types shows that these features are especially helpful in repairing attachment errors. In cases where prosodic features hurt performance, we observe a statistically significant negative effect caused by imperfect human transcriptions that make the "ground truth" parse tree and the acoustic signal inconsistent, which suggests that there is more to be gained from prosody than observed in prior studies. We thus plan to investigate aligning the Treebank and MS-State versions of Switchboard for future work.

Here, we assumed known sentence boundaries and hand transcripts, leaving open the question of whether increased benefits from prosody can be gained by incorporating sentence segmentation in parsing and/or in parsing ASR lattices. Most prior work using prosody in parsing has been on constituent parsing, since prosodic cues tend to align with constituent boundaries. However, it remains an open question as to whether dependency, constituency or other parsing frameworks are better suited to leveraging prosody. Our study builds on a parser that uses reverse order text processing, since it provides a stronger text-only baseline. However, the prosody modeling component relies only on a 1 second lookahead of the current word (for pause binning), so it could be easily incorporated in an incremental parser.

Acknowledgement

We thank the anonymous reviewers for their helpful feedback. We also thank Pranava Swaroop Madhyastha, Hao Tang, Jon Cai, Hao Cheng, and Navdeep Jaitly for their help with initial discussions and code setup. This research was partially funded by a Google Faculty Research Award to Mohit Bansal, Karen Livescu, and Kevin Gimpel; and NSF grant no. IIS-1617176. The opinions expressed in this work are those of the authors and do not necessarily reflect the views of the funding agency.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, ..., and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.

Sasha Calhoun, Jean Carletta, Jason M. Brenier, Neil Mayo, Dan Jurafsky, Mark Steedman, and David Beaver. 2010. The NXT-format Switchboard Corpus: a rich resource for investigating the syntax, semantics, pragmatics and prosody of dialogue. *Language Resources and Evaluation* 44(4).

Eugene Charniak and Mark Johnson. 2001. Edit Detection and Parsing for Transcribed Speech. In *Proc. NAACL*.

Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, KyungHyun Cho, and Yoshua Bengio. 2015. Attention-Based Models for Speech Recognition. *CoRR* abs/1506.07503.

Neeraj Deshmukh, Andi Gleeson, Joseph Picone, Aravind Ganapathiraju, and Jonathan Hamaker. 1998. Resegmentation of SWITCHBOARD. In *Proc. ICSLP*.

Markus Dreyer and Izhak Shafran. 2007. Exploiting prosody for PCFGs with latent annotations. In *Proc. Interspeech*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent Neural Network Grammars. In *Proc. NAACL*.

Bradley Efron and Robert J. Tibshirani. 1993. *An Introduction to the Bootstrap*. Chapman & Hall/CRC.

Cécile Fourgeron and Patricia A. Keating. 1997. Articulatory strengthening at edges of prosodic domains. *Journal of the Acoustical Society of America* 101(6).

John J. Godfrey and Edward Holliman. 1993. *Switchboard-1 Release 2*. Linguistic Data Consortium.

Michelle L Gregory, Mark Johnson, and Eugene Charniak. 2004. Sentence-Internal Prosody Does not Help Parsing the Way Punctuation Does. In *Proc. NAACL*.

Franćois Grosjean, Lysiane Grosjean, and Harlan Lane. 1979. The patterns of silence: Performance structures in sentence production. *Cognitive Psychology* .

John Hale, Izhak Shafran, Lisa Yung, Bonnie Dorr, Mary Harper, Anna Krasnyanskaya, Matthew Lease, Yang Liu, Brian Roark, Mathew Snover, and Robin Stewart. 2006. PCFGs with Syntactic and Prosodic Indicators of Speech Repairs. In *Proc. COLING-ACL*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9(8).

Matthew Honnibal and Mark Johnson. 2014. Joint Incremental Disfluency Detection and Dependency Parsing. *TACL* .

Mark Johnson and Eugene Charniak. 2004. A TAG-based Noisy Channel Model of Speech Repairs. In *Proc. ACL*.

Jeremy G. Kahn, Matthew Lease, Eugene Charniak, Mark Johnson, and Mari Ostendorf. 2005. Effective Use of Prosody in Parsing Conversational Speech. In *Proc. HLT/EMNLP*.

Jeremy G. Kahn and Mari Ostendorf. 2012. Joint reranking of parsing and word recognition with automatic segmentation. *Computer Speech & Language* .

Jeremy G. Kahn, Mari Ostendorf, and Ciprian Chelba. 2004. Parsing Conversational Speech Using Enhanced Segmentation. In *Proc. NAACL*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980.

Jonathan K. Kummerfeld, David Hall, James R. Curran, and Dan Klein. 2012. Parser Showdown at the Wall Street Corral: An Empirical Investigation of Error Types in Parser Output. In *Proc. EMNLP*.

Y. Liu, E. Shriberg, A. Stolcke, D. Hillard, M. Ostendorf, and M. Harper. 2006. Enriching Speech Recognition with Automatic Detection of Sentence Boundaries and Disfluencies. *IEEE TASLP* 14.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task Sequence to Sequence Learning. In *Proc. ICLR*.

John Pate and Sharon Goldwater. 2013. Unsupervised Dependency Parsing with Acoustic Cues. *TACL* 1.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning Accurate, Compact, and Interpretable Tree Annotation. In *Proc. COLING-ACL*.

Vu Pham, Théodore Bluche, Christopher Kermorvant, and Jérôme Louradour. 2014. Dropout improves Recurrent Neural Networks for Handwriting Recognition. In *Proc. ICFHR*.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. 2011. The Kaldi Speech Recognition Toolkit. In *Proc. ASRU*.

Patti Price, Mari Ostendorf, Stefanie Shattuck-Hufnagel, and Cynthia Fong. 1991. The Use of Prosody in Syntactic Disambiguation. In *Proc. Workshop on Speech and Natural Language*.

Mohammad Sadegh Rasooli and Joel Tetreault. 2013. Joint Parsing and Disfluency Detection in Linear Time. In *Proc. EMNLP*.

Elizabeth Shriberg. 1994. *Preliminaries to a theory of speech disfluencies*. Ph.D. thesis, Department of Psychology, University of California, Berkeley, CA.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a Foreign Language. In *Proc. NIPS*.

Shaolei Wang, Wanxiang Che, Yue Zhang, Meishan Zhang, and Ting Liu. 2017. Transition-based Disfluency Detection using LSTMs. In *Proc. EMNLP*.

Taro Watanabe and Eiichiro Sumita. 2015. Transition-based Neural Constituent Parsing. In *Proc. ACL*.

Colin W. Wightman, Stefanie Shattuck-Hufnagel, Mari Ostendorf, and Patti J. Price. 1992. Segmental durations in the vicinity of prosodic phrase boundaries. *Journal of the Acoustical Society of America* 91(3).

Masashi Yoshikawa, Hiroyuki Shindo, and Yuji Matsumoto. 2016. Joint Transition-based Dependency Parsing and Disfluency Detection for Automatic Speech Recognition Texts. In *Proc. EMNLP*.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent Neural Network Regularization. *CoRR* abs/1409.2329.

Victoria Zayats, Hannaneh Hajishirzi, and Mari Ostendorf. 2016. Disfluency Detection using a Bidirectional LSTM. In *Proc. Interspeech*.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and Accurate Shift-Reduce Constituent Parsing. In *Proc. ACL*.

A Appendix

A.1 Miscellany

Our main model code is available at https://github.com/shtoshni92/speech_parsing. Most of the data preprocessing code is available at https://github.com/trangham283/seq2seq_parser/tree/master/src/data_preps. Part of our data preprocessing pipeline also uses https://github.com/syllog1sm/swbd_tools.

Table 8 shows statistics of our Switchboard dataset. As defined, for example, in (Charniak and Johnson, 2001; Honnibal and Johnson, 2014), the splits are: conversations sw2000 to sw3000 for training, sw4500 to sw4936 for validation (dev), and sw4000 to sw4153 for evaluation (test). In addition, previous work has reserved sw4154 to sw4500 for "future use" (dev2), but we added this set to our training set. That is, all of our models are trained on Switchboard conversations sw2000 to sw3000 as well as sw4154 to sw4500.

Section	# sentences	# words
Train	97,113	729,252
Dev	5,769	50,445
Test	5,901	48,625

Table 8: Data statistics.

Figure 4 illustrates the data preprocessing step. On the decoder end, we also use a post-processing step that merges the original sentence with the decoder output to obtain the standard constituent tree representation. During inference, in rare cases (and virtually none as our models converge), the decoder does not generate a valid parse sequence, due to the mismatch in brackets and/or the mismatch in the number of pre-terminals and terminals, i.e., num(XX) $\neq$ num(tokens). In such cases, we simply add/remove brackets from either end of the parse, or add/remove pre-terminal symbols XX in the middle of the parse to match the number of input tokens.

Figure 5 shows the distribution of pause durations in our training data. Our pause buckets of

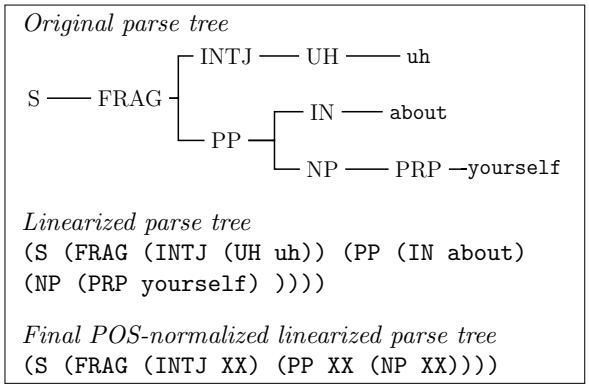


Figure 4: Data preprocessing. Trees are linearized; POS tags (pre-terminals) are normalized as "XX". Also note the annotation standard used for Switchboard data: The root node of the tree is an "S" node although it is not a complete sentence.

$0 < p \leq 0.05$ s, 0.05 s $< p \leq 0.2$ s, $0.2 < p \leq 1$ s, and $p > 1$ s described in the main paper were based on this distribution of pause lengths.

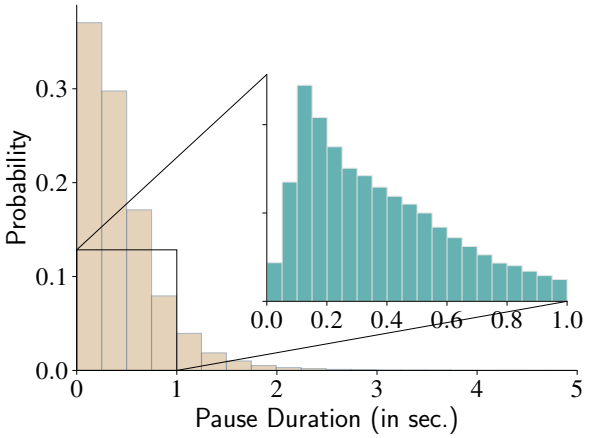


Figure 5: Histogram of inter-word pause durations in our training set. As expected, most of the pauses are less than 1 second. Further binning of pause durations ≤ 1 second reveals that the plot peaks around 0.2 seconds and continuously decays from there on. In some very rare cases, pauses of 5+ seconds occur within a sentence.

Table 9 shows the comprehensive error counts in all error categories defined in the Berkeley Parse Analyzer (Kummerfeld et al., 2012) in both the fluent and disfluent subsets.

A.2 Tree Examples

In figures 6, 7, and 8, we follow node correction notations as in (Kummerfeld et al., 2012). In particular, missing nodes are marked in blue on the gold tree, extra nodes are marked red in the predicted tree, and yellow nodes denote crossing.

Error Type	Fluent Subset			Disfluent Subset		
	text-only	text + p	best model	text-only	text + p	best model
Clause Attach.	126	132	123	631	595	600
Co-ordination	1	2	1	10	10	5
Different label	112	116	124	288	266	300
Modifier Attach.	119	127	112	320	289	325
NP Attach.	92	89	94	332	341	283
NP Internal	71	61	65	231	213	232
PP Attach.	171	152	149	524	471	470
1-Word Phrase	334	342	328	1054	988	1030
UNSET add	86	81	64	353	352	356
UNSET move	85	93	95	466	447	439
UNSET remove	73	70	56	334	324	318
Unary	246	239	236	1088	1100	1074
VP Attach.	36	41	25	167	167	172
XoverX Unary	36	35	34	54	57	54

Table 9: Parse error counts comparison on the fluent (2029 sentences) and disfluent (3689 sentences) subsets of the development set across three parsers.

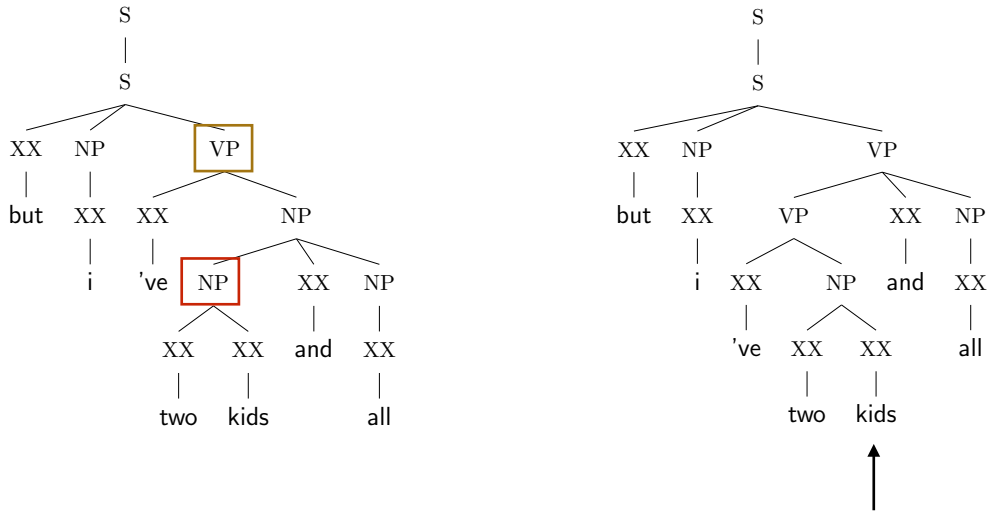


Figure 6: An example sentence from development data – *but i 've two kids and all*. Even though there are no pauses between all words, the word *kids* is lengthened in the audio sample, helping the prosody-enhanced parser (right) to recognize a major syntactic boundary, avoiding the NP Attachment error made by the text-only parser (left).

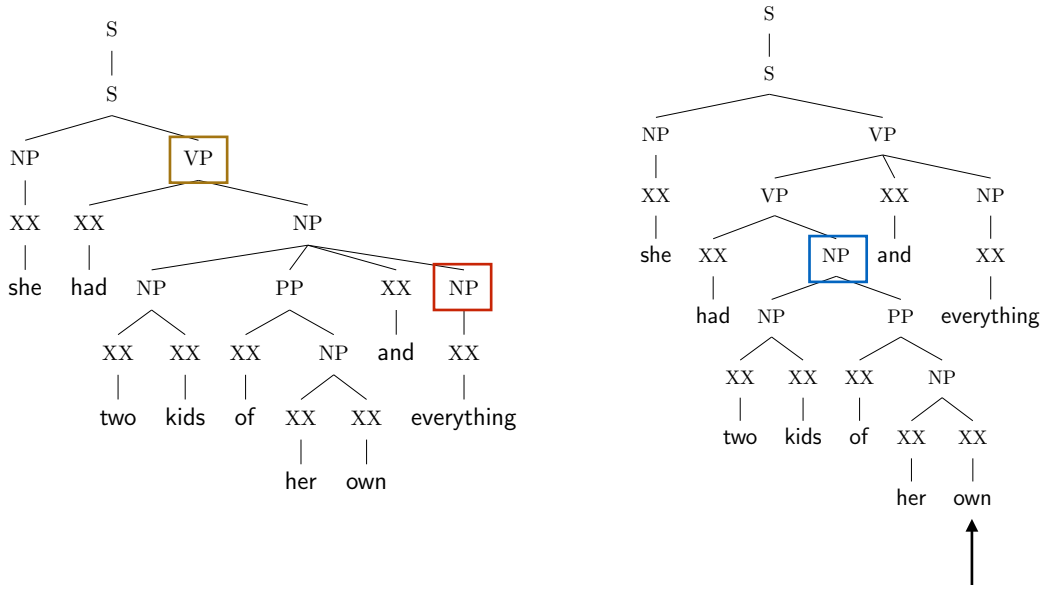


Figure 7: An example sentence from development data – *she had two kids of her own and everything*. There were no pauses between all words in this sentence, the audio sample showed that the word *own* was both lengthened and raised in intonation, giving the prosody-enhanced parser (right) a signal that *own* is on a syntactic boundary. On the other hand, the text-only parser (left) had no such information and made an NP-attachment error. This sentence also illustrates an interesting case where, in isolation, the text-only parse makes sense (i.e. *everything* being an object of *had*). However, in the context of this conversation (the speaker was talking about another person in an informal manner), *and everything* acts more like filler - e.g. "i play the violin *and stuff*"

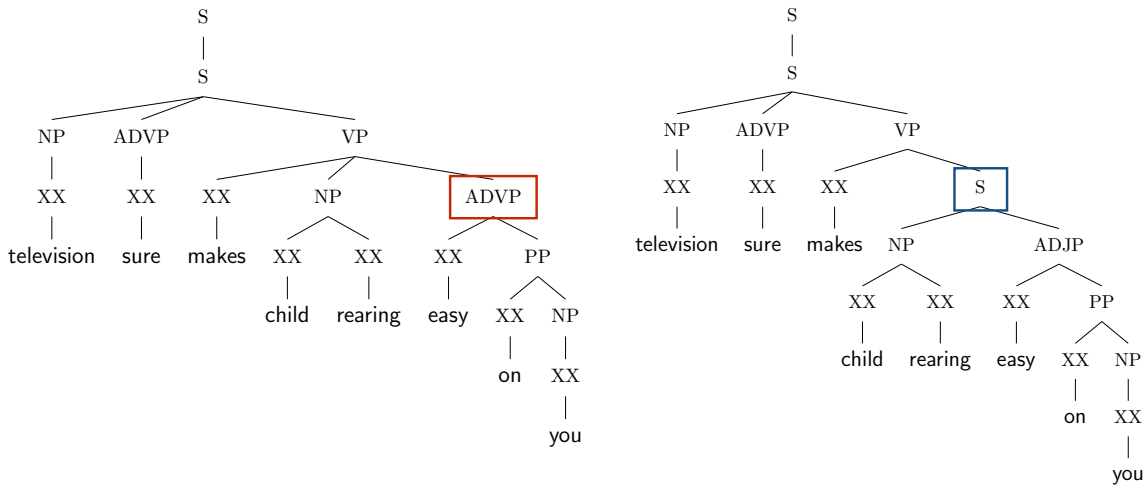


Figure 8: An example sentence from development data – *television sure makes child rearing easy on you*. This is an example where our prosody-enhanced parser (left) did worse than the text-only parser (right), which made no errors. The error type illustrated here is Different Label and Modifier Attachment. In the first iteration, the analyzer identifies a Different Label error (ADVP node), and in the second pass identifies the Modifier Attachment error.

Tied Multitask Learning for Neural Speech Translation

Antonios Anastasopoulos and **David Chiang**
Department of Computer Science and Engineeering
University of Notre Dame
{aanastas,dchiang}@nd.edu

Abstract

We explore multitask models for neural translation of speech, augmenting them in order to reflect two intuitive notions. First, we introduce a model where the second task decoder receives information from the decoder of the first task, since higher-level intermediate representations should provide useful information. Second, we apply regularization that encourages *transitivity* and *invertibility*. We show that the application of these notions on jointly trained models improves performance on the tasks of low-resource speech transcription and translation. It also leads to better performance when using attention information for word discovery over unsegmented input.

1 Introduction

Recent efforts in endangered language documentation focus on collecting spoken language resources, accompanied by spoken translations in a high resource language to make the resource interpretable (Bird et al., 2014a). For example, the BULB project (Adda et al., 2016) used the LIG-Aikuma mobile app (Bird et al., 2014b; Blachon et al., 2016) to collect parallel speech corpora between three Bantu languages and French. Since it's common for speakers of endangered languages to speak one or more additional languages, collection of such a resource is a realistic goal.

Speech can be interpreted either by transcription in the original language or translation to another language. Since the size of the data is extremely small, multitask models that jointly train a model for both tasks can take advantage of both signals. Our contribution lies in improving the sequence-to-sequence multitask learning paradigm, by drawing on two intuitive notions: that higher-level representations are more useful than lower-level representations, and that translation should be both transitive and invertible.

Higher-level intermediate representations, such as transcriptions, should in principle carry information useful for an end task like speech translation. A typical multitask setup (Weiss et al., 2017) shares information at the level of encoded frames, but intuitively, a human translating speech must work from a higher level of representation, at least at the level of phonemes if not syntax or semantics. Thus, we present a novel architecture for *tied* multitask learning with sequence-to-sequence models, in which the decoder of the second task receives information not only from the encoder, but also from the decoder of the first task.

In addition, *transitivity* and *invertibility* are two properties that should hold when mapping between levels of representation or across languages. We demonstrate how these two notions can be implemented through regularization of the attention matrices, and how they lead to further improved performance.

We evaluate our models in three experiment settings: low-resource speech transcription and translation, word discovery on unsegmented input, and high-resource text translation. Our high-resource experiments are performed on English, French, and German. Our low-resource speech experiments cover a wider range of linguistic diversity: Spanish-English, Mboshi-French, and Ainu-English.

In the speech transcription and translation tasks, our proposed model leads to improved performance against all baselines as well as previous multitask architectures. We observe improvements of up to 5% character error rate in the transcription task, and up to 2.8% character-level BLEU in the translation task. However, we didn't observe similar improvements in the text translation experiments. Finally, on the word discovery task, we improve upon previous work by about 3% F-score on both tokens and types.

Proceedings of NAACL-HLT 2018, pages 82–91
New Orleans, Louisiana, June 1 - 6, 2018.

2 Model

Our models are based on a sequence-to-sequence model with attention (Bahdanau et al., 2015). In general, this type of model is composed of three parts: a recurrent encoder, the attention, and a recurrent decoder (see Figure 1a).[1]

The encoder transforms an input sequence of words or feature frames $\mathbf{x}_1, \ldots, \mathbf{x}_N$ into a sequence of *input states* $\mathbf{h}_1, \ldots, \mathbf{h}_N$:

$$\mathbf{h}_n = \text{enc}(\mathbf{h}_{n-1}, \mathbf{x}_n).$$

The attention transforms the input states into a sequence of *context vectors* via a matrix of *attention weights*:

$$\mathbf{c}_m = \sum_n \alpha_{mn} \mathbf{h}_n.$$

Finally, the decoder computes a sequence of *output states* from which a probability distribution over output words can be computed.

$$\begin{aligned} \mathbf{s}_m &= \text{dec}(\mathbf{s}_{m-1}, \mathbf{c}_m, \mathbf{y}_{m-1}) \\ P(\mathbf{y}_m) &= \text{softmax}(\mathbf{s}_m). \end{aligned}$$

In a standard encoder-decoder *multitask* model (Figure 1b) (Dong et al., 2015; Weiss et al., 2017), we jointly model two output sequences using a shared encoder, but separate attentions and decoders:

$$\begin{aligned} \mathbf{c}^1_m &= \sum_n \alpha^1_{mn} \mathbf{h}_n \\ \mathbf{s}^1_m &= \text{dec}^1(\mathbf{s}^1_{m-1}, \mathbf{c}^1_m, \mathbf{y}^1_{m-1}) \\ P(\mathbf{y}^1_m) &= \text{softmax}(\mathbf{s}^1_m) \end{aligned}$$

and

$$\begin{aligned} \mathbf{c}^2_m &= \sum_n \alpha^2_{mn} \mathbf{h}_n \\ \mathbf{s}^2_m &= \text{dec}^2(\mathbf{s}^2_{m-1}, \mathbf{c}^2_m, \mathbf{y}^2_{m-1}) \\ P(\mathbf{y}^2_m) &= \text{softmax}(\mathbf{s}^2_m). \end{aligned}$$

We can also arrange the decoders in a *cascade* (Figure 1c), in which the second decoder attends only to the output states of the first decoder:

$$\begin{aligned} \mathbf{c}^2_m &= \sum_{m'} \alpha^{12}_{mm'} \mathbf{s}^1_{m'} \\ \mathbf{s}^2_m &= \text{dec}^2(\mathbf{s}^2_{m-1}, \mathbf{c}^2_m, \mathbf{y}^2_{m-1}) \\ P(\mathbf{y}^2_m) &= \text{softmax}(\mathbf{s}^2_m). \end{aligned}$$

[1]For simplicity, we have assumed only a single layer for both the encoder and decoder. It is possible to use multiple stacked RNNs; typically, the output of the encoder and decoder ($\mathbf{c}_m$ and $P(\mathbf{y}_m)$, respectively) would be computed from the top layer only.

Tu et al. (2017) use exactly this architecture to train on bitext by setting the second output sequence to be equal to the input sequence ($\mathbf{y}^2_i = \mathbf{x}_i$).

In our proposed *triangle* model (Figure 1d), the first decoder is as above, but the second decoder has two attentions, one for the input states of the encoder and one for the output states of the first decoder:

$$\begin{aligned} \mathbf{c}^2_m &= \left[\textstyle\sum_{m'} \alpha^{12}_{mm'} \mathbf{s}^1_{m'} \quad \sum_n \alpha^2_{mn} \mathbf{h}_n\right] \\ \mathbf{s}^2_m &= \text{dec}^2(\mathbf{s}^2_{m-1}, \mathbf{c}^2_m, \mathbf{y}^2_{m-1}) \\ P(\mathbf{y}^2_m) &= \text{softmax}(\mathbf{s}^2_m). \end{aligned}$$

Note that the context vectors resulting from the two attentions are concatenated, not added.

3 Learning and Inference

For compactness, we will write $\mathbf{X}$ for the matrix whose rows are the $\mathbf{x}_n$, and similarly $\mathbf{H}$, $\mathbf{C}$, and so on. We also write $\mathbf{A}$ for the matrix of attention weights: $[\mathbf{A}]_{ij} = \alpha_{ij}$.

Let θ be the parameters of our model, which we train on sentence triples $(\mathbf{X}, \mathbf{Y}^1, \mathbf{Y}^2)$.

3.1 Maximum likelihood estimation

Define the score of a sentence triple to be a log-linear interpolation of the two decoders' probabilities:

$$\begin{aligned} \text{score}(\mathbf{Y}^1, \mathbf{Y}^2 \mid \mathbf{X}; \theta) = \lambda \log P(\mathbf{Y}^1 \mid \mathbf{X}; \theta) + \\ (1 - \lambda) \log P(\mathbf{Y}^2 \mid \mathbf{X}, \mathbf{S}^1; \theta) \end{aligned}$$

where λ is a parameter that controls the importance of each sub-task. In all our experiments, we set λ to 0.5. We then train the model to maximize

$$\mathcal{L}(\theta) = \sum \text{score}(\mathbf{Y}^1, \mathbf{Y}^2 \mid \mathbf{X}; \theta),$$

where the summation is over all sentence triples in the training data.

3.2 Regularization

We can optionally add a regularization term to the objective function, in order to encourage our attention mechanisms to conform to two intuitive principles of machine translation: *transitivity* and *invertibility*.

***Transitivity* attention regularizer** To a first approximation, the translation relation should be transitive (Wang et al., 2006; Levinboim and Chiang, 2015): If source word $\mathbf{x}_i$ aligns to target word

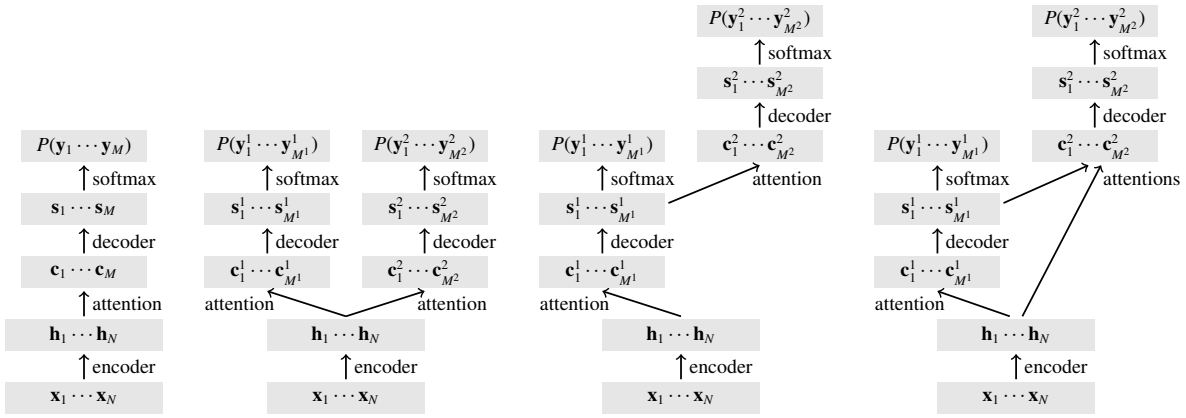


Figure 1: Variations on the standard attentional model. In the standard *single-task* model, the decoder attends to the encoder's states. In a typical *multitask* setup, two decoders attend to the encoder's states. In the *cascade* (Tu et al., 2017), the second decoder attends to the first decoder's states. In our proposed *triangle* model, the second decoder attends to both the encoder's states and the first decoder's states. Note that for clarity's sake there are dependencies not shown.

$\mathbf{y}^1_j$ and $\mathbf{y}^1_j$ aligns to target word $\mathbf{y}^2_k$, then $\mathbf{x}_i$ should also probably align to $\mathbf{y}^2_k$. To encourage the model to preserve this relationship, we add the following *transitivity* regularizer to the loss function of the *triangle* models with a small weight $\lambda_{\text{trans}} = 0.2$:

$$\mathcal{L}_{\text{trans}} = \text{score}(\mathbf{Y}^1, \mathbf{Y}^2) - \lambda_{\text{trans}} \left\|\mathbf{A}^{12}\mathbf{A}^1 - \mathbf{A}^2\right\|_2^2.$$

***Invertibility* attention regularizer** The translation relation also ought to be roughly invertible (Levinboim et al., 2015): if, in the *reconstruction* version of the *cascade* model, source word $\mathbf{x}_i$ aligns to target word $\mathbf{y}^1_j$, then it stands to reason that $\mathbf{y}_j$ is likely to align to $\mathbf{x}_i$. So, whereas Tu et al. (2017) let the attentions of the translator and the reconstructor be unrelated, we try adding the following *invertibility* regularizer to encourage the attentions to each be the inverse of the other, again with a weight $\lambda_{\text{inv}} = 0.2$:

$$\mathcal{L}_{\text{inv}} = \text{score}(\mathbf{Y}^1, \mathbf{Y}^2) - \lambda_{\text{inv}} \left\|\mathbf{A}^{1}\mathbf{A}^{12} - \mathbf{I}\right\|_2^2.$$

3.3 Decoding

Since we have two decoders, we now need to employ a two-phase beam search, following Tu et al. (2017):

1. The *first decoder* produces, through standard beam search, a set of triples each consisting of a candidate transcription $\hat{\mathbf{Y}}^1$, a score $P(\hat{\mathbf{Y}}^1)$, and a hidden state sequence $\hat{\mathbf{S}}$.

2. For each transcription candidate from the *first decoder*, the *second decoder* now produces through beam search a set of candidate translations $\hat{\mathbf{Y}}^2$, each with a score $P(\hat{\mathbf{Y}}^2)$.

3. We then output the combination that yields the highest total score$(\mathbf{Y}^1, \mathbf{Y}^2)$.

Corpus	Speakers	Segments	Hours
Ainu-English	1	2,668	2.5
Mboshi-French	3	5,131	4.4
Spanish-English	240	17,394	20

Table 1: Statistics on our speech datasets.

3.4 Implementation

All our models are implemented in DyNet (Neubig et al., 2017).[2] We use a dropout of 0.2, and train using Adam with initial learning rate of 0.0002 for a maximum of 500 epochs. For testing, we select the model with the best performance on dev. At inference time, we use a beam size of 4 for each decoder (due to GPU memory constraints), and the beam scores include length normalization (Wu et al., 2016) with a weight of 0.8, which Nguyen and Chiang (2017) found to work well for low-resource NMT.

4 Speech Transcription and Translation

We focus on speech transcription and translation of endangered languages, using three different cor-

[2]Our code is available at: https://bitbucket.org/antonis/dynet-multitask-models.

	Model		Search		Mboshi	French	Ainu	English	Spanish	English
	ASR	MT	ASR	MT	CER	BLEU	CER	BLEU	CER	BLEU
(1)	auto	text	1-best	1-best	42.3	21.4	44.0	16.4	63.2	24.2
(2)	gold	text	—	1-best	0.0	31.2	0.0	19.3	0.0	51.3
(3)	single-task		1-best		—	20.8	—	12.0	—	21.6
(4)	multitask		4-best	1-best	36.9	21.0	40.1	18.3	**57.4**	26.0
(5)	cascade		4-best	1-best	39.7	24.3	42.1	19.8	58.1	26.8
(6)	triangle		4-best	1-best	32.3	24.1	39.9	19.2	58.9	**28.6**
(7)	triangle+$\mathcal{L}_{\text{trans}}$		4-best	1-best	33.0	**24.7**	43.3	**20.2**	59.3	**28.6**
(8)	triangle		1-best	1-best	**31.8**	19.7	**38.9**	**19.8**	58.4	**28.8**
(9)	triangle+$\mathcal{L}_{\text{trans}}$		1-best	1-best	32.1	20.9	43.0	**20.3**	59.1	**28.5**

Table 2: The multitask models outperform the baseline single-task model and the pivot approach (auto/text) on all language pairs tested. The *triangle* model also outperforms the simple multitask models on both tasks in almost all cases. The best results for each dataset and task are highlighted.

pora on three different language directions: Spanish (es) to English (en), Ainu (ai) to English, and Mboshi (mb) to French (fr).

4.1 Data

Spanish is, of course, not an endangered language, but the availability of the CALLHOME Spanish Speech dataset (LDC2014T23) with English translations (Post et al., 2013) makes it a convenient language to work with, as has been done in almost all previous work in this area. It consists of telephone conversations between relatives (about 20 total hours of audio) with more than 240 speakers. We use the original train-dev-test split, with the training set comprised of 80 conversations and dev and test of 20 conversations each.

Hokkaido Ainu is the sole surviving member of the Ainu language family and is generally considered a language isolate. As of 2007, only ten native speakers were alive. The Glossed Audio Corpus of Ainu Folklore provides 10 narratives with audio (about 2.5 hours of audio) and translations in Japanese and English.[3] Since there does not exist a standard train-dev-test split, we employ a cross validation scheme for evaluation purposes. In each fold, one of the 10 narratives becomes the test set, with the previous one (mod 10) becoming the dev set, and the remaining 8 narratives becoming the training set. The models for each of the 10 folds are trained and tested separately. On average, for each fold, we train on about 2000 utterances; the dev and test sets consist of about 270 utterances. We report results on the concatenation of all folds. The Ainu text is split into characters, except for the equals (=) and underscore (_) characters, which are used as phonological or structural markers and are thus merged with the following character.[4]

Mboshi (Bantu C25 in the Guthrie classification) is a language spoken in Congo-Brazzaville, without standard orthography. We use a corpus (Godard et al., 2017) of 5517 parallel utterances (about 4.4 hours of audio) collected from three native speakers. The corpus provides non-standard grapheme transcriptions (close to the language phonology) produced by linguists, as well as French translations. We sampled 100 segments from the training set to be our dev set, and used the original dev set (514 sentences) as our test set.

4.2 Implementation

We employ a 3-layer speech encoding scheme similar to that of Duong et al. (2016). The first bidirectional layer receives the audio sequence in the form of 39-dimensional Perceptual Linear Predictive (PLP) features (Hermansky, 1990) computed over overlapping 25ms-wide windows every 10ms. The second and third layers consist of LSTMs with hidden state sizes of 128 and 512 respectively. Each layer encodes every second output of the previous layer. Thus, the sequence is downsampled by a factor of 4, decreasing the computation load for the attention mechanism and the decoders. In the speech experiments, the decoders

[3] http://ainucorpus.ninjal.ac.jp/corpus/en/

[4] The data preprocessing scripts are released with the rest of our code.

output the sequences at the grapheme level, so the output embedding size is set to 64.

We found that this simpler speech encoder works well for our extremely small datasets. Applying our models to larger datasets with many more speakers would most likely require a more sophisticated speech encoder, such as the one used by Weiss et al. (2017).

4.3 Results

In Table 2, we present results on three small datasets that demonstrate the efficacy of our models. We compare our proposed models against three baselines and one "skyline." The first baseline is a traditional pivot approach (line 1), where the ASR output, a sequence of characters, is the input to a character-based NMT system (trained on gold transcriptions). The "skyline" model (line 2) is the same NMT system, but tested on gold transcriptions instead of ASR output. The second baseline is translation directly from source speech to target text (line 3). The last baseline is the standard *multitask* model (line 4), which is similar to the model of Weiss et al. (2017).

The *cascade* model (line 5) outperforms the baselines on the translation task, while only falling behind the *multitask* model in the transcription task. On all three datasets, the *triangle* model (lines 6, 7) outperforms all baselines, including the standard *multitask* model. On Ainu-English, we even obtain translations that are comparable to the "skyline" model, which is tested on gold Ainu transcriptions.

Comparing the performance of all models across the three datasets, there are two notable trends that verify common intuitions regarding the speech transcription and translation tasks. First, an increase in the number of speakers hurts the performance of the speech transcription tasks. The character error rates for Ainu are smaller than the CER in Mboshi, which in turn are smaller than the CER in CALLHOME. Second, the character-level BLEU scores increase as the amount of training data increases, with our smallest dataset (Ainu) having the lowest BLEU scores, and the largest dataset (CALLHOME) having the highest BLEU scores. This is expected, as more training data means that the translation decoder learns a more informed character-level language model for the target language.

Note that Weiss et al. (2017) report much higher BLEU scores on CALLHOME: our model underperforms theirs by almost 9 *word-level* BLEU points. However, their model has significantly more parameters and is trained on 10 times more data than ours. Such an amount of data would never be available in our endangered languages scenario. When calculated on the word-level, all our models' BLEU scores are between 3 and 7 points for the extremely low resource datasets (Mboshi-French and Ainu-English), and between 7 and 10 for CALLHOME. Clearly, the size of the training data in our experiments is not enough for producing high quality speech translations, but we plan to investigate the performance of our proposed models on larger datasets as part of our future work.

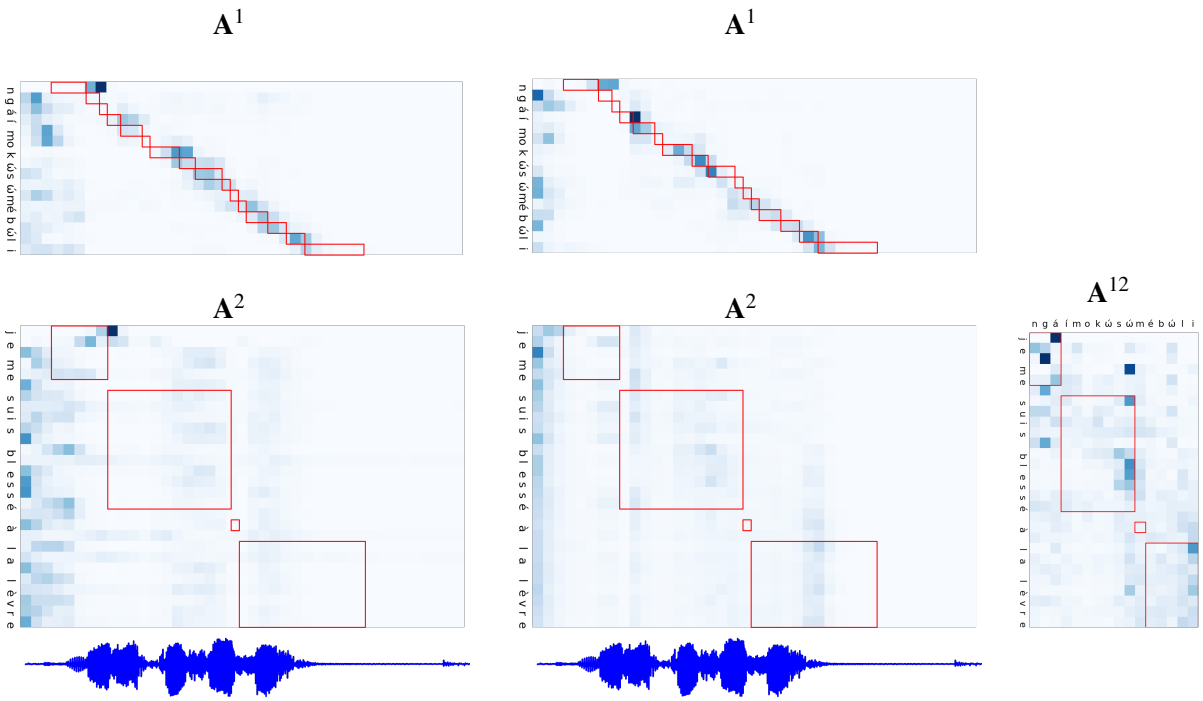
To evaluate the effect of using the combined score from both decoders at decoding time, we evaluated the *triangle* models using only the 1-best output from the speech model (lines 8, 9). One would expect that this would favor speech at the expense of translation. In transcription accuracy, we indeed observed improvements across the board. In translation accuracy, we observed a surprisingly large drop on Mboshi-French, but surprisingly little effect on the other language pairs – in fact, BLEU scores tended to go up slightly, but not significantly.

Finally, Figure 2 visualizes the attention matrices for one utterance from the baseline multitask model and our proposed *triangle* model. It is clear that our intuition was correct: the translation decoder receives most of its context from the transcription decoder, as indicated by the higher attention weights of $\mathbf{A}^{12}$. Ideally, the area under the red squares (gold alignments) would account for 100% of the attention mass of $\mathbf{A}^{12}$. In our triangle model, the total mass under the red squares is 34%, whereas the multitask model's correct attentions amount to only 21% of the attention mass.

5 Word Discovery

Although the above results show that our model gives large performance improvements, in absolute terms, its performance on such low-resource tasks leaves a lot of room for future improvement. A possible more realistic application of our methods is word discovery, that is, finding word boundaries in unsegmented phonetic transcriptions.

After training an attentional encoder-decoder model between Mboshi unsegmented phonetic se-



(a) multitask (b) triangle + transitivity

Figure 2: Attentions in an Mboshi-French sentence, extracted from two of our models. The red squares denote gold alignments. The second decoder of the *triangle* model receives most of its context from the first decoder through $\mathbf{A}^{12}$ instead of the source. The $\mathbf{A}^2$ matrix of the *triangle* model is more informed (34% correct attention mass) than the *multitask* one (21% correct), due to the *transitivity* regularizer.

quences and French word sequences, the attention weights can be thought of as soft alignments, which allow us to project the French word boundaries onto Mboshi. Although we could in principle perform word discovery directly on speech, we leave this for future work, and only explore single-task and reconstruction models.

5.1 Data

We use the same Mboshi-French corpus as in Section 4, but with the original training set of 4617 utterances and the dev set of 514 utterances. Our parallel data consist of the unsegmented phonetic Mboshi transcriptions, along with the word-level French translations.

5.2 Implementation

We first replicate the model of Boito et al. (2017), with a single-layer bidirectional encoder and single layer decoder, using an embedding and hidden size of 12 for the base model, and an embedding and hidden state size of 64 for the reverse model. In our own models, we set the embedding size to 32 for Mboshi characters, 64 for French words, and the hidden state size to 64. We smooth the attention weights $\mathbf{A}$ using the method of Duong et al. (2016) with a temperature $T = 10$ for the softmax computation of the attention mechanism.

Following Boito et al. (2017), we train models both on the *base* Mboshi-to-French direction, as well as the *reverse* (French-to-Mboshi) direction, with and without this smoothing operation. We further smooth the computed soft alignments of all models so that $a_{mn} = (a_{mn-1} + a_{mn} + a_{mn+1})/3$ as a post-processing step. From the *single-task* models we extract the $\mathbf{A}^1$ attention matrices. We also train *reconstruction* models on both directions, with and without the *invertibility* regularizer, extracting both $\mathbf{A}^1$ and $\mathbf{A}^{12}$ matrices. The two matrices are then combined so that $\mathbf{A} = \mathbf{A}^1 + (\mathbf{A}^{12})^T$.

5.3 Results

Evaluation is done both at the token and the type level, by computing precision, recall, and F-score over the discovered segmentation, with the best results shown in Table 3. We reimplemented the base (Mboshi-French) and reverse (French-Mboshi) models from Boito et al. (2017), and the performance of the base model was comparable to the one reported. However, we were unable to

Model (with smoothing)		Tokens			Types		
		Precision	Recall	F-score	Precision	Recall	F-score
Boito et al. 2017 (reported)	*base*	*5.85*	*6.82*	*6.30*	*6.76*	*15.00*	*9.32*
	reverse	*21.44*	*16.49*	*18.64*	*27.23*	*15.02*	*19.36*
Boito et al. 2017 (reimplementation)	base	6.87	6.33	6.59	6.17	13.02	8.37
	reverse	7.58	8.16	7.86	9.22	11.97	10.42
our single-task	base	7.99	7.57	7.78	7.59	**16.41**	10.38
	reverse	**11.31**	**11.82**	**11.56**	9.29	14.75	11.40
reconstruction + $0.2\mathcal{L}_{\text{inv}}$		8.93	9.78	9.33	8.66	15.48	11.02
reconstruction + $0.5\mathcal{L}_{\text{inv}}$		7.42	10.00	8.52	**10.46**	16.36	**12.76**

Table 3: The reconstruction model with the *invertibility* regularizer produces more informed attentions that result in better word discovery for Mboshi with an Mboshi-French model. Scores reported by previous work are in *italics* and best scores from our experiments are in **bold**.

reproduce the significant gains that were reported when using the reverse model (*italicized* in Table 3). Also, our version of both the base and reverse singletask models performed better than our reimplementation of the baseline.

Furthermore, we found that we were able to obtain even better performance at the type level by combining the attention matrices of a reconstruction model trained with the *invertibility* regularizer. Boito et al. (2017) reported that combining the attention matrices of a base and a reverse model significantly reduced performance, but they trained the two models separately. In contrast, we obtain the base ($\mathbf{A}^1$) and the reverse attention matrices ($\mathbf{A}^{12}$) from a model that trains them jointly, while also tying them together through the *invertibility* regularizer. Using the regularizer is key to the improvements; in fact, we did not observe any improvements when we trained the reconstruction models without the regularizer.

6 Negative Results: High-Resource Text Translation

6.1 Data

For evaluating our models on text translation, we use the Europarl corpus which provides parallel sentences across several European languages. We extracted 1,450,890 three-way parallel sentences on English, French, and German. The concatenation of the newstest 2011–2013 sets (8,017 sentences) is our dev set, and our test set is the concatenation of the newstest 2014 and 2015 sets (6,003 sentences). We test all architectures on the six possible translation directions between English (en), French (fr) and German (de). All the sequences are represented by subword units with byte-pair encoding (BPE) (Sennrich et al., 2016) trained on each language with 32000 operations.

6.2 Experimental Setup

On all experiments, the encoder and the decoder(s) have 2 layers of LSTM units with hidden state size and attention size of 1024, and embedding size of 1024. For this high resource scenario, we only train for a maximum of 40 epochs.

6.3 Results

The accuracy of all the models on all six language pair directions is shown in Table 4. In all cases, the best models are the baseline single-task or simple multitask models. There are some instances, such as English-German, where the *reconstruction* or the *triangle* models are not statistically significantly different from the best model. The reason for this, we believe, is that in the case of text translation between so linguistically close languages, the lower level representations (the output of the encoder) provide as much information as the higher level ones, without the search errors that are introduced during inference.

A notable outcome of this experiment is that we do not observe the significant improvements with the *reconstruction* models that Tu et al. (2017) observed. A few possible differences between our experiment and theirs are: our models are BPE-based, theirs are word-based; we use Adam for optimization, they use Adadelta; our model has slightly fewer parameters than theirs; we test on less typologically different language pairs than

Model	$s \rightarrow t$					
	en→fr	en→de	fr→en	fr→de	de→en	de→fr
singletask	**20.92**	**12.69**	**20.96**	**11.24**	**16.10**	**15.29**
multitask $s \rightarrow x, t$	20.54	**12.79**	20.01	**11.18**	**16.31**	**15.07**
cascade $s \rightarrow x \rightarrow t$	15.93	11.31	16.58	7.60	13.46	13.24
cascade $s \rightarrow t \rightarrow x$	20.34	12.27	19.17	**11.09**	15.24	14.78
reconstruction	20.19	**12.44**	20.63	10.88	15.66	13.44
reconstruction $+\mathcal{L}_{\text{inv}}$	**20.72**	**12.64**	20.11	10.46	15.43	12.64
triangle $s \xrightarrow{\nearrow x \searrow} t$	20.39	**12.70**	17.93	10.17	14.94	14.07
triangle $s \xrightarrow{\nearrow x \searrow} t + \mathcal{L}_{\text{trans}}$	20.52	**12.64**	18.34	10.42	15.22	14.37
triangle $s \xrightarrow{\nearrow t \searrow} x$	20.38	**12.40**	18.50	10.22	15.62	14.77
triangle $s \xrightarrow{\nearrow t \searrow} x + \mathcal{L}_{\text{trans}}$	20.64	**12.42**	19.20	10.21	**15.87**	14.89

Table 4: BLEU scores for each model and translation direction $s \rightarrow t$. In the multitask, cascade, and triangle models, x stands for the third language, other than s and t. In each column, the best results are highlighted. The non-highlighted results are statistically significantly worse than the single-task baseline.

English-Chinese.

However, we also observe that in most cases our proposed regularizers lead to increased performance. The *invertibility* regularizer aids the *reconstruction* models in achiev slightly higher BLEU scores in 3 out of the 6 cases. The *transitivity* regularizer is even more effective: in 9 out the 12 source-target language combinations, the *triangle* models achieve higher performance when trained using the regularizer. Some of them are statistical significant improvements, as in the case of French to English where English is the intermediate target language and German is the final target.

7 Related Work

The speech translation problem has been traditionally approached by using the output of an ASR system as input to a MT system. For example, Ney (1999) and Matusov et al. (2005) use ASR output lattices as input to translation models, integrating speech recognition uncertainty into the translation model. Recent work has focused more on modelling speech translation without explicit access to transcriptions. Duong et al. (2016) introduced a sequence-to-sequence model for speech translation without transcriptions but only evaluated on alignment, while Anastasopoulos et al. (2016) presented an unsupervised alignment method for speech-to-translation alignment. Bansal et al. (2017) used an unsupervised term discovery system (Jansen et al., 2010) to cluster recurring audio segments into pseudowords and translate speech using a bag-of-words model. Bérard et al. (2016) translated synthesized speech data using a model similar to the Listen Attend and Spell model (Chan et al., 2016). A larger-scale study (Bérard et al., 2018) used an end-to-end neural system system for translating audio books between French and English. On a different line of work, Boito et al. (2017) used the attentions of a sequence-to-sequence model for word discovery.

Multitask learning (Caruana, 1998) has found extensive use across several machine learning and NLP fields. For example, Luong et al. (2016) and Eriguchi et al. (2017) jointly learn to parse and translate; Kim et al. (2017) combine CTC- and attention-based models using multitask models for speech transcription; Dong et al. (2015) use multitask learning for multiple language translation. Toshniwal et al. (2017) apply multitask learning to neural speech recognition in a less traditional fashion: the lower-level outputs of the speech encoder are used for fine-grained auxiliary tasks such as predicting HMM states or phonemes, while the final output of the encoder is passed to a character-level decoder.

Our work is most similar to the work of Weiss et al. (2017). They used sequence-to-sequence models to transcribe Spanish speech and translate it in English, by jointly training the two tasks in a multitask scenario where the decoders share the encoder. In contrast to our work, they use a large corpus for training the model on roughly 163 hours of data, using the Spanish Fisher and CALL-

HOME conversational speech corpora. The parameter number of their model is significantly larger than ours, as they use 8 encoder layers, and 4 layers for each decoder. This allows their model to adequately learn from such a large amount of data and deal well with speaker variation. However, training such a large model on endangered language datasets would be infeasible.

Our model also bears similarities to the architecture of the model proposed by Tu et al. (2017). They report significant gains in Chinese-English translation by adding an additional *reconstruction* decoder that attends on the last states of the *translation* decoder, mainly inspired by auto-encoders.

8 Conclusion

We presented a novel architecture for multitask learning that provides the second task with higher-level representations produced from the first task decoder. Our model outperforms both the single-task models as well as traditional multitask architectures. Evaluating on extremely low-resource settings, our model improves on both speech transcription and translation. By augmenting our models with regularizers that implement transitivity and invertibility, we obtain further improvements on all low-resource tasks.

These results will hopefully lead to new tools for endangered language documentation. Projects like BULB aim to collect about 100 hours of audio with translations, but it may be impractical to transcribe this much audio for many languages. For future work, we aim to extend these methods to settings where we don't necessarily have sentence triples, but where some audio is only transcribed and some audio is only translated.

Acknowledgements This work was generously supported by NSF Award 1464553. We are grateful to the anonymous reviewers for their useful comments.

References

Gilles Adda, Sebastian Stüker, Martine Adda-Decker, Odette Ambouroue, Laurent Besacier, David Blachon, Hélène Bonneau-Maynard, Pierre Godard, Fatima Hamlaoui, Dmitry Idiatov, et al. 2016. Breaking the unwritten language barrier: The BULB project. *Procedia Computer Science*, 81:8–14.

Antonios Anastasopoulos, David Chiang, and Long Duong. 2016. An unsupervised probability model for speech-to-translation alignment of low-resource languages. In *Proc. EMNLP*.

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

Sameer Bansal, Herman Kamper, Adam Lopez, and Sharon Goldwater. 2017. Towards speech-to-text translation without speech recognition. In *Proc. EACL*.

Alexandre Bérard, Laurent Besacier, Ali Can Kocabiyikoglu, and Olivier Pietquin. 2018. End-to-end automatic speech translation of audiobooks. arXiv:1802.04200.

Alexandre Bérard, Olivier Pietquin, Christophe Servan, and Laurent Besacier. 2016. Listen and translate: A proof of concept for end-to-end speech-to-text translation. In *Proc. NIPS Workshop on End-to-end Learning for Speech and Audio Processing*.

Steven Bird, Lauren Gawne, Katie Gelbart, and Isaac McAlister. 2014a. Collecting bilingual audio in remote indigenous communities. In *Proc. COLING*.

Steven Bird, Florian R. Hanke, Oliver Adams, and Haejoong Lee. 2014b. Aikuma: A mobile app for collaborative language documentation. In *Proc. of the 2014 Workshop on the Use of Computational Methods in the Study of Endangered Languages*.

David Blachon, Elodie Gauthier, Laurent Besacier, Guy-Noël Kouarata, Martine Adda-Decker, and Annie Rialland. 2016. Parallel speech collection for under-resourced language studies using the LIG-Aikuma mobile device app. In *Proc. SLTU (Spoken Language Technologies for Under-Resourced Languages)*, volume 81.

Marcely Zanon Boito, Alexandre Bérard, Aline Villavicencio, and Laurent Besacier. 2017. Unwritten languages demand attention too! word discovery with encoder-decoder models. arXiv:1709.05631.

Rich Caruana. 1998. Multitask learning. In *Learning to learn*, pages 95–133. Springer.

William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. 2016. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *Proc. ICASSP*, pages 4960–4964. IEEE.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proc. ACL-IJCNLP*.

Long Duong, Antonios Anastasopoulos, David Chiang, Steven Bird, and Trevor Cohn. 2016. An attentional model for speech translation without transcription. In *Proc. NAACL HLT*.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *Proc. ACL*.

P. Godard, G. Adda, M. Adda-Decker, J. Benjumea, L. Besacier, J. Cooper-Leavitt, G-N. Kouarata, L. Lamel, H. Maynard, M. Mueller, et al. 2017. A very low resource language speech corpus for computational language documentation experiments. arXiv:1710.03501.

Hynek Hermansky. 1990. Perceptual linear predictive (PLP) analysis of speech. *J. Acoustical Society of America*, 87(4):1738–1752.

Aren Jansen, Kenneth Church, and Hynek Hermansky. 2010. Towards spoken term discovery at scale with zero resources. In *Proc. INTERSPEECH*.

Suyoun Kim, Takaaki Hori, and Shinji Watanabe. 2017. Joint CTC-attention based end-to-end speech recognition using multi-task learning. In *Proc. ICASSP*.

Tomer Levinboim and David Chiang. 2015. Multi-task word alignment triangulation for low-resource languages. In *Proc. NAACL HLT*.

Tomer Levinboim, Ashish Vaswani, and David Chiang. 2015. Model invertibility regularization: Sequence alignment with or without parallel data. In *Proc. NAACL HLT*.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *Proc. ICLR*.

Evgeny Matusov, Stephan Kanthak, and Hermann Ney. 2005. On the integration of speech recognition and statistical machine translation. In *Ninth European Conference on Speech Communication and Technology*.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. DyNet: The dynamic neural network toolkit. arXiv:1701.03980.

Hermann Ney. 1999. Speech translation: Coupling of recognition and translation. In *Proc. ICASSP*, volume 1.

Toan Q. Nguyen and David Chiang. 2017. Transfer learning across low-resource related languages for neural machine translation. In *Proc. IJCNLP*.

Matt Post, Gaurav Kumar, Adam Lopez, Damianos Karakos, Chris Callison-Burch, and Sanjeev Khudanpur. 2013. Improved speech-to-text translation with the Fisher and Callhome Spanish-English speech translation corpus. In *Proc. IWSLT*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proc. ACL*.

Shubham Toshniwal, Hao Tang, Liang Lu, and Karen Livescu. 2017. Multitask learning with low-level auxiliary tasks for encoder-decoder based speech recognition. In *Proc. Interspeech*.

Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. 2017. Neural machine translation with reconstruction. In *Proc. AAAI*.

Haifeng Wang, Hua Wu, and Zhanyi Liu. 2006. Word alignment for languages with scarce resources using bilingual corpora of other language pairs. In *Proc. COLING/ACL*, pages 874–881.

Ron J. Weiss, Jan Chorowski, Navdeep Jaitly, Yonghui Wu, and Zhifeng Chen. 2017. Sequence-to-sequence models can directly transcribe foreign speech. In *Proc. INTERSPEECH*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv:1609.08144.

Please Clap: Modeling Applause in Campaign Speeches

Jon Gillick
School of Information
University of California, Berkeley
jongillick@berkeley.edu

David Bamman
School of Information
University of California, Berkeley
dbamman@berkeley.edu

Abstract

This work examines the rhetorical techniques that speakers employ during political campaigns. We introduce a new corpus of speeches from campaign events in the months leading up to the 2016 U.S. presidential election and develop new models for predicting moments of audience applause. In contrast to existing datasets, we tackle the challenge of working with transcripts that derive from uncorrected closed captioning, using associated audio recordings to automatically extract and align labels for instances of audience applause. In prediction experiments, we find that lexical features carry the most information, but that a variety of features are predictive, including prosody, long-term contextual dependencies, and theoretically motivated features designed to capture rhetorical techniques.

1 Introduction

Every public speech involving a large audience can be seen as a game of coordination (Asch, 1951): at each moment, each individual member of the audience must decide in a split second whether to applaud at what has just been said. Applause is a potentially risky action: if an individual spontaneously claps but no one joins in, they suffer some negative social cost; the game is to judge from their own private information and content of the speech whether the rest of the audience will applaud at the same time they do.

Because of this cost, audiences respond to several interacting factors in a speaker's behavior: a.) the content of the message; b.) their delivery (so that changes in pitch, duration and gaze signal salient moments for which applause may be licensed); and c.) the verbal *design* of the message—those rhetorical strategies that speakers use to signal that applause is welcome (Atkinson, 1984; Heritage and Greatbatch, 1986).

In this work, we attempt to model all three of these dimensions in developing a computational model for applause. While past work has focused on these elements in isolation (Guerini et al., 2015; Liu et al., 2017) or for related problems such as laughter detection (Purandare and Litman, 2006; Chen and Lee, 2017; Bertero and Fung, 2016), we find that developing a holistic model encompassing all three aspects yields the most robust predictor of applause.

We focus on political speeches, and in particular those at campaign rallies, which lend themselves well to analysis of rhetorical strategies for several reasons. First, the speakers at these events prioritize maintaining the crowd's attention (Strangert, 2005). Motivated to drum up excitement and fervor among their supporters that they hope will carry beyond the event and into the voting booth, speakers pull out their strongest rhetorical tactics. Second, campaign speeches usually consist of a series of self-contained messages that can be fully expressed within a few utterances (Heritage and Greatbatch, 1986), yielding a well-defined observation of a complete rhetorical strategy. Lastly, these speeches are delivered by a single speaker to a partisan crowd, and clapping, cheering, and other responses are invited and expected.

We focus in particular in this work on operationalizating the verbal design of the speech; in so doing, one contribution we make is operationalizing the concepts of *tension* and *release*. Writers and performers often communicate with their audience on a fundamental level by building up tension, and then, at the proper time, delivering a satisfying release. These simple but pervasive concepts structure our experience of different modes of communication used throughout everyday life, including music (Madsen and Fredrickson, 1993), literature (Rabkin, 1973) and film (Carroll, 1996).

Tension in music can be built up by harmonic

Proceedings of NAACL-HLT 2018, pages 92–102
New Orleans, Louisiana, June 1 - 6, 2018.

movement away from a tonal center; release then comes with a return to that established tonic (Hindemith, 1937). One form of tension in literature is realized as suspense (Barthes and Duisit, 1975; Vorderer et al., 1996; Algee-Hewitt, 2016), in which a reader's knowledge of events is uncertain (either because those events take place in the narrative future or are withheld from narration), and released when that knowledge is revealed. In film, sudden changes in camera perspective create graphic tension, which is then released as the shot returns to a stable position (Bordwell, 2013). Often, it is the confluence of multiple sources of tension that mark the climax of a narrative (Hume, 2017). We draw on each of these strands of work in operationalizing tension and release as a rhetorical strategy.

In this work, we make the following contributions:

- We collect a new dataset of text and audio from 310 speeches from campaign events leading up to the 2016 U.S presidential election with associated tags for over 19,000 instances of audience applause.
- We introduce new textual and acoustic features inspired by tension and release, combine and compare them with features used in previous work, and deploy those features in a logistic regression model and in an LSTM to predict when applause is likely to occur. Code, data, and trained models are openly available to the public at `https://github.com/jrgillick/Applause/`.

2 Background and Previous Work

2.1 Rhetoric and Response

Heritage and Greatbatch (1986) conduct an extensive analysis of nearly 500 speeches from British political party conferences, manually associating each of over 2000 instances of applause with coded message types (e.g. External Attacks or Statements of Approval), rhetorical devices (e.g. Contrast/Antithesis or Headline-Punchline), and performance factors (e.g. speech stress or body language). They find most of these factors to be positively correlated with applause; one especially striking result is over two thirds of observed instances of applause can be explained through a set of seven rhetorical devices (including contrast, pursuit, position taking, and "the 3-part list"). Though each device is different, a common feature of most of these techniques is that they are not always carried out within a single sentence or utterance; they often depend on the relationship between a series of utterances or phrases. We argue in this work that some of these relationships can be characterized and subsequently operationalized within models as tension and release.

2.2 Predicting Applause

Recent work from Guerini et al. (2015) and Liu et al. (2017) approaches the task of applause prediction by looking at textual features of the individual sentences that immediately precede audience applause. Both follow the methodology proposed by Danescu-Niculescu-Mizil et al. (2012) in constructing a data set for binary classification, which is composed of sentences that generated applause, each paired with a single nearby sentence from the same document that did not lead to applause.

Guerini et al. (2015) examine a set of features designed to capture aspects of euphony, or "the inherent pleasantness of the sounds of words" that might make an utterance memorable or persuasive—such as rhyme, alliteration, homogeneity, and plosives. On the CORPS dataset (Guerini et al., 2013), which consists of the text of several thousand political speeches dating from 1917 to 2011, they define persuasive sentences as those that preceded annotations of either applause or laughter.

Liu et al. (2017), working with a corpus of TED talks, use logistic regression to predict applause from sentences using a combination of features: euphony (again from Guerini et al. (2015)), linguistic style markers derived from membership in LIWC categories, markers of emotional expression derived from membership in the NRC Emotion Lexicon, mentions of names, rhetorical questions (string matching for "?"), expressions of gratitude (matching a handcrafted list of word stems including "thank*" and "grateful*"), and expressions seeking applause (matching the pattern "applau*"). Liu et al. (2017) also report that adding the same features for earlier sentences beyond the final sentence that preceded the applause caused the prediction accuracy to go down. Chen and Lee (2017) and Bertero and Fung (2016) run similar binary classification experiments but pre-

dict laughter as opposed to applause. Bertero and Fung (2016) analyze punchlines from the TV sitcom "The Big Bang Theory" and report 70% accuracy using an LSTM. They touch briefly on the notion of tension and release in humor, as punchlines typically depend on a previous line as a setup in order to be funny.

3 Data

3.1 Corpus Acquisition

In this work, we focus on a new data set of campaign speeches from the 2016 U.S. presidential race, which we obtain from the public domain broadcasts of C-SPAN. We downloaded about 500 speeches from presidential candidates, vice presidential candidates, or former presidents, collecting audio files and transcripts that were tagged in the categories "Campaign 2016" and "Speech" and which took place between 12/01/2015 and 12/01/2016. We then excluded events that took place outside of a traditional campaign speech setting (e.g. town hall events) or events that contained multiple speakers without a speaker identification tied to the transcript, which yielded a final set of 310 speeches from 16 speakers. Because different types of events have different social norms around when and whether applause is appropriate (Atkinson, 1984; Heritage and Greatbatch, 1986), we control for these factors to some degree by restricting our dataset to events in similar settings and within a single year. As a point of comparison, the C-SPAN dataset contains 62 instances of applause per speech on average, whereas the CORPS data (Guerini et al., 2013) contains 13.

3.2 Applause Detection in Audio

Since our C-SPAN data originates in video, we have access to the audio information of a speech event, which we employ both for feature extraction and for automatically identifying when applause occurs. Following Clement and McLaughlin (2016), we train an acoustic model using a set of poetry readings from the PennSound archive to distinguish applause from speech. We used logistic regression on the standard set of MFCC features and found similar results on the PennSound data to the reported classification accuracy of 99.4%. In a manual inspection of 100 applause segments from 5 different speeches in the C-SPAN corpus, our applause detector achieved 92% precision, 90% recall, and 91% F1 score. Due to variation in the nature of applause in a crowd (sometimes we observe examples of isolated clapping and cheering, mixed laughter and applause, or applause interrupting the speaker), some ambiguity is inherent among the labels.

We also measure the applause by first running the speeches through the audio source separation algorithm from Chandna et al. (2017), which was trained to separate voice from music, and then measuring the RMSE loudness of the separated non-vocal track. We found that the separation worked well, qualitatively matching with the results from the applause detection classifier.

3.3 Forced Alignment

To match the identified segments of applause in the audio files with the relevant text from the transcriptions, we ran forced alignment using the Kaldi Toolkit (Povey et al., 2011). Since the C-SPAN transcripts are sourced from uncorrected closed captioning, the text contains a number of misspellings and paraphrases, which we handled by discarding the 12% of words for which forced alignment failed. Though these transcriptions are not as accurate as what we would find in professionally transcribed datasets, previous work has shown that it is possible to achieve good accuracy in downstream tasks even with high error rates in transcription (Peskin et al., 1993; Novotney and Callison-Burch, 2010). Moreover, the caliber of transcripts derived from closed captioning is representative of the data that would be available in real time for practical use at future speech events.

To estimate the accuracy of the closed captions, we manually transcribed selections from 5 speeches in the C-SPAN data totaling about 25 minutes and 2250 words, finding 30.9% WER relative to the reference transcriptions in our sample. Many of the errors are due to omitted words and phrases in the closed captions, which may occur as a result of transcribers' inability to keep up with the pace of fast speeches; in this sample, the closed caption texts contained 17% fewer words than our gold standard transcriptions.

After finding the alignments, we segmented out a list of utterances by defining a minimum period of silence between words. Since many of the transcripts do not have punctuation, we find that dividing the text into utterances yielded qualitatively more coherent units than sentence boundary detec-

Speaker	Number of Speeches	Number of Utterances	Number Applauded	Percentage
Donald Trump	86	27493	7357	0.27
Hilary Clinton	72	12825	3933	0.31
Bernie Sanders	40	10994	3529	0.32
Ted Cruz	23	5873	1041	0.18
Marco Rubio	20	4407	797	0.18
John Kasich	17	4023	319	0.08
Barack Obama	10	3888	920	0.24
Bill Clinton	8	2087	292	0.14
Joe Biden	7	1847	270	0.15
Mike Pence	6	1302	246	0.19
Carly Fiorina	5	1222	129	0.11
Jeb Bush	5	1482	191	0.13
Rand Paul	4	939	134	0.14
Gary Johnson	3	354	56	0.16
Chris Christie	3	1868	42	0.022
Rick Santorum	1	245	17	0.07
Total	**310**	**80849**	**19273**	**0.24**

Table 1: Speakers and applause in C-SPAN corpus

tion. Dividing into utterances is also conducive to building a dataset for binary classification, since every pause by the speaker yields an opportunity for applause. We chose a pause length of 0.7 seconds, but in future work we might be able to improve our models by adapting this threshold to the rate of speech in order to maintain consistent phrase sizes across different speakers. Given this set of utterances, we paired each utterance with a "positive" or "negative" label, determined by whether applause occurred within 1.5 seconds of the end of the utterance. All of these preprocessing choices were made during the corpus preparation phase, prior to any experimental evaluation.

Table 1 provides summary statistics for the number of speakers, speeches, utterances, and acts of applause in our data.

4 Models

In our models, we draw features from previous work on applause or humor prediction and then supplement them with a new set of features inspired by the ideas of tension and release and by the rhetorical strategies of Heritage and Greatbatch (1986).

4.1 Features adapted from existing work

LIWC. Features for membership in 73 LIWC categories proved to be the most effective for applause prediction in TED talks (Liu et al., 2017).

Euphony. We adopt the 4 features for "euphony" defined by Guerini et al. (2015): Rhyme, Alliteration, Homogeneity, and Plosives.

Lexical. Guerini et al. (2015) find n-grams to be highly predictive of both applause and laughter. We operationalize these features with bigrams, including in our model all bigrams that appear at least 5 times in the corpus.

Embeddings. Bertero and Fung (2016) use sentence embeddings learned from a CNN encoder as input to an LSTM. We adopt this feature for use in our neural models, encoding phrases using the Skip-Thought model of Kiros et al. (2015).

Acoustic. Purandare and Litman (2006) use a set of features intended to capture elements of prosody in a model for humor prediction in television dialogue. These features include the mean, max, min, range, and standard deviation values in an utterance's pitch (F0) and energy (RMS), along with features for internal silence and for tempo. We compute the F0 statistics with Reaper (Talkin, 2015) and the energy statistics with Librosa (McFee et al., 2015).

4.2 New Features

4.2.1 Repetition

Repeated Words. Rhetorical strategies such as "The 3-part List" and "Contrast" rely on repetition to drive home important points. We capture this phenomenon by computing the proportion of words in each utterance that also appear in the immediately preceding phrase.

Longest Common Subsequence. Repeating an entire phrase, especially one with a politically charged topic, serves to build tension through the notion of "theme and variation" as is often realized

in music (Cope, 2005); an example of this phenomenon in our data can be found in the following passage:

> *We will not allow the* party of Lincoln and Reagan to fall into the hands of a con artist. *We will not allow the* next president of the United States to be a socialist like Bernie Sanders. And *we will not allow the* next president of the United States to be someone under FBI investigation like Hillary Clinton.
>
> [Marco Rubio, Mar. 1, 2016]

We calculate this theme and variation by measuring the longest common subsequence between adjacent phrases.

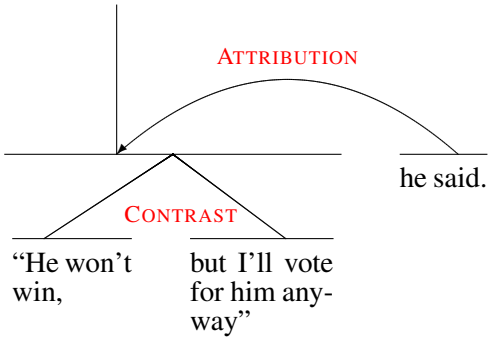
4.2.2 Deltas

Delta features (local approximations to derivatives) are commonly used in speech recognition and audio classification systems (Povey et al., 2011). In a discourse, either highly similar or drastically different neighboring pairs of utterances may indicate dramatic moments. We operationalize these features by explicitly adding a delta measurement for every feature in our model, which captures the difference between every feature at time t and the same feature at time $t-1$. For K-dimensional vector embeddings, we calculate deltas as their cosine distance.

4.2.3 RST

Rhetorical Structure Theory (RST) provides a foundation for describing the ways in which functional components of a text combine to form a coherent whole (Thompson and Mann, 1987). At the core of RST is a categorization system consisting of relations between elementary discourse units (EDUs). Relations between units are typically hierarchical (a nucleus and a satellite), but can also be defined between equally significant units (two nuclei).

A typical RST tree can be seen below, where the sentence *"He won't win, but I'll vote for him anyway", he said* is decomposed into three elementary discourse units (EDUs); those discourse units form the leaves of a tree with intermediate structure between subphrases and labeled edges along each branch.



Some of the rhetorical strategies defined by Heritage and Greatbatch (1986), such as "Contrast," map directly to RST relations, while others do not have a clear one-to-one mapping but are qualitatively similar in their descriptions. While RST has been used with success for classification problems in the past (Ji and Smith, 2017; Bhatia et al., 2015), it has not yet been employed in existing models for applause prediction. In our work, we parse the rhetorical structure of the extracted sequence of phrases using the RST parser of Ji and Eisenstein (2014). From the structure of this RST tree, we extract two classes of features.

RST label. First, we operationalize the rhetorical category for an individual elementary discourse unit. While the span of text within a single EDU is implicated in several rhetorical relations throughout the tree (as *He won't win* bears a CONTRAST relationship with *but I'll vote for him anyway* and is part of the ATTRIBUTION relationship with *he said*), each EDU bears exactly one leaf relationship with the rest of the tree—here, *He won't win* is a nucleus of a CONTRAST relationship, *but I'll vote for him anyway* is also a nucleus of a CONTRAST relationship, and *he said* is the satellite of an ATTRIBUTION relationship.

We featurize a sentence as the set of all such typed relationships that EDUs within it hold; each typed relationship is the conjunction of the label (e.g., CONTRAST, ATTRIBUTION) and directionality (Nucleus, Satellite).

Rhetorical phrase closures. In order to further operationalize the notion of predictability of applause, we measure the number of rhetorical phrases that a given discourse segment brings to *closure*. We can illustrate this with figure 1, which presents a sample RST tree with only the spans annotated (i.e., without RST labels or nucleus/satellite directed edges). This tree spans 10 elementary discourse units; each non-terminal node is annotated with the span of the subtree

rooted at that node (so the root spans all ten EDUs, while its left child spans only the first five). The final discourse unit (EDU 10) is the final EDU in three rhetorical phrases (those spanning EDUs 9-10, 6-10 and the entire discourse 1-10). We might hypothesize that the greater number of discourse phrases that a given discourse unit closes, the stronger the signal it provides that applause is licensed (and hence the greater likelihood to be followed by applause empirically). For a sentence with multiple discourse units, we featurize this value as the maximum number of rhetorical phrases closed by any unit it contains.

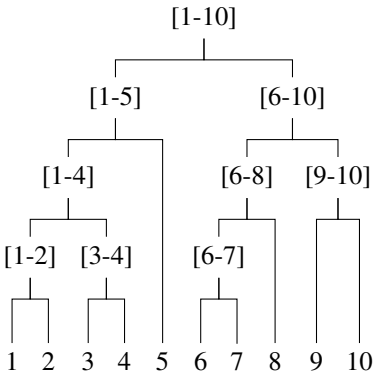


Figure 1: Unlabeled RST phrase tree; non-terminal nodes list the ranges of the elementary discourse units they span.

5 Experiments

We present two experiments to uncover the degree to which we are able to predict applause from different operationalizations of a politician's campaign speech: one in which have access to a politician's previous speeches, and can learn their specific nuances and stock phrases used to solicit applause; and another in which we seek to uncover the broader rhetorical strategies common to multiple speakers.

We refer to the following sets of features when we summarize results:

- **Guerini.** Euphony features from Guerini et al. (2015).
- **Liu.** LIWC features and additional matchers for handcrafted regular expressions from Liu et al. (2017)
- **Audio.** All acoustic features described in §4.1 above.
- **Combined.** Combination of features from Guerini, Liu, and Audio.
- **Tension.** Combination of RST (§4.2.3), repetition (§4.2.1), and delta features (§4.2.2).
- **N-gram.** Bigram features.
- **Skip-Thought.** 4800 dimensional Skip-Thought embeddings.

5.1 Intra-speaker validation

Access to a politician's previous speeches provides a great deal of evidence for understanding their rhetorical strategies for soliciting applause; speakers often give variations of the same speech at different campaign events, and rely on a fixed set of stock phrases (e.g., "Yes, We Can," "Make America Great Again") and general strategies to solicit reactions (Lu, 1999; Miller, 1939; Petrow and Sullivan, 2007). To model this, we attempt to predict a speaker's likelihood of applause using only information from their own speeches.

We use logistic regression with ℓ_2 regularization for this experiment, with hyperparameters chosen through cross-validation on the training data. We run 10-fold cross validation for each speaker, and leave-one-out cross validation for those speakers with fewer than 10 speeches (we exclude Rick Santorum from this experiment because we have only one speech from him), with whole speeches divided across folds so that no utterances from the same speech ever appear in both training and test sets. Reported results aggregate the predictions across all speakers to calculate the final accuracies. We choose utterances (or sequences of utterances) that directly precede applause as positive examples, pairing each one with a negative example randomly chosen from the same speech. Since we use different amounts of data for each speaker, we are not able to compare accuracies across all speakers, but we can see that some speakers are significantly easier to model: for example, our best model reaches 0.719 accuracy on Bernie Sanders but only 0.660 on Donald Trump.

Table 2 summarizes the results, comparing across different combinations of features as well as across a scope of a single phrase or multiple phrases. All feature combinations are scoped over a single utterance unless otherwise noted.

5.2 Inter-speaker validation

At the same time, many of the strategies identified by Heritage and Greatbatch (1986) are gener-

Model	Mean Accuracy	Mean F1	Max F1	Min F1
Guerini	0.566	0.533	0.659 (Bernie Sanders)	0.422 (Donald Trump)
Liu	0.601	0.594	0.649 (Bernie Sanders)	0.499 (Jeb Bush)
Audio	0.598	0.574	0.634 (Hilary Clinton)	0.516 (Donald Trump)
Combined	0.646	0.640	0.685 (Bernie Sanders)	0.598 (Marco Rubio)
N-gram	0.637	0.578	0.672 (Bernie Sanders)	0.478 (Barack Obama)
Combined+Tension	0.639	0.635	0.682 (Bernie Sanders)	0.585 (Jeb Bush)
Combined (3-Phrase)	0.645	0.640	0.671 (Bernie Sanders)	0.587 (Bill Clinton)
Combined+Tension (3-Phrase)	0.626	0.624	0.665 (Bernie Sanders)	0.602 (Marco Rubio)
Combined+N-gram	0.673	0.661	0.711 (Bernie Sanders)	0.600 (Marco Rubio)
Combined+Tension+N-gram	0.671	0.658	0.711 (Bernie Sanders)	0.599 (Marco Rubio)

Table 2: Intra-speaker predictive accuracy (logistic regression). The 95% confidence interval for Mean Accuracy and Mean F1 is within $\pm$ 0.005, and the 95% confidence interval for Max F1 and Min F1 (1 speaker at a time) is within $\pm$ 0.05.

alized rhetorical devices used to solicit applause; we should expect then that a model trained on a fixed set of speakers should be able to generalize to speakers not in the training data. To test this more realistic scenario, we performed K-fold cross-validation on all of the speakers in our dataset, holding out one speaker in turn for each fold (so that the same speaker did not appear in the training and test partitions).

In this experiment, we use both logistic regression and neural models (sharing training data between speakers has the added benefit of allowing us enough data to reasonably train a neural model). All logistic regression models were trained in the same way is in the intra-speaker case. Our feed-forward and LSTM models use a hidden state size of 100 for models including phrase embeddings (4800 dimensions) and a hidden state of size 25 for models without phrase embeddings. All LSTM models use a standard formulation of attention (Bahdanau et al., 2014), and all neural models are trained with dropout (Srivastava et al., 2014) and the ADAM optimizer (Kingma and Ba, 2014). We implemented the models using Keras (Chollet et al., 2015) and Tensorflow (Abadi et al., 2016).

Table 3 summarizes these results, and table 4 shows the coefficients for the most significant features.

Logistic Regression Models	Acc.	F1
Guerini	0.557	0.534
Liu	0.577	0.541
Audio	0.573	0.548
Combined	0.615	0.601
N-gram	0.594	0.578
Combined+Tension	0.617	0.605
Combined (3-Phrase)	0.614	0.601
Combined+Tension (3-Phrase)	0.615	0.600
Combined+N-gram	0.633	0.598
Combined+Tension+N-gram	0.630	0.594
Neural Models	**Acc.**	**F1**
Feed-Forward:Skip-Thought	0.577	0.562
Feed-Forward:Combined+Tension	0.620	0.620
LSTM:Skip-Thought(3-Phrase)	0.585	0.583
LSTM:Combined+Tension(3-Phrase)	0.626	0.616
LSTM:Combined+Tension(5-Phrase)	0.628	0.625
LSTM:Combined+Tension(8-Phrase)	0.629	0.621

Table 3: Inter-speaker predictive accuracy. The 95% confidence interval for each measurement of accuracy is within $\pm$ 0.005.

6 Analysis

Each of the feature classes we operationalize offers some ability to recognize what Heritage and Greatbatch (1986) term the "projectability" of applause—the ability of an audience to see an applaudable moment on the horizon.

Audio. Perhaps not surprising in retrospect is the ability of acoustic features (only summary statistics of the pitch and energy) to solicit applause: higher pitch and energy, and a broader pitch range are all predictive of applause; while past work has focused on textual indicators of applause, these results suggest that *how* a message is delivered is equally important.

Lexical. The use of explicit n-grams improves performance significantly in the intra-speaker setting, where they are able to capture stock phrases employed by the same speaker at different events. N-grams are also predictive across different speakers, though the performance gains are not as high in the inter-speaker setting.

The strongest bigrams predictive of applause include moral declaratives like *should not* (e.g., "and billionaires **should not** be able to buy elections" [Bernie Sanders]), *right to* ("you have a **right to** be angry" [Marco Rubio]), and *should be* ("They should be ashamed of that kind of behavior" [Hillary Clinton]); call-outs to the audience such as *this room* ("Love the people in **this room**"

Significant Features	Coefficient
Expression of Gratitude	0.472
LIWC FOCUSFUTURE	0.340
Homogeneity (Guerini)	0.301
Mean Energy (Audio)	0.293
LIWC BODY	0.203
Min Energy (Audio)	0.165
Max Pitch (Audio)	0.157
LIWC TENTATIVE	-0.161
LIWC THEY	-0.172
LIWC VERB	-0.216
LIWC FUNCTION	-0.228
Pitch Standard Deviation (Audio)	-0.249
LIWC SHEHE	-0.275
LIWC FOCUSPAST	-0.342

Table 4: Most significant positive and negative features for the Combined+Tension regression model in the inter-speaker setting.

[Donald Trump]) and *listening to* ("our campaign is **listening to** our Latino brothers and sisters" [Bernie Sanders]); and politically charged topics such as *political revolution*, *equal pay*, *immigration reform*, *planned parenthood*, *campaign contributors* and *police officers*.

LIWC. Among broader lexical category features, we see the LIWC FOCUSFUTURE category strongly indicative of applause; this category includes auxilaries like *will, going, gonna* (including conjunctions *I'll*) and future-oriented verbs like *anticipate*; also important are categories of BODY (including *heart, hands, brain*) and REWARD (including *succeed, optimism, great*).

Rhetorical. While RST features were not as predictive for applause as other (likely correlated) features, we still see a strong alignment between the RST features most associated with applause and those rhetorical devices outlined by Heritage and Greatbatch (1986): in particular, a clear relationship between applause and the RST category of ANTITHESIS (a contrastive relation between two discourse units with a clear nucleus and satellite, rather than two equal nuclei) and PURPOSE (a relation between a discourse unit that must take place in order for another to be realized). As expected, phrases that close more discourse units tend to be more predictive of applause.

Contextual. Though lexical features from the final utterance significantly outweigh the effects of previous context in the intra-speaker setting, in the inter-speaker case we leveraged gains from long-term context in the LSTM to reach a similar level of performance attained from the lexical features, but without access to lexical cues provided by the n-grams at all. This result suggests that the improved performance in the intra-speaker setting may be largely due to the presence of specific words and catch-phrases; the other stylistic features are more easily generalized to new speakers.

7 "Please clap"

As a further measure of out-of-sample validity, we can analyze the predictions we make for the single example where a speaker wears his communicative intent on his sleeve. On February 2, 2016, presidential candidate Jeb Bush spoke to a crowd in New Hampshire a week before their state primary. His speech ended with the following:

> So here's my pledge to you. [I] will be a commander-in-chief who will have the back of the military, I won't trash talk, I won't be a divider-in-chief or an agitator-in-chief, I won't be out there blowharding talking a big game without backing it up; I think the next President needs to be a lot quieter but send a signal that we're prepared to act in the national security interests of this country to get back in the business of creating a more peaceful world . . . . . Please clap.
>
> [Jeb Bush, Feb 2, 2016][1]

Bush's admonition to the audience ("please clap") earned criticism in news coverage at the time (Benen, 2016), but also presents us with a rare insight into a speaker's true rhetorical intention; in this case, Bush was soliciting applause and was vocal about not being able to do so.

Does our model recover this true intention? Indeed it does; while the opening *So here's my pledge to you* is predicted to not solicit applause (with applause probability of 24.8%), the segment that ends with *peaceful world* is strongly predicted to have been followed by applause (with an applause probability of 94.5%). The strongest features are again lexical (*this country*, *commander in chief*), a LIWC focus on the future (elicited by *will*), and an RST PURPOSE relation (evoked by *to get back in the business of creating a more peaceful world*).

[1] Video of this speech can be found at: https://www.youtube.com/watch?v=DdCYMvaUcrA

8 Conclusion

We present in this work a new dataset for the analysis of political rhetoric derived from the public campaign speeches of politicians during the 2016 United States presidential election, along with empirical results assessing the performance of different operationalizations of rhetoric derived from the theoretical work of Heritage and Greatbatch (1986) and others in order to measure and predict the occurrence of applause. We introduce several new features designed to capture elements of tension and release in public performance, including rhetorical contrast, closure, repetition and movement across speech segments; while each of these features in isolation is able to predict applause to varying degree and comport with our prior understanding of their utility, we find that lexicalized features are among the strongest source of information in determining applause; while audiences react to many dimensions of a speaker's style, the words they use—as slogan, stock phrases, and indicators of more complex rhetorical functions like moral valuations and imperatives—matter most.

As detailed in previous work (Liu et al., 2017; Haider et al., 2017; Clement and McLaughlin, 2016), understanding and identifying climactic moments in speeches can be useful for a variety of reasons, including learning to give better talks, automatically summarizing videos and transcripts, and analyzing social dynamics within crowds. One additional interesting application of this work is to bring to the surface occasions where a speaker uses typical applause-seeking devices but does not receive applause (the "Please Clap" moments); we leave to future work identifying the reverse, when speakers receive applause without invoking common techniques (for example, to identify instances of *claques* paid to clap).

9 Acknowledgments

Many thanks to the anonymous reviewers for their helpful feedback. The research reported in this article was supported by a UC Berkeley Fellowship for Graduate Study to J.G. and by resources provided by NVIDIA.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* .

Mark Algee-Hewitt. 2016. The machinery of suspense. http://markalgeehewitt.org/index.php/main-page/projects/the-machinery-of-suspense/.

S. E. Asch. 1951. Effects of group pressure on the modification and distortion of judgments. In H. Guetzkow, editor, *Groups, Leadership and Men*. Carnegie Press.

J. Maxwell Atkinson. 1984. Public speaking and audience responses: some techniques for inviting applause. In *Structures of Social Action*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Roland Barthes and Lionel Duisit. 1975. An introduction to the structural analysis of narrative. *New Literary History* 6(2):237–272. http://www.jstor.org/stable/468419.

Steve Benen. 2016. Jeb Bush urges audience, 'Please clap'. http://www.msnbc.com/rachel-maddow-show/jeb-bush-urges-audience-please-clap.

Dario Bertero and Pascale Fung. 2016. A long short-term memory framework for predicting humor in dialogues. In *HLT-NAACL*. pages 130–135.

Parminder Bhatia, Yangfeng Ji, and Jacob Eisenstein. 2015. Better document-level sentiment analysis from RST discourse parsing. *arXiv preprint arXiv:1509.01599* .

David Bordwell. 2013. *Narration in the fiction film*. Routledge.

Noel Carroll. 1996. Toward a theory of film suspense. In *Theorizing the Moving Image*. Cambridge University Press.

Pritish Chandna, Marius Miron, Jordi Janer, and Emilia Gómez. 2017. Monoaural audio source separation using deep convolutional neural networks. In *International Conference on Latent Variable Analysis and Signal Separation*. Springer, pages 258–266.

Lei Chen and Chong Min Lee. 2017. Predicting audience's laughter during presentations using convolutional neural network. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*. pages 86–90.

François Chollet et al. 2015. Keras.

Tanya Clement and Stephen McLaughlin. 2016. Measured applause: Toward a cultural analysis of audio collections. *Journal of Cultural Analytics* .

David Cope. 2005. *Computer models of musical creativity*. MIT Press Cambridge.

Cristian Danescu-Niculescu-Mizil, Justin Cheng, Jon Kleinberg, and Lillian Lee. 2012. You had me at hello: How phrasing affects memorability. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics, pages 892–901.

Marco Guerini, Danilo Giampiccolo, Giovanni Moretti, Rachele Sprugnoli, and Carlo Strapparava. 2013. The new release of CORPS: A corpus of political speeches annotated with audience reactions. In *Multimodal Communication in Political Speech. Shaping Minds and Social Action*, Springer, pages 86–98.

Marco Guerini, Gözde Özbal, and Carlo Strapparava. 2015. Echoes of persuasion: The effect of euphony in persuasive communication. *arXiv preprint arXiv:1508.05817* .

Fasih Haider, Fahim A Salim, Saturnino Luz, Carl Vogel, Owen Conlan, and Nick Campbell. 2017. Visual, laughter, applause and spoken expression features for predicting engagement within ted talks. *Feedback* 10:20.

John Heritage and David Greatbatch. 1986. Generating applause: A study of rhetoric and response at party political conferences. *American journal of sociology* 92(1):110–157.

Paul Hindemith. 1937. *The craft of musical composition*. Associated Music Publishers.

A Hume. 2017. Hook, line and sinker: How songwriters get into your head. *PORESO 2015: Redefining the boundaries of the Event* .

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *ACL (1)*. pages 13–24.

Yangfeng Ji and Noah A. Smith. 2017. Neural discourse structure for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 996–1005. http://aclweb.org/anthology/P17-1092.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*. pages 3294–3302.

Zhe Liu, Anbang Xu, Mengdi Zhang, Jalal Mahmud, and Vibha Sinha. 2017. Fostering user engagement: Rhetorical devices for applause generation learnt from ted talks. *arXiv preprint arXiv:1704.02362* .

Xing Lu. 1999. An ideological/cultural analysis of political slogans in Communist China. *Discourse Society, 10 (4), 487-508* .

Clifford K. Madsen and William E. Fredrickson. 1993. The experience of musical tension: A replication of Nielsen's research using the continuous response digital interface. *Journal of Music Therapy* 30(1):46–63. https://doi.org/10.1093/jmt/30.1.46.

Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. 2015. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*. pages 18–25.

C.R. Miller. 1939. *How to detect and analyze propaganda*. Town Hall, Inc.

Scott Novotney and Chris Callison-Burch. 2010. Cheap, fast and good enough: Automatic speech recognition with non-expert transcription. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 207–215.

Barbara Peskin, Larry Gillick, Yoshiko Ito, Stephen Lowe, Robert Roth, Francesco Scattone, James Baker, Janet Baker, John Bridle, Melvyn Hunt, et al. 1993. Topic and speaker identification via large vocabulary continuous speech recognition. In *Proceedings of the workshop on Human Language Technology*. Association for Computational Linguistics, pages 119–124.

Gregory A. Petrow and Terry Sullivan. 2007. Presidential persuasive advantage: Strategy, compliance-gaining and sequencing. *Congress and the Presidency* .

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. 2011. The Kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*. IEEE Signal Processing Society, EPFL-CONF-192584.

Amruta Purandare and Diane Litman. 2006. Humor: Prosody analysis and automatic recognition for F*R*I*E*N*D*S. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 208–215.

Eric S. Rabkin. 1973. *Narrative suspense: When Slim turned sideways*. University of Michigan Press.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research* 15(1):1929–1958.

Eva Strangert. 2005. Prosody in public speech: analyses of a news announcement and a political interview. In *Ninth European Conference on Speech Communication and Technology*.

D. Talkin. 2015. Reaper: Robust epoch and pitch estimator. https://github.com/google/REAPER.

Sandra A Thompson and William C Mann. 1987. Rhetorical structure theory. *IPRA Papers in Pragmatics* 1(1):79–105.

Peter Vorderer, Hans Jurgen Wulff, and Mike Friedrichsen, editors. 1996. *Suspense: Conceptualizations, Theoretical Analyses, and Empirical Explorations*. Routledge.

Attentive Interaction Model: Modeling Changes in View in Argumentation

Yohan Jo[†], **Shivani Poddar**[†], **Byungsoo Jeon**[‡], **Qinlan Shen**[†], **Carolyn P. Rosé**[†], **Graham Neubig**[†]
[†]Language Technologies Institute, [‡]Computer Science Department
School of Computer Science
Carnegie Mellon University
{yohanj,spoddar2,byungsoj,qinlans,cprose,gneubig}@cs.cmu.edu

Abstract

We present a neural architecture for modeling argumentative dialogue that explicitly models the interplay between an Opinion Holder's (OH's) reasoning and a challenger's argument, with the goal of predicting if the argument successfully changes the OH's view. The model has two components: (1) *vulnerable region detection*, an attention model that identifies parts of the OH's reasoning that are amenable to change, and (2) *interaction encoding*, which identifies the relationship between the content of the OH's reasoning and that of the challenger's argument. Based on evaluation on discussions from the Change My View forum on Reddit, the two components work together to predict an OH's change in view, outperforming several baselines. A posthoc analysis suggests that sentences picked out by the attention model are addressed more frequently by successful arguments than by unsuccessful ones.[1]

1 Introduction

Through engagement in argumentative dialogue, interlocutors present arguments with the goals of winning the debate or contributing to the joint construction of knowledge. Especially modeling the knowledge co-construction process requires understanding of both the substance of viewpoints and how the substance of an argument connects with what it is arguing against. Prior work on argumentation in the NLP community, however, has focused mainly on the first goal and has often reduced the concept of a viewpoint as a discrete side (e.g., pro vs against, or liberal vs conservative), missing more nuanced and complex details of viewpoints. In addition, while the strength of the argument and the side it represents have been addressed relatively often, the dialogical aspects of argumentation have received less attention.

To bridge the gap, we present a model that jointly considers an Opinion Holder's (OH's) expressed viewpoint with a challenger's argument in order to predict if the argument succeeded in altering the OH's view. The first component of the architecture, **vulnerable region detection**, aims to identify important parts in the OH's reasoning that are key to impacting their viewpoint. The intuition behind our model is that addressing certain parts of the OH's reasoning often has little impact in changing the OH's view, even if the OH realizes the reasoning is flawed. On the other hand, some parts of the OH's reasoning are more open to debate, and thus, it is reasonable for the model to learn and attend to parts that have a better chance to change an OH's view when addressed.

The second component of the architecture, **interaction encoding**, aims to identify the connection between the OH's sentences and the challenger's sentences. Meaningful interaction in argumentation may include agreement/disagreement, topic relevance, or logical implication. Our model encodes the interaction between every pair of the OH's and the challenger's sentences as interaction embeddings, which are then aggregated and used for prediction. Intuitively, the interactions with the most vulnerable regions of the OH's reasoning are most critical. Thus, in our complete model, the interaction embeddings are weighted by the vulnerability scores computed in the first component.

We evaluate our model on discussions from the Change My View forum on Reddit, where users (OHs) post their views on various issues, partic-

[1]Our code is available at https://github.com/yohanjo/aim.

Proceedings of NAACL-HLT 2018, pages 103–116
New Orleans, Louisiana, June 1 - 6, 2018.

ipate in discussion with challengers who try to change the OH's view, and acknowledge when their views have been impacted. Particularly, we aim to answer the following questions:

- RQ1. Does the architecture of vulnerable region detection and interaction encoding help to predict changes in view?
- RQ2. Can the model identify vulnerable sentences, which are more likely to change the OH's view when addressed? If so, what properties constitute vulnerability?
- RQ3. What kinds of interactions between arguments are captured by the model?

We use our model to predict whether a challenger's argument has impacted the OH's view and compare the result with several baseline models. We also present a posthoc analysis that illuminates the model's behavior in terms of vulnerable region detection and meaningful interaction.

For the remainder of the paper, we position our work in the literature (Section 2) and examine the data (Section 3). Then we explain our model design (Section 4). Next, we describe the experiment settings (Section 5), discuss the results (Section 6), and conclude the paper (Section 7).

2 Background

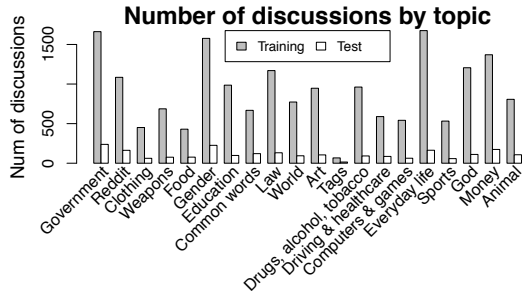
Argumentation theories have identified important dialogical aspects of (non-)persuasive argumentation, which motivate our attempt to model the interaction of OH's and challenger's arguments. Persuasive arguments build on the hearer's accepted premises (Walton, 2008) and appeal to emotion effectively (Aristotle and Kennedy, 2007). From a challenger's perspective, effective strategies for these factors could be derived from the OH's background and reasoning. On the other hand, non-persuasive arguments may commit fallacies, such as contradicting the OH's accepted premises, diverting the discussion from the relevant and salient points suggested by the OH, failing to address the issues in question, misrepresenting the OH's reasoning, and shifting the burden of proof to the OH by asking a question (Walton, 2008). These fallacies can be identified only when we can effectively model how the challenger argues in relation to the OH's reasoning.

While prior work in the NLP community has studied argumentation, such as predicting debate winners (Potash and Rumshisky, 2017; Zhang et al., 2016; Wang et al., 2017; Prabhakaran et al., 2013) and winning negotiation games (Keizer et al., 2017), this paper addresses a different angle: predicting whether an argument against an OH's reasoning will successfully impact the OH's view. Some prior work investigates factors that underlie viewpoint changes (Tan et al., 2016; Lukin et al., 2017; Hidey et al., 2017; Wei et al., 2016), but none target our task of identifying the specific arguments that impact an OH's view.

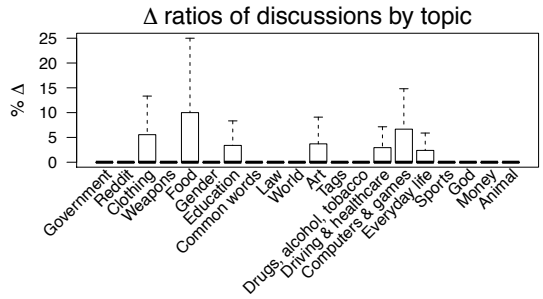
Changing an OH's view depends highly on argumentation quality, which has been the focus of much prior work. Wachsmuth et al. (2017) reviewed theories of argumentation quality assessment and suggested a unified framework. Prior research has focused mainly on the presentation of an argument and some aspects in this framework without considering the OH's reasoning. Specific examples include politeness, sentiment (Tan et al., 2016; Wei et al., 2016), grammaticality, factuality, topic-relatedness (Habernal and Gurevych, 2016b), argument structure (Niculae et al., 2017), topics (Wang et al., 2017), and argumentative strategies (e.g., anecdote, testimony, statistics) (Al Khatib et al., 2017). Some of these aspects have been used as features to predict debate winners (Wang et al., 2017) and view changes (Tan et al., 2016). Habernal and Gurevych (2016a) used crowdsourcing to develop an ontology of reasons for strong/weak arguments.

The persuasiveness of an argument, however, is highly related to the OH's reasoning and how the argument connects with it. Nonetheless, research on this relationship is quite limited in the NLP community. Existing work uses word overlap between the OH's reasoning and an argument as a feature in predicting the OH's viewpoint (Tan et al., 2016). Some studies examined the relationship between the OH's personality traits and receptivity to arguments with different topics (Ding and Pan, 2016) or degrees of sentiment (Lukin et al., 2017).

The most relevant to our work is the related task by Tan et al. (2016). Their task used the same discussions from the Change My View forum as in our work and examined various stylistic features (sentiment, hedging, question marks, etc.) and word overlap features to identify discussions that impacted the OH's view. However, our task is different from theirs in that they made predictions on



(a) Number of discussions by topic.



(b) Delta ratios in discussions by topic. (e.g., a discussion has a 10% ratio if 10% of the OH's replies have a Δ.)

Figure 1: Discussion characteristics by topic.

Opinion Holder (OH)

CMV: DNA tests (especially for dogs) are bullshit. For my line of work (which is not the DNA testing), … I have NEVER seen a DNA test return that a dog is purebred, or even anywhere close to purebred. … these tests are consistently way off on their results. … **My mother recently had a DNA test done showing she is 1/4 black.** I believe this is also incorrect since she knows who her parents and grandparents are, and none of them are black. …

Challenger 1

I'm not sure what exactly these particular DNA tests are looking at, but they are probably analyzing either SNPs or VNTRs. There's nothing stopping a SNP from mutating at any given generation, or a VNTR from shrinking or expanding due to errors during DNA replication. … **The take-home message is that DNA testing isn't complete bullshit, but it does have limitations.**

Challenger 2

Knowing your grandparents "aren't black" doesn't really rule out being 25% African American, genetically, because genes combine during fertilization almost completely randomly. … Basically, the biggest conclusion from this information is that **race is only barely genetic. It's mostly a social construct.**

Figure 2: A discussion from Change My View.

initial comments only, while we did so for all comments replied to by the OH in each discussion. Our task is more challenging because comments that come later in a discussion have a less direct connection to the original post. Another challenge is the extreme skew in class distribution in our data, whereas Tan et al. (2016) ensured a balance between the positive and negative classes.

The Change My View forum has received attention from recent studies. For example, *ad hominem* (attacking an arguer) arguments have been studied, along with their types and causes (Habernal et al., 2018). Another study annotated semantic types of arguments and analyzed the relationship between semantic types and a change in view (Hidey et al., 2017). Although this work did not look at the interaction between OHs and specific challengers, it provides valuable insight into persuasive arguments. Additionally, the semantic types may potentially allow our model to better model complex interaction in argumentation.

3 Data

Our study is based on discussions from the Change My View (CMV) forum[2] on Reddit. In this forum, users (opinion holders, OHs) post their views on a wide range of issues and invite other users (challengers) to change their expressed viewpoint. If an OH gains a new insight after reading a comment, he/she replies to that comment with a Δ symbol and specifies the reasons behind his/her view change. DeltaBot monitors the forum and marks comments that received a Δ, which we will use as labels indicating whether the comment successfully changed the OH's view.

CMV discussions provide interesting insights into how people accept new information through argumentation, as OHs participate in the discussions with the explicit goal of exposing themselves to new perspectives. In addition, the rules and moderators of this forum assure high quality discussions by requiring that OHs provide enough reasoning in the initial post and replies.

We use the CMV dataset compiled by Tan et al. (2016)[3]. The dataset is composed of 18,363 discussions from January 1, 2013–May 7, 2015 for training data and 2,263 discussions from May 8–September 1, 2015 for test data.

[2] https://www.reddit.com/r/changemyview

[3] https://chenhaot.com/pages/changemyview.html

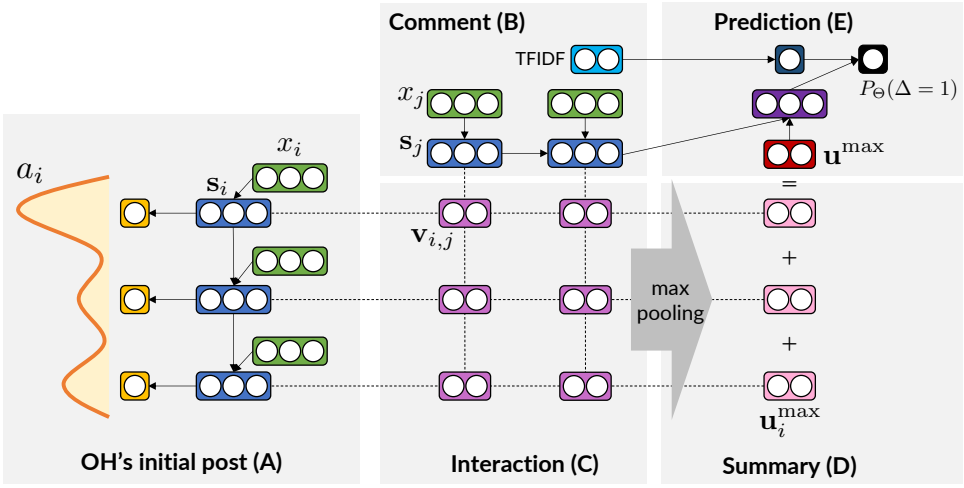


Figure 3: Architecture of Attentive Interaction Model.

Qualitative analysis We conducted qualitative analysis to better understand the data. First, to see if there are topical effects on changes in view, we examined the frequency of view changes across different topics. We ran Latent Dirichlet Allocation (Blei et al., 2003) with 20 topics, taking each discussion as one document. We assigned each discussion the topic that has the highest standardized probability. The most discussed topics are government, gender, and everyday life (Figure 1a). As expected, the frequency of changes in view differs across topics (Figure 1b). The most malleable topics are food, computers & games, clothing, art, education, and everyday life. But even in the food domain, OHs give out a Δ in less than 10% of their replies in most discussions.

In order to inform the design of our model, we sampled discussions not in the test set and compared comments that did and did not receive a Δ. A common but often unsuccessful argumentation strategy is to correct detailed reasons and minor points of the OH's reasoning—addressing those points often has little effect, regardless of the validity of the points. On the contrary, successful arguments usually catch incomplete parts in the OH's reasoning and offer another way of looking at an issue without threatening the OH. For instance, in the discussion in Figure 2, the OH presents a negative view on DNA tests, along with his/her reasoning and experiences that justify the view. Challenger 1 addresses the OH's general statement and provides a new fact, which received a Δ. On the other hand, Challenger 2 addresses the OH's issue about race but failed to change the OH's view.

When a comment addresses the OH's points, its success relies on various interactions, including the newness of information, topical relatedness, and politeness. For example, Challenger 1 provides new information that is topically dissimilar to the OH's original reasoning. In contrast, Challenger 2's argument is relatively similar to the OH's reasoning, as it attempts to directly correct the OH's reasoning. These observations motivate the design of our Attentive Interaction Model, described in the next section.

4 Model Specification

Our **Attentive Interaction Model** predicts the probability of a comment changing the OH's original view, $P(\Delta = 1)$, given the OH's initial post and the comment. The architecture of the model (Figure 3) consists of detecting vulnerable regions in the OH's post (sentences important to address to change the OH's view), embedding the interactions between every sentence in the OH's post and the comment, summarizing the interactions weighted by the vulnerability of OH sentences, and predicting $P(\Delta = 1)$.

The main idea of our model is the architecture for capturing interactions in vulnerable regions, rather than methods for measuring specific argumentation-related features (e.g., agreement/disagreement, contraction, vulnerability, etc.). To better measure these features, we need much richer information than the dataset provides (discussion text and Δs). Therefore, our proposed architecture is not to replace prior work on argumentation features, but rather to complement it at a higher, architectural level that can poten-

tially integrate various features. Moreover, our architecture serves as a lens for analyzing the vulnerability of OH posts and interactions with arguments.

Formal definition of the model (Figure 3 (A) and (B)) Denote the OH's initial post by $d^O = (x_1^O, ..., x_{M^O}^O)$, where x_i is the ith sentence, and M^O is the number of sentences. The sentences are encoded via an RNN, yielding a hidden state for the ith sentence $\mathbf{s}_i^O \in \mathbb{R}^{D^S}$, where D^S is the dimensionality of the hidden states. Similarly, for a comment $d^C = (x_1^C, ..., x_{M^C}^C)$, hidden states of the sentences $\mathbf{s}_j^C, j = 1, \cdots, M^C$, are computed.

Vulnerable region detection (Figure 3 (A)) Given the OH's sentences, the model computes the vulnerability of the ith sentence $g(\mathbf{s}_i^O) \in \mathbb{R}^1$ (e.g., using a feedforward neural network). From this vulnerability, the attention weight of the sentence is calculated as

$$a_i = \frac{\exp g(\mathbf{s}_i^O)}{\sum_{i'=1}^{M^O} \exp g(\mathbf{s}_{i'}^O)}.$$

Interaction encoding (Figure 3 (C)) The model computes the interaction embedding of every pair of the OH's ith sentence and the comment's jth sentence,

$$\mathbf{v}_{i,j} = \mathbf{h}(\mathbf{s}_i^O, \mathbf{s}_j^C) \in \mathbb{R}^{D^I},$$

where D^I is the dimensionality of interaction embeddings, and $\mathbf{h}$ is an interaction function between two sentence embeddings. $\mathbf{h}$ can be a simple inner product (in which case $D^I = 1$), a feedforward neural network, or a more complex network. Ideally, each dimension of $\mathbf{v}_{i,j}$ indicates a particular type of interaction between the pair of sentences.

Interaction summary (Figure 3 (D)) Next, for each of the OH's sentences, the model summarizes what types of meaningful interaction occur with the comment's sentences. That is, given all interaction embeddings for the OH's ith sentence, $\mathbf{v}_{i,1}, \cdots, \mathbf{v}_{i,M^C}$, the model conducts max pooling for each dimension,

$$\mathbf{u}_i^{\max} = \left(\max_j(\mathbf{v}_{i,j,1}), \cdots, \max_j(\mathbf{v}_{i,j,D^I}) \right),$$

where $\mathbf{v}_{i,j,k}$ is the kth dimension of $\mathbf{v}_{i,j}$ and $\mathbf{u}_i^{\max} \in \mathbb{R}^{D^I}$. Intuitively, max pooling is to capture the existence of an interaction and its highest intensity for each of the OH's sentences—the interaction does not have to occur in all sentences of the comment. Since we have different degrees of interest in the interactions in different parts of the OH's post, we take the attention-weighted sum of $\mathbf{u}_i^{\max}$ to obtain the final summary vector

$$\mathbf{u}^{\max} = \sum_{i=1}^{M^O} a_i \mathbf{u}_i^{\max}.$$

Prediction (Figure 3 (E)) The prediction component consists of at least one feedforward neural network, which takes as input the summary vector $\mathbf{u}^{\max}$ and optionally the hidden state of the last sentence in the comment $\mathbf{s}_{M^C}$. More networks may be used to integrate other features as input, such as TFIDF-weighted n-grams of the comment. The outputs of the networks are concatenated and fed to the final prediction layer to compute $P(\Delta = 1)$. Using a single network that takes different kinds of features as input does not perform well, because the features are in different spaces, and linear operations between them are probably not meaningful.

Loss The loss function is composed of binary cross-entropy loss and margin ranking loss. Assume there are total N^D initial posts written by OHs, and the lth post has N_l comments. The binary cross-entropy of the lth post and its tth comment measures the similarity between the predicted $P(\Delta = 1)$ and the true Δ as:

$$\begin{aligned} BCE_{l,t} = &- \Delta_{l,t} \log P_\Theta(\Delta_{l,t} = 1) \\ &- (1 - \Delta_{l,t}) \log(1 - P_\Theta(\Delta_{l,t} = 1)), \end{aligned}$$

where $\Delta_{l,t}$ is the true $\Delta \in \{0, 1\}$ of the comment and P_Θ is the probability predicted by our model with parameters Θ. Since our data is skewed to negatives, the model may overpredict $\Delta = 0$. To adjust this bias, we use margin ranking loss to drive the predicted probability of positives to be greater than the predicted probability of negatives to a certain margin. The margin ranking loss is defined on a pair of comments C_1 and C_2 with $\Delta_{C_1} > \Delta_{C_2}$ as:

$$\begin{aligned} &MRL_{C_1,C_2} = \\ &\quad \max\{0, P_\Theta(\Delta_{C_2} = 1) - P_\Theta(\Delta_{C_1} = 1) + \epsilon\}, \end{aligned}$$

where ϵ is a margin. Combining the two losses, our final loss is

$$\frac{1}{N^D} \sum_{l=1}^{N^D} \frac{1}{N_l} \sum_{t=1}^{N_l} BCE_{l,t} + \mathbb{E}_{C_1,C_2}\left[MRL_{C_1,C_2}\right].$$

	Train	Val	Test	CD
# discussions	4,357	474	638	1,548
# pairs	42,710	5,153	7,356	18,909
# positives	1,890	232	509	1,097

Table 1: Data statistics. (CD: cross-domain test)

For the expectation in the ranking loss, we consider all pairs of comments in each minibatch and take the mean of their ranking losses.

5 Experiment

Our task is to predict whether a comment would receive a Δ, given the OH's initial post and the comment. We formulate this task as binary prediction of $\Delta \in \{0, 1\}$. Since our data is highly skewed, we use as our evaluation metric the AUC score (Area Under the Receiver Operating Characteristic Curve), which measures the probability of a positive instance receiving a higher probability of $\Delta = 1$ than a negative instance.

5.1 Data Preprocessing

We exclude (1) DeltaBot's comments with no content, (2) comments replaced with *[deleted]*, (3) system messages that are included in OH posts and DeltaBot's comments, (4) OH posts that are shorter than 100 characters, and (5) discussions where the OH post is excluded. We treat the title of an OH post as its first sentence. After this, every comment to which the OH replies is paired up with the OH's initial post. A comment is labeled as $\Delta = 1$ if it received a Δ and $\Delta = 0$ otherwise. Details are described in Appendix B.

The original dataset comes with training and test splits (Figure 1a). After tokenization and POS tagging with Stanford CoreNLP (Manning et al., 2014), our vocabulary is restricted to the most frequent 40,000 words from the training data. For a validation split, we randomly choose 10% of training discussions for each topic.

We train our model on the seven topics that have the highest Δ ratios (Figure 1b). We test on the same set of topics for in-domain evaluation and on the other 13 topics for cross-domain evaluation. The main reason for choosing the most malleable topics is that these topics provide more information about people learning new perspectives, which is the focus of our paper. Some statistics of the resulting data are in Table 1.

5.2 Inputs

We use two basic types of inputs: sentence embeddings and TFIDF vectors. These basic inputs are by no means enough for our complex task, and most prior work utilizes higher-level features (politeness, sentiment, etc.) and task-specific information. Nevertheless, our experiment is limited to the basic inputs to minimize feature engineering and increase replicability, but our model is general enough to incorporate other features as well.

Sentence embeddings Our input sentences x are sentence embeddings obtained by a pretrained sentence encoder (Conneau et al., 2017) (this is different from the sentence encoder in our model). The pretrained sentence encoder is a BiLSTM with max pooling trained on the Stanford Natural Language Inference corpus (Bowman et al., 2015) for textual entailment. Sentence embeddings from this encoder, combined with logistic regression on top, showed good performance in various transfer tasks, such as entailment and caption-image retrieval (Conneau et al., 2017).

TFIDF A whole post or comment is represented as a TFIDF-weighted bag-of-words, where IDF is based on the training data. We consider the top 40,000 n-grams ($n = 1, 2, 3$) by term frequency.

Word Overlap Although integration of handcrafted features is behind the scope of this paper, we test the word overlap features between a comment and the OH's post, introduced by Tan et al. (2016), as simple proxy for the interaction. For each comment, given the set of its words C and that of the OH's post O, these features are defined as $\left[|C \cap O|, \frac{|C \cap O|}{|C|}, \frac{|C \cap O|}{|O|}, \frac{|C \cap O|}{|C \cup O|}\right]$.

5.3 Model Setting

Network configurations For sentence encoding, Gated Recurrent Units (Cho et al., 2014) with hidden state sizes 128 or 192 are explored. For attention, a single-layer feedforward neural network (FF) with one output node is used. For interaction encoding, we explore two interaction functions: (1) the inner product of the sentence embeddings and (2) a two-layer FF with 60 hidden nodes and three output nodes with a concatenation of the sentence embeddings as input. For prediction, we explore (1) a single-layer FF with either one output node if the summary vector $\mathbf{u}^{\max}$ is the only input or 32 or 64 output nodes with ReLU activation

if the hidden state of the comment's last sentence is used as input, and optionally (2) a single-layer FF with 1 or 3 output nodes with ReLU activation for the TFIDF-weighted n-grams of the comment. The final prediction layer is a single-layer FF with one output node with sigmoid activation that takes the outputs of the two networks above and optionally the word overlap vector. The margin ϵ for the ranking margin loss is 0.5. Optimization is performed using AdaMax with the initial learning rate 0.002, decayed by 5% every epoch. Training stops after 10 epochs if the average validation AUC score of the last 5 epochs is lower than that of the first 5 epochs; otherwise, training runs 5 more epochs. The minibatch size is 10.

Input configurations The prediction component of the model takes combinations of the inputs: MAX ($\mathbf{u}^{\max}$), HSENT (the last hidden state of the sentence encoder $\mathbf{s}^C_{MC}$), TFIDF (TFIDF-weighted n-grams of the comment), and WDO (word overlap).

5.4 Baseline

The most similar prior work to ours (Tan et al., 2016) predicted whether an OH would ever give a Δ in a discussion. The work used logistic regression with bag-of-words features. Hence, we also use logistic regression as our baseline to predict $P(\Delta = 1)$. Simple logistic regression using TFIDF is a relatively strong baseline, as it beat more complex features in the aforementioned task.

Model configurations Different regularization methods (L1, L2), regularization strengths ($2^\wedge\{-1, 0, 1, 2\}$), and class weights for positives (1, 2, 5) are explored. Class weights penalize false-negatives differently from false-positives, which is appropriate for the skewed data.

Input configurations The model takes combinations of the inputs: TFIDF (TFIDF-weighted n-grams of the comment), TFIDF (+OH) (concatenation of the TFIDF-weighted n-grams of the comment and the OH's post), WDO (word overlap), and SENT (the sum of the input sentence embeddings of the comment).

6 Results

Table 2 shows the test AUC scores for the baseline and our model in different input configurations. For each configuration, we chose the optimal parameters based on validation AUC scores.

Model	Inputs	ID	CD
LR	SENT	62.8	62.5
LR	TFIDF (+OH)	69.5	69.1
LR	TFIDF	70.9	**69.6**
LR	SENT+TFIDF	64.0	63.1
LR	TFIDF+WDO	71.1	69.5
AIM	MAX	70.5	67.5
AIM	MAX+TFIDF	**72.0***	69.4
AIM	MAX+TFIDF+WDO	70.9	68.4
(A)IM	HSENT	69.6	67.6
(A)IM	HSENT+TFIDF	69.0	67.6
(A)IM	MAX+TFIDF	69.5	68.1

Table 2: AUC scores. (ID: in-domain AUC (%), CD: cross-domain AUC (%), LR: logistic regression, AIM: Attention Interaction Model, (A)IM: AIM without attention.) *: $p < 0.05$ using the DeLong test compared to LR with TFIDF.

RQ1. Does the architecture of vulnerable region detection and interaction encoding help to predict changes in view? Both interaction information learned by our model and surface-level n-grams in TFIDF have strong predictive power, and attending to vulnerable regions helps. The highest score is achieved by our model (AIM) with both MAX and TFIDF as input (72.0%). The performance drops if the model does not use interaction information—(A)IM with HSENT (69.6%)—or vulnerability information—(A)IM with MAX+TFIDF (69.5%).

TFIDF by itself is also a strong predictor, as logistic regression with TFIDF performs well (70.9%). There is a performance drop if TFIDF is not used in most settings. This is unsurprising because TFIDF captures some topical or stylistic information that was shown to play important roles in argumentation in prior work (Tan et al., 2016; Wei et al., 2016). Simply concatenating both comment's and OH's TFIDF features does not help (69.5%), most likely due to the fact that a simple logistic regression does not capture interactions between features.

When the hand-crafted word overlap features are integrated to LR, the accuracy is increased slightly, but the difference is not statistically significant compared to LR without these features nor to the best AIM configuration. These features do not help AIM (70.9%), possibly because the information is redundant, or AIM requires a more de-

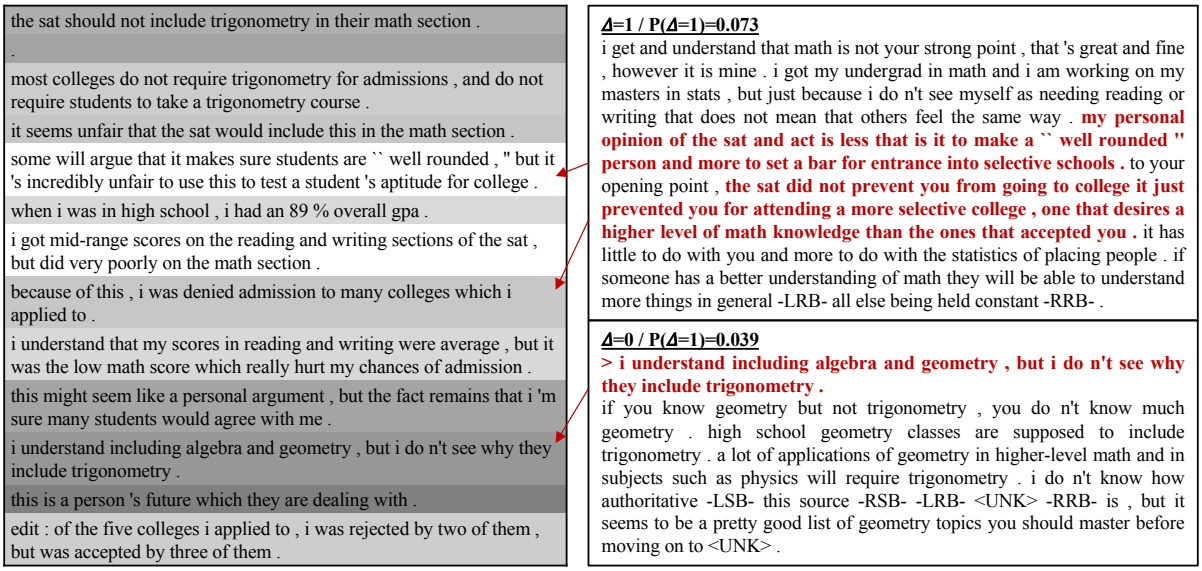


Figure 4: Example discussion with the OH's initial post (left), a successful comment (top right), and an unsuccessful comment (bottom right). The OH's post is colored based on attention weights (the higher attention the brighter). Sentences with college and SAT sections (*reading*, *writing*, *math*) get more attention than sentences with other subjects (*algebra*, *geometry*). The successful comment addresses parts with high attention, whereas the unsuccessful comment addresses parts with low attention.

liberate way of integrating hand-crafted features.

For cross-domain performance, logistic regression with TFIDF performs best (69.6%). Our interaction information does not transfer to unseen topics as well as TFIDF. This weakness is alleviated when our model uses TFIDF in addition to MAX, increasing the cross-domain score (from 67.5% to 69.4%). We expect that information about vulnerability would have more impact within domain than across domains because it may learn domain-specific information about which kinds of reasoning are vulnerable.

The rest of the section reports our qualitative analysis based on the best model configuration.

RQ2. Can the model identify vulnerable sentences, which are more likely to change the OH's view when addressed? If so, what properties constitute vulnerability? Our rationale behind vulnerable region detection is that the model is able to learn to pay more attention to sentences that are more likely to change the OH's view when addressed. If the model successfully does this, then we expect more alignment between the attention mechanism and sentences that are actually addressed by successful comments that changed the OH's view.

To verify if our model works as designed, we randomly sampled 30 OH posts from the test set, and for each post, the first successful and unsuccessful comments. We asked a native English speaker to annotate each comment with the two most relevant sentences that it addresses in the OH post, without knowledge of how the model computes vulnerability and whether the comment is successful or not.

After this annotation, we computed the average attention weight of the two selected sentences for each comment. We ran a paired sample t-test and confirmed that the average attention weight of sentences addressed by successful comments was significantly greater than that of sentences addressed by unsuccessful comments ($p < 0.05$). Thus, as expected in the case where the attention works as designed, the model more often picks out the sentences that successful challengers address.

As to what the model learns as vulnerability, in most cases, the model attends to sentences that are not punctuation marks, bullet points, or irrelevant to the topic (e.g., *can you cmv?*). A successful example is illustrated in Figure 4. More successful and unsuccessful examples are included in Appendix C.

RQ3. What kinds of interactions between arguments are captured by the model? We first use existing argumentation theories as a lens for interpreting interaction embeddings (refer to Sec-

tion 2). For this, we sampled 100 OH posts with all their comments and examined the 150 sentence pairs that have the highest value for each dimension of the interaction embedding (the dimensionality of interaction embeddings is 3 for the best performing configuration). 22% of the pairs in a dimension capture the comment asking the OH a question, which could be related to shifting the burden of proof. In addition, 23% of the top pairs in one dimension capture the comment pointing out that the OH may have missed something (e.g., *you don't know the struggles ...*). This might represent the challengers' attempt to provide premises that are missing in the OH's reasoning.

As providing missing information plays an important role in our data, we further examine if this attempt by challengers is captured in interaction embeddings even when it is not overtly signaled (e.g., *You don't know ...*). We first approximate the novelty of a challenger's information with the topic similarity between the challenger's sentence and the OH's sentence, and then see if there is a correlation between topic similarity and each dimension of interaction embeddings (details are in Appendix D). As a result, we found only a small but significant correlation (Pearson's $r = -0.04$) between topic similarity with one of the three dimensions.

Admittedly, it is not trival to interpret interaction embeddings and find alignment between embedding dimensions and argumentation theories. The neural network apparently learns complex interactions that are difficult to interpret in a human sense. It is also worth noting that the top pairs contain many duplicate sentences, possibly because the interaction embeddings may capture sentence-specific information, or because some types of interaction are determined mainly by one side of a pair (e.g., disagreement is manifested mostly on the challenger's side).

TFIDF We examine successful and unsuccessful styles reflected in TFIDF-weighted n-grams, based on their weights learned by logistic regression (top n-grams with the highest and lowest weights are in Appendix E). First, challengers are more likely to change the OH's view when talking about themselves than mentioning the OH in their arguments. For instance, first-person pronouns (e.g., *i* and *me*) get high weights, whereas second-person pronouns (e.g., *you_are* and *then_you*) get low weights. Second, different kinds of politeness seem to play roles. For example, markers of negative politeness (*can* and *can_be*, as opposed to *should* and *no*) and negative face-threatening markers (*thanks*), are associated with receiving a Δ. Third, asking a question to the OH (e.g., *why*, *do_you*, and *are_you*) is negatively associated with changing the OH's view.

7 Conclusion

We presented the Attentive Interaction Model, which predicts an opinion holder (OH)'s change in view through argumentation by detecting vulnerable regions in the OH's reasoning and modeling the interaction between the reasoning and a challenger's argument. According to the evaluation on discussions from the Change My View forum, sentences identified by our model to be vulnerable were addressed more by successful challengers than by unsuccessful ones. The model also effectively captured interaction information so that both vulnerability and interaction information increased accuracy in predicting an OH's change in view.

One key limitation of our model is that making a prediction based only on one comment is not ideal because we miss context information that connects successive comments. As a discussion between a challenger and the OH proceeds, the topic may digress from the initial post. In this case, detecting vulnerable regions and encoding interactions for the initial post may become irrelevant. We leave the question of how to transfer contextual information from the overall discussion as future work.

Our work is a step toward understanding how to model argumentative interactions that are aimed to enrich an interlocutor's perspective. Understanding the process of productive argumentation would benefit both the field of computational argumentation and social applications, including cooperative work and collaborative learning.

Acknowledgments

This research was funded by the Kwanjeong Educational Foundation and NSF grants IIS-1546393, ACI-1443068, and DGE-1745016. The authors thank Amazon for their contribution of computation time through the AWS Educate program.

References

Khalid Al Khatib, Henning Wachsmuth, Matthias Hagen, and Benno Stein. 2017. Patterns of Argumentation Strategies across Topics. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 1362–1368.

Aristotle and George Alexander Kennedy. 2007. *On Rhetoric*. A Theory of Civic Discourse. Oxford University Press, USA.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *JMLR '03: The Journal of Machine Learning Research* 3.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Association for Computational Linguistics, Doha, Qatar, pages 103–111.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Lo i c Barrault, and Antoine Bordes. 2017. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 681–691.

Tao Ding and Shimei Pan. 2016. Personalized Emphasis Framing for Persuasive Message Generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1432–1441.

Ivan Habernal and Iryna Gurevych. 2016a. What makes a convincing argument? Empirical analysis and detecting attributes of convincingness in Web argumentation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1214–1223.

Ivan Habernal and Iryna Gurevych. 2016b. Which argument is more convincing? Analyzing and predicting convincingness of Web arguments using bidirectional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1589–1599.

Ivan Habernal, Henning Wachsmuth, Iryna Gurevych, and Benno Stein. 2018. Before Name-calling: Dynamics and Triggers of Ad Hominem Fallacies in Web Argumentation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, New Orleans, LA, USA.

Christopher Hidey, Elena Musi, Alyssa Hwang, Smaranda Muresan, and Kathy McKeown. 2017. Analyzing the Semantic Types of Claims and Premises in an Online Persuasive Forum. In *Proceedings of the 4th Workshop on Argument Mining*. Association for Computational Linguistics, Copenhagen, Denmark, pages 11–21.

Simon Keizer, Markus Guhe, Heriberto Cuayahuitl, Ioannis Efstathiou, Klaus-Peter Engelbrecht, Mihai Dobre, Alex Lascarides, and Oliver Lemon. 2017. Evaluating Persuasion Strategies and Deep Reinforcement Learning methods for Negotiation Dialogue agents. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, Valencia, Spain, pages 480–484.

Stephanie Lukin, Pranav Anand, Marilyn Walker, and Steve Whittaker. 2017. Argument Strength is in the Eye of the Beholder: Audience Effects in Persuasion. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, Valencia, Spain, pages 742–753.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*. pages 55–60.

Vlad Niculae, Joonsuk Park, and Claire Cardie. 2017. Argument Mining with Structured SVMs and RNNs. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 985–995.

Peter Potash and Anna Rumshisky. 2017. Towards Debate Automation: a Recurrent Model for Predicting Debate Winners. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 2455–2465.

Vinodkumar Prabhakaran, Ajita John, and Dor e e D Seligmann. 2013. Who Had the Upper Hand? Ranking Participants of Interactions Based on Their Relative Power. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*. Asian Federation of Natural Language Processing, Nagoya, Japan, pages 365–373.

Chenhao Tan, Vlad Niculae, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2016. Winning Arguments: Interaction Dynamics and Persuasion Strategies in Good-faith Online Discussions. In *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, pages 613–624.

Henning Wachsmuth, Nona Naderi, Yufang Hou, Yonatan Bilu, Vinodkumar Prabhakaran, Tim Alberdingk Thijm, Graeme Hirst, and Benno Stein. 2017. Computational Argumentation Quality Assessment in Natural Language. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, Valencia, Spain, pages 176–187.

Douglas Walton. 2008. *Informal Logic*. A Pragmatic Approach. Cambridge University Press.

Lu Wang, Nick Beauchamp, Sarah Shugars, and Kechen Qin. 2017. Winning on the Merits: The Joint Effects of Content and Style on Debate Outcomes. *Transactions of Association of Computational Linguistics* 5:219–232.

Zhongyu Wei, Yang Liu, and Yi Li. 2016. Is This Post Persuasive? Ranking Argumentative Comments in Online Forum. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 195–200.

Justine Zhang, Ravi Kumar, Sujith Ravi, and Cristian Danescu-Niculescu-Mizil. 2016. Conversational Flow in Oxford-style Debates. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 136–141.

A Implementation Details

A.1 Topics in the Data

Topics are extracted using LatentDirichletAllocation in scikit-learn v0.19.1, with the following setting:

- `n_components`: 20
- `max_iter`: 200
- `learning_method`: online
- `learning_offset`: 50

A.2 AIM

We implemented our model in PyTorch 0.3.0.

A.3 Baseline

We use LogisticRegression in scikit-learn v0.19.1, with the default settings.

A.4 TFIDF Features

TFIDF is extracted using TfidfVectorizer in scikit-learn v0.19.1, with the default setting.

B Data Preprocessing

In the CMV forum, DeltaBot replies to an OH's comment with the confirmation of a Δ, along with the user name to which the OH replied. For most OH replies, the (non-)existence of a Δ indicates whether a comment to which the OH replied changed the OH's view. However, an OH's view is continually influenced as they participate in argumentation, and thus a Δ given to a comment may not necessarily be attributed to the comment itself. One example is when a comment does not receive a Δ when the OH reads it for the first time, but the OH comes back and gives it a Δ after they interact with other comments. In such cases, we may want to give a credit to the comment that actually led the OH to reconsider a previous comment and change the view.

Hence, we use the following labeling that considers the order in which OHs read comments. We treat the (non-)existence of a Δ in an OH comment as a label for the last comment that the OH read. We reconstruct the order in which the OH reads comments as follows. We assume that when the OH writes a comment, he/she has read all prior comments in the path to that comment.

Based on this assumption, we linearize (i.e., flatten) the original tree structure of the initial post and all subsequent comments into a linear sequence S. Starting with empty S, for each of the OH's comments in chronological order, its ancestor comments that are yet to be in S and the comment itself are appended to S. And for each of the OH's comments, its preceding comment in S is labeled with $\Delta = 1$ if the OH's comment has a Δ and 0 otherwise.

This ensures that the label of a comment to which the OH replied is the (non-)existence of a Δ in the OH's first reply. If an OH reply is not the first reply to a certain comment (as in the scenario mentioned above), or a comment to which the OH replied is missing, the (non-)existence of a Δ in that reply is assigned to the comment that we assume the OH read last, which is located right before the OH's comment in the restructured sequence.

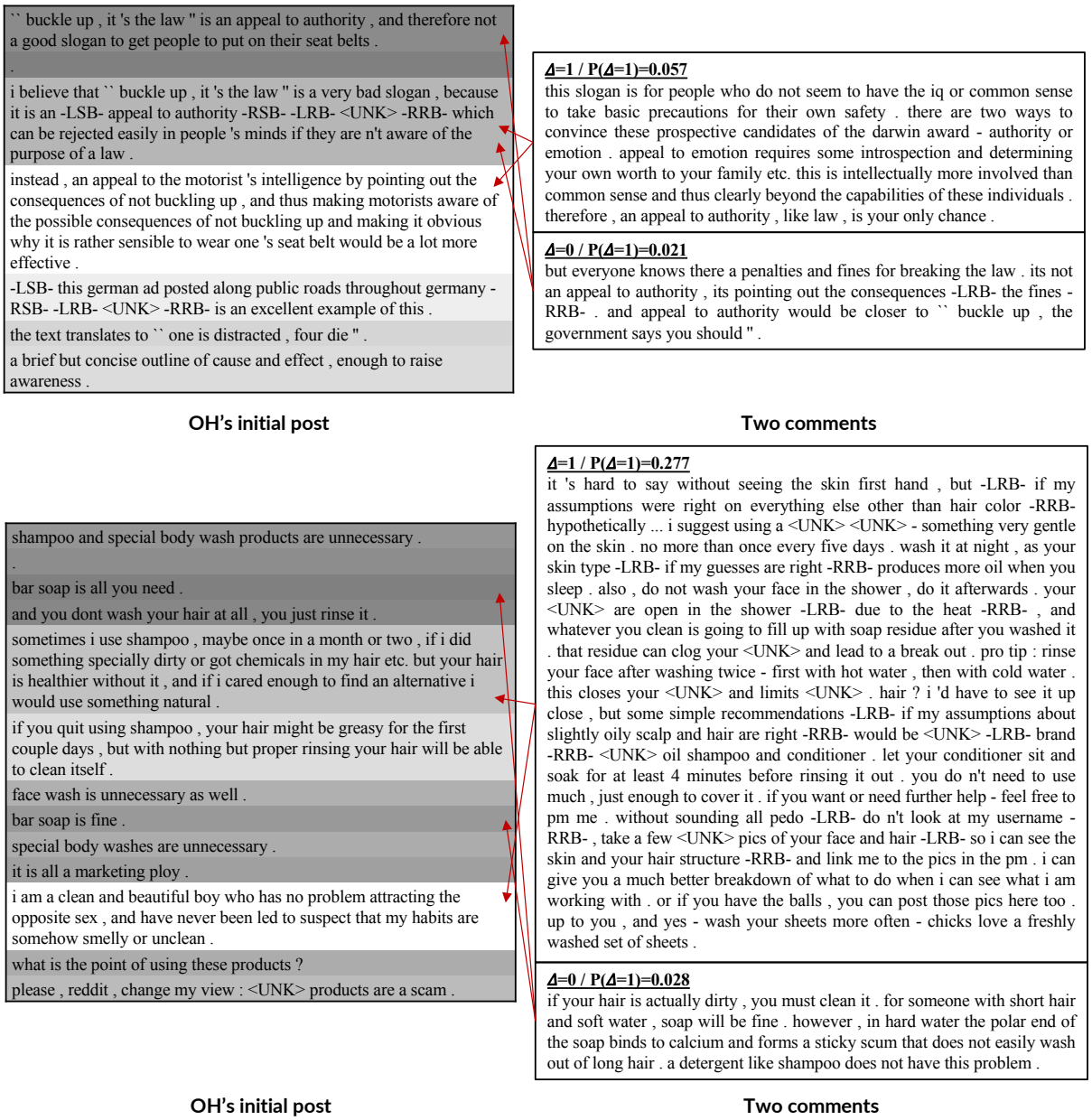


Figure 5: Successful examples of vulnerable region detection.

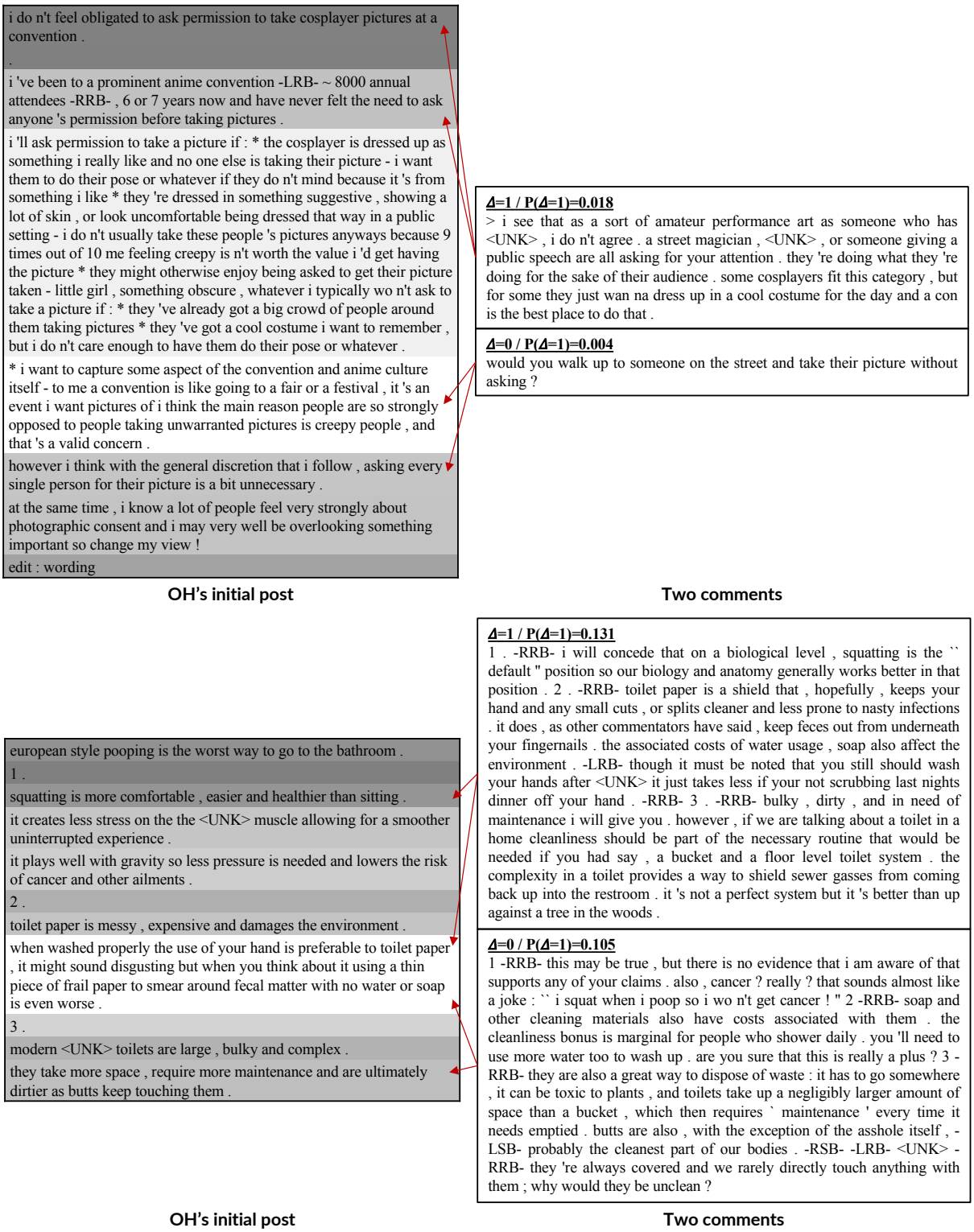


Figure 6: Unsuccessful examples of vulnerable region detection.

n-grams for $\Delta = 1$	n-grams for $\Delta = 0$
and, in, for, use, it, on, thanks, often, delta, time, depression, -RRB-, lot, -LRB-, or, i, can, &, with, more, as, band, *, #, me, -LRB-_-RRB-, can_be, has, deltas, when	?, >, sex, why, do_you, wear, relationship, child, are_you, op, mother, should, wearing, teacher, then, it_is, same, no, circumcision, you_are, then_you, baby, story

Table 3: Top n-grams with the most positive/negative weights for logistic regression.

C Vulnerability Examples

Figure 5 and Figure 6 show successful and unsuccessful examples of vulnerable region detection. All examples are from the test set.

D Topic Similarity between Sentences

The topic similarity between a pair of sentences is computed as the consine similarity between the topic distributions of the sentences.

The first step is to extract topics. Using LatentDirichletAllocation in scikit-learn v0.19.1, we ran LDA on the entire data with 100 topics, taking each post/comment as a document. We treat the top 100 words for each topic as topic words.

The second step is to compute the topic distribution of each sentence. We simply counted the frequency of occurrences of topic words for each topic, and normalized the frequencies across topics.

Lastly, we computed the cosine similarity between the topic distributions of a pair of sentences.

E Top TFIDF n-grams

The n-grams that contribute most to Δ prediction for logistic regression are shown in table 3.

Automatic Focus Annotation: Bringing Formal Pragmatics Alive in Analyzing the Information Structure of Authentic Data

Ramon Ziai **Detmar Meurers**
Collaborative Research Center 833
University of Tübingen
{rziai,dm}@sfs.uni-tuebingen.de

Abstract

Analyzing language in context, both from a theoretical and from a computational perspective, is receiving increased interest. Complementing the research in linguistics on discourse and information structure, in computational linguistics identifying discourse concepts was also shown to improve the performance of certain applications, for example, Short Answer Assessment systems (Ziai and Meurers, 2014).

Building on the research that established detailed annotation guidelines for manual annotation of information structural concepts for written (Dipper et al., 2007; Ziai and Meurers, 2014) and spoken language data (Calhoun et al., 2010), this paper presents the first approach automating the analysis of focus in authentic written data. Our classification approach combines a range of lexical, syntactic, and semantic features to achieve an accuracy of 78.1% for identifying focus.

1 Introduction

The interpretation of language is well known to depend on context. Both in theoretical and computational linguistics, discourse and information structure of sentences are thus receiving increased interest: attention has shifted from the analysis of isolated sentences to the question how sentences are structured in discourse and how information is packaged in sentences analyzed in context.

As a consequence, a rich landscape of approaches to discourse and information structure has been developed (Kruijff-Korbayová and Steedman, 2003). Among these perspectives, the Focus-Background dichotomy provides a particularly valuable structuring of the information in a sentence in relation to the discourse. (1) is an example question-answer pair from Krifka and Musan (2012, p. 4) where the focus in the answer is marked by brackets.

(1) Q: *What did John show Mary?*
A: *John showed Mary* $[\![$*the PICtures*$]\!]_F$.

In the answer in (1), the NP *the pictures* is focussed and hence indicates that there are alternative things that John could show Mary. It is commonly assumed that focus here typically indicates the presence of alternative *denotations* (denotation focus, Krifka and Musan 2012, p.8), making it a semantic notion. Depending on the language, different devices are used to mark focus, such as prosodic focus marking or different syntactic constructions (e.g. clefts). In this paper, we adopt a notion of focus based on alternatives, as advanced by Rooth (1992) and more recently, Krifka and Musan (2012), who define focus as indicating "the presence of alternatives that are relevant for the interpretation of linguistic expressions" (Krifka and Musan, 2012, p. 7). Formal semantics has tied the notion of alternatives to an explicit relationship between questions and answers called Question-Answer Congruence (Stechow, 1991), where the idea is that an answer is congruent to a question if both evoke the same set of alternatives. Questions can thus be seen as a way of making alternatives explicit in the discourse, an idea also taken up by the Question-Under-Discussion (QUD) approach (Roberts, 2012) to discourse organization.

Complementing the theoretical linguistic approaches, in the last decade corpus-based approaches started exploring which information structural notions can reliably be annotated in what kind of language data. While the information status (Given-New) dimension can be annotated successfully (Riester et al., 2010; Nissim et al., 2004) and even automated (Hempelmann et al., 2005; Nissim, 2006; Cahill and Riester, 2012), the inter-annotator agreement results for Focus-Background (Ritz et al., 2008; Calhoun et al., 2010) show that it is difficult to obtain high levels of agreement, especially due to disagreement

Proceedings of NAACL-HLT 2018, pages 117–128
New Orleans, Louisiana, June 1 - 6, 2018.

about the extent or size of the focused unit.

More recently, Ziai and Meurers (2014) showed that for data collected in task contexts including explicit questions, such as answers to reading comprehension questions, reliable focus annotation is possible. In addition, an option for externally validating focus annotation was established by showing that such focus annotation improves the performance of Short Answer Assessment (SAA) systems. Focus enables the system to zoom in on the part of the answer addressing the question instead of considering all parts of the answer as equal.

In this paper, we want to build on this strand of research and develop an approach for automatically identifying focus in authentic data including explicit question contexts. In contrast to Calhoun (2007) and Sridhar et al. (2008), who make use of prosodic properties to tackle the identification of focus for content words in spoken language data, we target the analysis of written texts.

We start in section 2 by discussing relevant related work before introducing the gold standard focus annotation we are using as foundation of our work in section 3. Section 4 then presents the different types of features used for predicting which tokens form a part of the focus. In section 5 we employ a supervised machine learning setup to evaluate the perspective and specific features in terms of the ability to predict the gold standard focus labeling. Building on these intermediate results and the analysis thereof in section 6, in section 7 we then present two additional feature groups which lead to our final focus detection model. Finally, section 8 explores options for extrinsically showing the value of the automatic focus annotation for the automatic meaning assessment of short answers. It confirms that focus analysis pays off when aiming to generalize assessment to previously unseen data and contexts.

2 Previous Approaches

There is only a very small number of approaches dealing with automatically labeling information structural concepts.[1] Most approaches related to detecting focus automatically almost exclusively center on detecting the 'kontrast' notion in the English Switchboard corpus (Calhoun et al., 2010). We therefore focus on the Switchboard-based approaches here.

The availability of the annotated Switchboard corpus (Calhoun et al., 2005, 2010) sparked interest in information-structural categories and enabled several researchers to publish studies on detecting focus. This is especially true for the Speech Processing community, and indeed many approaches described below are intended to improve computational speech applications in some way, by detecting prominence through a combination of various linguistic factors. Moreover, with the exception of Badino and Clark (2008), all approaches use prosodic or acoustic features.

All approaches listed below tackle the task of detecting 'kontrast' (as focus is called in the Switchboard annotation) automatically on various subsets of the corpus using different features and classification approaches. For each approach, we therefore report the features and classifier used, the data set size as reported by the authors, the (often very high) majority baseline for a binary distinction between 'kontrast' and background, and the best accuracy obtained. If available in the original description of the approach, we also report the accuracy obtained without acoustic and prosodic features.

Calhoun (2007) investigated how focus can be predicted through what she calls "prominence structure". The essential claim is that a "focus is more likely if a word is more prominent than expected given its syntactic, semantic and discourse properties". The classification experiment is based on 9,289 words with a 60% majority baseline for the 'background' class. Calhoun (2007) reports 77.7% for a combination of prosodic, syntactic and semantic features in a logistic regression model. Without the prosodic and acoustic features, the accuracy obtained is at 74.8%. There is no information on a separation between training and test set, likely due to the setup of the study being geared towards determining relevant factors in predicting focus, not building a focus prediction model for a real application case. Relatedly, the approach uses only gold-standard annotation already available in the corpus as the basis for features, not automatic annotation.

Sridhar et al. (2008) use lexical, acoustic and part-of-speech features in trying to detect pitch accent, givenness and focus. Concerning focus, the work attempts to extend Calhoun (2007)'s analysis to "understand what prosodic and acoustic dif-

[1] For a broader perspective of computational approaches in connection with information structure, see Stede (2012).

ferences exist between the focus classes and background items in conversational speech". 14,555 words of the Switchboard corpus are used in total, but filtered for evaluation later to balance the skewed distribution between 'kontrast' and 'background'. With the thus obtained random baseline of 50%, Sridhar et al. (2008) obtain 73% accuracy when using all features, which again drops only slightly to 72.95% when using only parts of speech. They use a decision tree classifier to combine the features in 10-fold cross-validation for training and testing.

Badino and Clark (2008) aim to model contrast both for its role in analyzing discourse and information structure, and for its potential in speech applications. They use a combination of lexical, syntactic and semantic features in an SVM classifier. No acoustic or prosodic features are employed in the model. In selecting the training and testing data, they filter out many 'kontrast' instances, such as those triggered across sentence boundaries, those above the word level, and those not sharing the same broad part of speech with the trigger word. The resulting data set has 8,602 instances, of which 96.8% are 'background'. Badino and Clark (2008) experiment with different kernel settings for the SVM and obtain the best result of 97.19% using a second-order polynomial kernel, and leave-one-out testing.

In contrast to all approaches above, we target the analysis of written texts, for which prosodic and acoustic information is not available, so we must rely on lexis, syntax and semantics exclusively. Also, the vast majority of the approaches discussed make direct use of the manually annotated information in the corpus they use in order to derive their features. While this is a viable approach when the aim is to determine the relevant factors for focus detection, it does not represent a real-life case where annotated data often unavailable. In our focus detection model, we only use automatically determined annotation as the basis for our features for predicting focus.

Since our approach also makes use of question properties, it is also worth mentioning that there are a number of approaches on Answer Typing as a step in Question Answering (QA) approaches in order to constrain the search space of possible candidate answers and improve accuracy. While earlier approaches such as Li and Roth (2002) used a fixed set of answer types for classifying factoid questions, other approaches such as Pinchak and Lin (2006) avoid assigning pre-determined classes to questions and instead favor a more data-driven label set. In more recent work, Lally et al. (2012) use a sophisticated combination of deep parsing, lexical clues and broader question labels to analyze questions.

3 Data

The present work is based on the German CREG corpus (Ott et al., 2012). CREG contains responses by American learners of German to comprehension questions on reading texts. Each response is rated by two teaching assistants with regard to whether it answers the question or not. While many responses contain ungrammatical language, the explicit questions in CREG generally make it possible to interpret responses. More importantly for our work, they can be seen as Questions Under Discussion and thus form an ideal foundation for focus annotation in authentic data.

As a reference point for the automatic detection of focus, we used the CREG-ExpertFocus data set (De Kuthy et al., 2016) containing 3,187 student answers and 990 target answers (26,980 words in total). It was created using the incremental annotation scheme described in Ziai and Meurers (2014), where annotators first look at the surface question form, then determine the set of alternatives, and finally mark instances of the alternative set in answers. De Kuthy et al. (2016) report substantial agreement in CREG-ExpertFocus ($\kappa \geq .7$) and provide an adjudicated gold standard, which thus presents a high-quality basis for training our focus detection classifier.

4 Focus Detection Model

As described in section 3 above, focus was marked in a span-based way in the data set used: each instance of focus starts at a specific word and ends at another word. Since in principle any part of speech can be focused, we cannot constrain ourselves to a pre-defined set of markables for automatic classification. We therefore conceptualized the task of automatic focus detection on a per-word level: for each word in an answer, as identified by the OpenNLP tokenizer and sentence segmenter[2], the classifier needs to decide whether it is an instance of *focus* or *background*. Besides the choice of

[2] `http://opennlp.apache.org`

classification algorithm, the crucial question naturally is the choice of linguistic features, which we turn to next.

4.1 Features

Various types of linguistic information on different linguistic levels can in principle be relevant for focus identification, from morphology to semantics. We start by exploring five groups of features, which are outlined below. In section 7, we discuss two more groups designed to address specific problems observed with the initial model.

Syntactic answer properties (SynAns) A word's part-of-speech and syntactic function are relevant general indicators with respect to focus: since we are dealing with meaning alternatives, the meaning of e.g. a noun is more likely to denote an alternative than a grammatical function word such as a complementizer or article.

Similarly, a word in an argument dependency relation is potentially a stronger indicator for a focused alternative in a sentence than a word in an adjunct relation. We therefore included two features: the word's **part-of-speech** tag in the STTS tag set (Schiller et al., 1995) determined using TreeTagger (Schmid, 1994), and the **dependency relation to the word's head** in the Hamburg dependency scheme (Foth et al., 2014, p. 2327) determined using MaltParser (Nivre et al., 2007) as features in our model.

Question properties The question constitutes the direct context for the answer and dictates its information structure and information requirements to fulfill. In particular, the type of *wh*-phrase (if present) of a question is a useful indicator of the type of required information: a *who*-question, such as 'Who rang the doorbell?', will typically be answered with a noun phrase, such as 'the milkman'. We identified **surface question forms** such as *who*, *what*, *how* etc. using a regular expression approach developed by Rudzewitz (2015) and included them as features. Related to question forms, we also extracted the question word's **dependency relation to its head**, analogous to the answer feature described above.

Surface givenness As a rough and robust approximation to information status, we add a boolean feature indicating the **presence of the current word in the question**. We use the lemmatized form of the word as determined by TreeTagger (Schmid, 1994).

Positional properties Where a word occurs in the answer or the question can be relevant for its information structural status. It has been observed since Halliday (1967) that given material tends to occur earlier in sentences (here: answers), while new or focused content tends to occur later. We encode this observation in three different features: the **position of the word in the answer** (normalized by sentence length), the **distance from the finite verb** (in words), and the **position of the word in the question** (if it is given).

Conjunction features To explicitly tie answer properties to question properties, we explored different combinations of the features described above. Specifically, we encoded the **current word's POS depending on the question form**, and the **current word's POS depending on the *wh*-word's POS**. To constrain the feature space and get rid of unnecessary distinctions, we converted the answer word's POS to a coarse-grained version before computing these features, which collapses all variants of determiners, pronouns, adjectives/adverbs, prepositions, nouns and verbs into one label, respectively.[3]

5 Intrinsic Evaluation

5.1 Setup

To employ the features described above in an actual classifier, we trained a logistic regression model using the WEKA toolkit (Hall et al., 2009). We also experimented with other classification algorithms such as SVMs, but found that they did not offer superior performance for this task. The data set used consists of all expert focus annotation available (3,187 student answers, see section 3), with the exception of the answers occurring in the extrinsic evaluation test set we use in section 8, which leaves a total of 2,240 student answers with corresponding target answers and questions. We used 10-fold cross-validation on this data set to experiment and select the optimal model for focus detection.

[3]For a list (in German) of the full tag set, see `http://www.ims.uni-stuttgart.de/forschung/ressourcen/lexika/TagSets/stts-table.html`

5.2 Results

Table 1 lists the accuracies[4] obtained for our different feature groups, as well as three baselines: a POS baseline, following Sridhar et al. (2008), a baseline that only includes the simple givenness feature, and the majority baseline. The majority class is *focus*, occurring in 58.1% of the 26,980 cases (individual words).

	Accuracy for		
Feature set	*focus*	*backgr.*	both
Majority baseline	100%	0%	58.1%
Givenness baseline	81.5%	42.5%	65.1%
POS baseline	89.2%	39.6%	68.4%
SynAns	82.8%	50.3%	69.2%
+ Question	83.8%	53.1%	70.9%
+ Given	84.8%	62.0%	74.8%
+ Position	84.9%	66.5%	77.2%
+ Conjunction	85.2%	66.7%	**77.4%**

Table 1: Initial focus detection model

We can see that each feature group incrementally adds to the final model's performance, with particularly noticeable boosts coming from the givenness and positional features. Another clear observation is that the classifier is much better at detecting *focus* than *background*, possibly also due to the skewedness of the data set. Note that performance on *background* increases also with the addition of the 'Question' feature set, indicating the close relation between the set of alternatives introduced by the question and the focus selecting from that set, even though our approximation to computationally determining alternatives in questions is basic. It is also clear that the information intrinsic in the answers, as encoded in the 'SynAns' and 'Position' feature sets, already provides significant performance benefits, suggesting that a classifier trained only on these features could be trained and applied to settings where no explicit questions are available.

6 Qualitative Analysis

In order to help explain the gap between automatic and manual focus annotation, let us take a step back from quantitative evaluation and examine a few characteristic examples in more detail.

Figure 1 shows a case where a *why*-question is answered with an embedded 'weil' (because) clause. The classifier successfully marked 'weil' and the end of the clause as *focus*, but left out the pronoun 'es' (it) in the middle, presumably because pronouns are given and often not focused in other answers. We did experiment with using a sequence classification approach in order to remedy such problems, but it performed worse overall than the logistic regression model we presented in section 4. We therefore suggest that in such cases, a global constraint stating that *why*-questions are typically answered with a full clause would be a more promising approach, combining knowledge learned bottom-up from data with top-down linguistic insight.

In Figure 2, we can see two different problems. One is again a faulty gap, namely the omission of the conjunction 'und' (and). The other is the focus marking of the word 'AG' (corporation) in the beginning of the sentence: since the question asks for an enumeration of the institutions that form a corporation, marking 'AG' as focused is erroneous. This problem likely occurs often with nouns because the classifier has learned that content words are often focused. Moreover, the surface givenness feature does not encode that 'AG' is in fact an abbreviation of 'Aktiengesellschaft' and therefore given. It would thus be beneficial to extend our analysis of givenness beyond surface identity, a direction we explore in the next section.

Finally, Figure 3 presents a case where an enumeration is marked correctly, including the conjunctive punctuation in between, showing that cases of longer foci are indeed within reach for a word-by-word focus classifier.

7 Extending the Model

Based on our analysis of problematic cases outlined in the previous section, we explored two different avenues for improving our focus detection model, which we describe below.

7.1 Distributional Givenness

We have seen in section 5.2 that surface-based givenness is helpful in predicting focus. However, it clearly has limitations, as for example synonymy cannot be captured on the surface. We also exemplified one such limitation in Figure 2. In order to overcome these limitations, we implemented an approach based on distributional semantics. This avenue is motivated by the fact that Ziai et al. (2016) have shown Givenness modeled

[4]We show per-class and overall accuracies, the former is also known as recall or true positive rate.

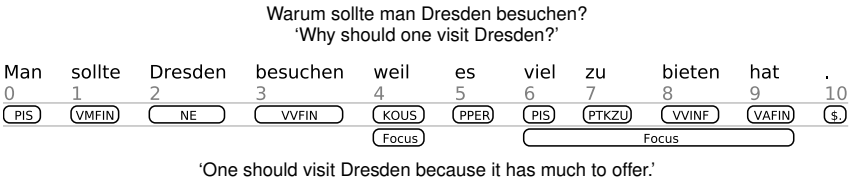


Figure 1: Focus with a faulty gap in between

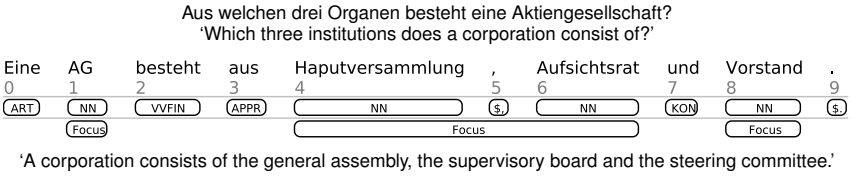


Figure 2: Focus with a faulty outlier (and a faulty gap)

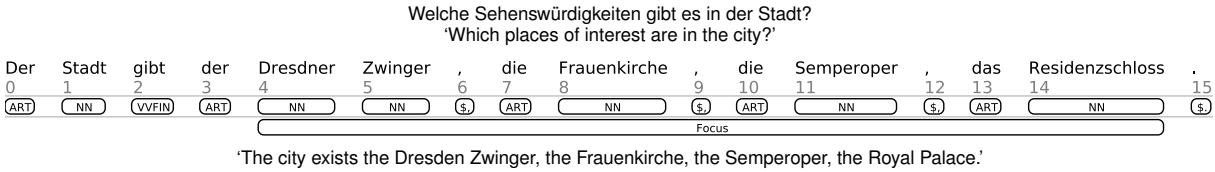


Figure 3: Enumeration with correct focus

as distributional similarity to be helpful for SAA at least in some cases. We used the word vector model they derived from the DeWAC corpus (Baroni et al., 2009) using word2vec's continuous bag-of-words training algorithm with hierarchical softmax (Mikolov et al., 2013). The model has a vocabulary of 1,825,306 words and uses 400 dimensions for each.

Having equipped ourselves with a word vector model, the question arises how to use it in focus detection in such a way that it complements the positive impact that surface-based givenness already demonstrates. Rather than using an empirically determined (and hence data-dependent) empirical threshold for determining givenness as done by Ziai et al. (2016), we here use raw cosine similarities[5] as features and let the classifier assign appropriate weights to them during training. Concretely, we calculate **maximum, minimum and average cosine between the answer word and the question words**. As a fourth feature, we calculate the **cosine between the answer word and the additive question word vector**, which is the sum of the individual question word vectors.

7.2 Constituency-based Features

Another source of evidence we wanted to exploit is constituency-based syntactic annotation. So far, we have worked with part-of-speech tags and dependency relations as far as syntactic representation is concerned. However, while discontinuous focus is possible, focus as operationalized in the scheme by Ziai and Meurers (2014) most often marks an adjacent group of words, a tendency that our word-based classifier did not always follow, as exemplified by the cases in Figures 1 and 2. Such groups very often correspond to a syntactic phrase, so constituent membership is likely indicative in predicting the focus status of an individual word. Similarly, the topological field (Höhle, 1986) identifying the major section of a sentence in relation to the clausal main verb is potentially relevant for a word's focus status.

Cheung and Penn (2009) present a parsing model that demonstrates good performance in determining both topological fields and phrase structure for German. The model is trained on the TüBa-D/Z treebank (Telljohann et al., 2004), whose rich syntactic model encodes topological fields as nodes in the syntax tree itself. Following Cheung and Penn (2009), we trained an updated version of their model using the current version of the Berkeley Parser (Petrov and Klein, 2007) and release 10 of the TüBa-D/Z.[6]

Based on the new parsing model, we integrated two new features into our focus detection model:

[5] We normalize cosine similarity as cosine distance to obtain positive values between 0 and 2: $dist = 1 - sim$

[6] `http://www.sfs.uni-tuebingen.de/en/ascl/resources/corpora/tueba-dz.html`

the **direct parent constituent node of a word** and **the nearest topological field node of a word**.

7.3 Final Results

Table 2 shows the impact of the new feature groups discussed above.

Feature set	Accuracy for *focus*	*backgr.*	both
Majority baseline	100%	0%	58.1%
Givenness baseline	81.5%	42.5%	65.1%
POS baseline	89.2%	39.6%	68.4%
Initial model (sec. 5.2)	85.2%	66.7%	77.4%
+ dist. Givenness	84.7%	68.0%	77.7%
+ constituency	84.8%	68.7%	**78.1%**

Table 2: Final focus detection performance

While the improvements may seem modest quantitatively, they show that the added features are well-motivated and do make an impact. Overall, it is especially apparent that the key to better performance is reducing the number of false positives in this data set: while the accuracy for focus stays roughly the same, the one for background improves steadily with each feature set addition.

8 Extrinsic Evaluation

Complementing the intrinsic evaluation above, in this section we demonstrate how focus can be successfully used to improve performance in an authentic CL task, namely Short Answer Assessment (SAA).

8.1 Setup

It has been pointed out that evaluating the annotation of a theoretical linguistic notion only intrinsically is problematic because there is no non-theoretical grounding involved (Riezler, 2014). Therefore, besides a comparison to the gold standard, we also evaluated the resulting annotation in a larger computational task, the automatic meaning assessment of short answers to reading comprehension questions. Here the goal is to decide, given a question (Q) and a correct target answer (TA), whether the student answer (SA) actually answers the question or not. An example from Meurers et al. (2011) is shown in Figure 4.

We used the freely available CoMiC system (Comparing Meaning in Context, Meurers et al. 2011) as a testbed for our experiment. CoMiC is an alignment-based system operating in three stages:

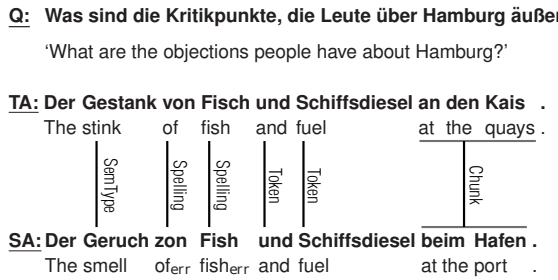


Figure 4: Short Answer Assessment example

1. Annotating linguistic units (words, chunks and dependencies) in student and target answer on various levels of abstraction

2. Finding alignments of linguistic units between student and target answer based on annotation (see Figure 4)

3. Classifying the student answer based on number and type of alignments (see Table 3), using a supervised machine learning setup

Feature	Description
1. Keyword Overlap	Percent of dependency heads aligned (relative to target)
2./3. Token Overlap	Percent of aligned target/student tokens
4./5. Chunk Overlap	Percent of aligned target/student chunks (as identified by OpenNLP[3])
6./7. Triple Overlap	Percent of aligned target/student dependency triples
8. Token Match	Percent of token alignments that were token-identical
9. Similarity Match	Percent of token alignments resolved using PMI-IR (Turney, 2001)
10. Type Match	Percent of token alignments resolved using GermaNet hierarchy (Hamp and Feldweg, 1997)
11. Lemma Match	Percent of token alignments that were lemma-resolved
12. Synonym Match	Percent of token alignments sharing same GermaNet synset
13. Variety of Match (0-5)	Number of kinds of token-level alignments (features 8–12)

Table 3: Standard features in the CoMiC system

In stage 2, CoMiC integrates a simplistic approach to givenness, excluding all words from alignment that are mentioned in the question. We transferred the underlying method to the notion of focus and implemented a component that excludes all non-focused words from alignment, resulting

[3] http://opennlp.apache.org/

in alignments between focused parts of answers only. The hypothesis is that the alignment of focused elements in answers adds information about the quality of the answer with respect to the question, leading to a higher answer classification accuracy.

We experimented with two different settings involving the standard CoMiC system and a focus-augmented variant: i) using standard CoMiC with the givenness filter by itself as a baseline, and ii) augmenting standard CoMiC by additionally producing a focus version of each classification feature in Table 3. In each case, we used WEKA's *k*-nearest-neighbor implementation for CoMiC, following positive results by Rudzewitz (2016).

We use two test sets randomly selected from the CREG-5K data set (Ziai et al., 2016), one based on an 'unseen answers' and one based on an 'unseen questions' test scenario, based on the methodology of (Dzikovska et al., 2013): in 'unseen answers', the test set can contain answers to the same questions already part of the training set (but not the answers themselves), whereas in 'unseen questions' both questions and answers are new in the test set. In order to arrive at a fair and generalizable testing setup, we removed all answers from the CREG-5K training set that also occur in the CREG-ExpertFocus set used to train our focus detection classifier. This ensures that neither the focus classifier nor CoMiC have seen any of the test set answers before.

The resulting smaller training set contains 1606 student answers, while the test sets contain 1002 (unseen answers) and 1121 (unseen questions), respectively.

8.2 Results

Table 4 summarizes the results for the different CoMiC variants and test sets in terms of accuracy in classifying answers as *correct* vs. *incorrect*. 'Standard CoMiC' refers to the standard CoMiC system and '+Focus' refers to the augmented system using both feature versions. For reference on what is possible with Focus information, we provide the results of the oracle experiment by De Kuthy et al. (2016), even though the test setup and data setup are slightly different. In addition to our two test sets introduced above, we tested the systems on the training set using 10-fold cross validation. We also provide the majority baseline of the respective data set along with the majority class.

One can see that in general, the focus classifier seems to introduce too much noise to positively impact classification results. The standard CoMiC system outperforms the focus-augmented version for the cross validation case and the 'unseen answers' set. This is in contrast to the experiments reported by De Kuthy et al. (2016) using manual focus information, where the augmented system clearly outperforms all other variants. This shows that while focus information is clearly useful in Short Answer Assessment, it needs to be reliable enough to be of actual benefit. Recall also that the way we use focus information in CoMiC implies a strong commitment: only focused words are aligned and included in feature extraction, which does not produce the desired result if the focus information is not accurate. A possible way of remedying this situation would be to use focus as an extra feature or less strict modifier of existing features. There is thus room for improvement both in the automatic detection of focus and its use in extrinsic tasks.

However, one result stands out encouragingly: in the 'unseen questions' case, the focus-augmented version beats standard CoMiC, if only by a relatively small margin. This shows that even automatically determined information structural properties provide benefits when more concrete information, in the form of previously seen answers to the same questions, is not available. Our classifier thus successfully transfers general knowledge about focus to new question material.

9 Conclusion

We presented the first automatic focus detection approach for written data, and the first such approach for German. The approach uses a rich feature set including abstractions to grammatical notions (parts of speech, dependencies), word order aspects captured by a topological field model of German, an approximation of Givenness and the relation between material in the answer and that of the question word.

Using a word-by-word classification approach that takes into account both syntactic and semantic properties of answer and question words, we achieve an accuracy of 78.1% on a data set of 26,980 words in 10-fold cross validation. The focus detection pipeline developed for the experiment is freely available to other researchers.

Complementing the intrinsic evaluation, we

Test set	Instances	Majority baseline	CoMiC	+Focus
Oracle experiment reported by De Kuthy et al. (2016) on CREG-ExpertFocus				
leave-one-out	3187	51.0% *(correct)*	83.2%	85.6%
10-fold CV	1606	54.4% *(correct)*	**83.2%**	82.3%
Unseen answers	1002	51.3% *(correct)*	**80.6%**	80.5%
Unseen questions	1121	51.1% *(incorrect)*	77.4%	**78.4%**

Table 4: CoMiC results on different test sets using standard and focus-augmented features

provide an extrinsic evaluation of the approach as part of a larger CL task, the automatic content assessment of answers to reading comprehension questions. We show that while automatic focus detection does not yet improve content assessment for answers similar to the ones previously seen, it does provide a benefit in test cases where the questions and answers are completely new, i.e., where the system needs to generalize beyond the specific cases and contexts previously seen.

Contextualizing our work, one can see two different strands of research in the automatic analysis of focus. In comparison to Calhoun (2007) and follow-up approaches, who mainly concentrate on linking prosodic prominence to focus in dialogues, we do not limit our analysis to content words, but analyze every word of an utterance. This is made feasible due to the explicit task context we have in the form of answers to reading comprehension questions. We believe this nicely illustrates two avenues for obtaining relevant evidence on information structure: On the one hand, there is evidence obtained bottom-up through the data such as the rich information on prominence in spoken language data such as the corpus used by Calhoun (2007). On the other hand, there is top-down evidence from the task context, which sets up expectations about what is to be addressed for the current question under discussion. Following the QUD research strand, the approach presented in this paper could be scaled up beyond explicit question-answer pairs: De Kuthy et al. (2018) spell out an explicit analysis of text in terms of QUDs and show that it is possible to annotate explicit QUDs with high inter-annotator agreement. Combined with an automated approach to question generation, it could thus be possible to recover implicit QUDs from text and subsequently apply our current approach to any text, based on an independently established, general formal pragmatic analysis.

Finally, the qualitative analysis we exemplified is promising in terms of obtaining valuable insights to be addressed in future work. For example, the analysis identified faulty gaps in focus marking. In future work, integrating insights from theoretical linguistic approaches to focus and the notion of focus projection established there (cf., e.g., De Kuthy and Meurers 2012) could provide more guidance for ensuring contiguity of focus domains.

Acknowledgements

We would like to thank Kordula De Kuthy and the anonymous reviewers for detailed and helpful comments on different versions of this paper. This work has been funded by the Deutsche Forschungsgemeinschaft through Collaborative Research Center 833.

References

Leonardo Badino and Robert A. J. Clark. 2008. Automatic labeling of contrastive word pairs from spontaneous spoken English. In *Proceedings of the 2008 IEEE Spoken Language Technology Workshop*. pages 101–104. https://doi.org/10.1109/SLT.2008.4777850.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: A collection of very large linguistically processed web-crawled corpora. *Journal of Language Resources and Evaluation* 3(43):209–226. http://wacky.sslmit.unibo.it/lib/exe/fetch.php?media=papers:wacky_2008.pdf.

Aoife Cahill and Arndt Riester. 2012. Automatically acquiring fine-grained information status distinctions in german. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, pages 232–236. http://aclweb.org/anthology/W12-1632.

Sasha Calhoun. 2007. Predicting focus through prominence structure. In *Proceedings of Interspeech*. Antwerp, Belgium. http:

//www.cstr.inf.ed.ac.uk/downloads/publications/2007/calhounIS07.pdf.

Sasha Calhoun, Jean Carletta, Jason Brenier, Neil Mayo, Dan Jurafsky, Mark Steedman, and David Beaver. 2010. The NXT-format switchboard corpus: A rich resource for investigating the syntax, semantics, pragmatics and prosody of dialogue. *Language Resources and Evaluation* 44:387–419. http://link.springer.com/article/10.1007\%2Fs10579-010-9120-1.

Sasha Calhoun, Malvina Nissim, Mark Steedman, and Jason Brenier. 2005. A framework for annotating information structure in discourse. In *Proceedings of the Workshop on Frontiers in Corpus Annotations II: Pie in the Sky*. Association for Computational Linguistics, Ann Arbor, Michigan, pages 45–52. http://aclweb.org/anthology/W/W05/W05-0307.

Jackie Chi Kit Cheung and Gerald Penn. 2009. Topological field parsing of german. In *ACL-IJCNLP '09: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1*. Association for Computational Linguistics, Morristown, NJ, USA, pages 64–72. http://aclweb.org/anthology/P09-1008.

Kordula De Kuthy and Detmar Meurers. 2012. Focus projection between theory and evidence. In Sam Featherston and Britta Stolterfoht, editors, *Empirical Approaches to Linguistic Theory – Studies in Meaning and Structure*, De Gruyter, volume 111 of *Studies in Generative Grammar*, pages 207–240. http://purl.org/dm/papers/dekuthy-meurers-11.html.

Kordula De Kuthy, Nils Reiter, and Arndt Riester. 2018. Qud-based annotation of discourse structure and information structure: Tool and evaluation. In *Proceedings of the 11th Language Resources and Evaluation Conference*. Miyazaki, JP.

Kordula De Kuthy, Ramon Ziai, and Detmar Meurers. 2016. Focus annotation of task-based data: a comparison of expert and crowd-sourced annotation in a reading comprehension corpus. In *Proceedings of the 10th Edition of the Language Resources and Evaluation Conference (LREC)*. Portorož, Slovenia, pages 3928–3934. http://www.lrec-conf.org/proceedings/lrec2016/pdf/1083_Paper.pdf.

Stefanie Dipper, Michael Götze, and Stavros Skopeteas, editors. 2007. *Information Structure in Cross-Linguistic Corpora: Annotation Guidelines for Phonology, Morphology, Syntax, Semantics and Information Structure*, volume 7 of *Interdisciplinary Studies on Information Structure*. Universitätsverlag Potsdam, Potsdam, Germany. http://www.sfb632.uni-potsdam.de/publications/isis07.pdf.

Myroslava Dzikovska, Rodney Nielsen, Chris Brew, Claudia Leacock, Danilo Giampiccolo, Luisa Bentivogli, Peter Clark, Ido Dagan, and Hoa Trang Dang. 2013. Semeval-2013 task 7: The joint student response analysis and 8th recognizing textual entailment challenge. In *Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. Association for Computational Linguistics, Atlanta, Georgia, USA, pages 263–274. http://aclweb.org/anthology/S13-2045.

Kilian A. Foth, Arne Köhn, Niels Beuck, and Wolfgang Menzel. 2014. Because size does matter: The hamburg dependency treebank. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. European Language Resources Association (ELRA), Reykjavik, Iceland, pages 2326–2333. http://www.lrec-conf.org/proceedings/lrec2014/pdf/860_Paper.pdf.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA data mining software: An update. In *The SIGKDD Explorations*. volume 11, pages 10–18.

Michael Halliday. 1967. Notes on Transitivity and Theme in English. Part 1 and 2. *Journal of Linguistics* 3:37–81, 199–244.

Birgit Hamp and Helmut Feldweg. 1997. GermaNet – a lexical-semantic net for german. In *Proceedings of ACL workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*. Madrid. http://aclweb.org/anthology/W97-0802.

Christian F. Hempelmann, David Dufty, Philip M. McCarthy, Arthur C. Graesser, Zhiqiang Cai, and Danielle S. McNamara. 2005. Using LSA to automatically identify Givenness and Newness of noun phrases in written discourse. In B. G. Bara, L. Barsalou, and M. Bucciarelli, editors, *Proceedings of the 27th Annual Meeting of the Cognitive Science Society*. Erlbaum, Stresa, Italy, pages 941–949. https://doi.org/10.1.1.116.5716.

Tilman N. Höhle. 1986. Der Begriff 'Mittelfeld'. Anmerkungen über die Theorie der topologischen Felder. In A. Schöne, editor, *Kontroversen alte und neue. Akten des VII. Internationalen Germanistenkongresses Göttingen 1985*, Niemeyer, Tübingen, pages 329–340. Bd. 3.

Manfred Krifka and Renate Musan. 2012. Information structure: overview and linguistic issues. In Manfred Krifka and Renate Musan, editors, *The Expression of Information Structure*, De Gruyter Mouton, Berlin/Boston, volume 5 of *The Expression of Cognitive Categories*, pages 1–43.

Ivana Kruijff-Korbayová and Mark Steedman. 2003. Discourse and information structure. *Journal of Logic, Language and Information (Introduction to the Special Issue)* 12(3):249–259.

Adam Lally, John M. Prager, Michael C. McCord, Branimir K. Boguraev, Siddharth Patwardhan, James Fan, Paul Fodor, and Jennifer Chu-Carroll. 2012. Question analysis: How Watson reads a clue. *IBM Journal of Research and Development* 56(3/4):2:1–14.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*. Taipei, Taiwan, pages 1–7. http://aclweb.org/anthology/C02-1150.

Detmar Meurers, Ramon Ziai, Niels Ott, and Janina Kopp. 2011. Evaluating answers to reading comprehension questions in context: Results for German and the role of information structure. In *Proceedings of the TextInfer 2011 Workshop on Textual Entailment*. Edinburgh, pages 1–9. http://aclweb.org/anthology/W11-2401.pdf.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Malvina Nissim. 2006. Learning information status of discourse entities. In *Proceedings of the 2006 Conference on Emprical Methods in Natural Language Processing*. Sydney, Australia.

Malvina Nissim, Shipra Dingare, Jean Carletta, and Mark Steedman. 2004. An annotation scheme for information status in dialogue. In *Proceedings of the 4th Conference on Language Resources and Evaluation*. Lisbon, Portugal. http://www.lrec-conf.org/proceedings/lrec2004/pdf/638.pdf.

Joakim Nivre, Jens Nilsson, Johan Hall, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering* 13(1):1–41. http://w3.msi.vxu.se/~nivre/papers/nle07.pdf.

Niels Ott, Ramon Ziai, and Detmar Meurers. 2012. Creation and analysis of a reading comprehension exercise corpus: Towards evaluating meaning in context. In Thomas Schmidt and Kai Wörner, editors, *Multilingual Corpora and Multilingual Corpus Analysis*, Benjamins, Amsterdam, Hamburg Studies in Multilingualism (HSM), pages 47–69. https://benjamins.com/\#catalog/books/hsm.14.05ott.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*. Rochester, New York, pages 404–411.

Christopher Pinchak and Dekang Lin. 2006. A probabilistic answer type model. In *Proceedings of the 11th Conference of the European Chapter of the Association of Computational Lingustics (EACL)*. pages 393–400.

Arndt Riester, David Lorenz, and Nina Seemann. 2010. A recursive annotation scheme for referential information status. In *Proceedings of the 7th International Conference on Language Resources and Evaluation*. Valletta, Malta. http://www.lrec-conf.org/proceedings/lrec2010/pdf/764_Paper.pdf.

Stefan Riezler. 2014. On the problem of theoretical terms in empirical computational linguistics. *Computational Linguistics* 40(1):235–245.

Julia Ritz, Stefanie Dipper, and Michael Götze. 2008. Annotation of information structure: An evaluation across different types of texts. In *Proceedings of the 6th International Conference on Language Resources and Evaluation*. Marrakech, Morocco, pages 2137–2142. http://www.lrec-conf.org/proceedings/lrec2008/pdf/543_paper.pdf.

Craige Roberts. 2012. Information structure in discourse: Towards an integrated formal theory of pragmatics. *Semantics and Pragmatics* 5(6):1–69. https://doi.org/10.3765/sp.5.6.

Mats Rooth. 1992. A theory of focus interpretation. *Natural Language Semantics* 1(1):75–116.

Björn Rudzewitz. 2015. *Alignment Weighting for Short Answer Assessment*. Bachelor's thesis, University of Tübingen. www.sfs.uni-tuebingen.de/~brzdwtz/resources/BA_Thesis.pdf.

Björn Rudzewitz. 2016. Exploring the intersection of short answer assessment, authorship attribution, and plagiarism detection. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*. San Diego, CA, pages 235–241. https://aclweb.org/anthology/W16-0527.pdf.

Anne Schiller, Simone Teufel, and Christine Thielen. 1995. The Stuttgart-Tübingen Tagset (STTS). Technical report, Universität Stuttgart, Universität Tübingen, Germany. http://www.sfs.uni-tuebingen.de/Elwis/stts/stts.html.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*. Manchester, UK, pages 44–49. http://www.ims.uni-stuttgart.de/ftp/pub/corpora/tree-tagger1.pdf.

Vivek Kumar Rangarajan Sridhar, Ani Nenkova, Shrikanth Narayanan, and Dan Jurafsky. 2008. Detecting prominence in conversational speech: pitch accent, givenness and focus. In *Proceedings of Speech Prosody*. Campinas, Brazil, pages 380–388.

Arnim von Stechow. 1991. Focusing and backgrounding operators. In W. Abraham, editor, *Discourse Particles*, John Benjamins Publishing Co., Amsterdam/Philadelphia, pages 37–84.

Manfred Stede. 2012. Computation and modeling of information structure. In Manfred Krifka and Renate Musan, editors, *The Expression of Information Structure*, De Gruyter Mouton, Berlin/Boston, volume 5 of *The Expression of Cognitive Categories*, pages 363–408.

Heike Telljohann, Erhard Hinrichs, and Sandra Kübler. 2004. The TüBa-D/Z treebank: Annotating German with a context-free backbone. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*. Lissabon.

Peter Turney. 2001. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the Twelfth European Conference on Machine Learning (ECML-2001)*. Freiburg, Germany, pages 491–502.

Ramon Ziai, Kordula De Kuthy, and Detmar Meurers. 2016. Approximating Givenness in Content Assessment through Distributional Semantics. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics (*SEM)*. ACL, Berlin, Germany, pages 209–218. `http://aclweb.org/anthology/S16-2026.pdf`.

Ramon Ziai and Detmar Meurers. 2014. Focus annotation in reading comprehension data. In *Proceedings of the 8th Linguistic Annotation Workshop (LAW VIII, 2014)*. COLING, ACL, Dublin, Ireland, pages 159–168. `http://aclweb.org/anthology/W14-4922.pdf`.

Dear Sir or Madam, May I Introduce the GYAFC Dataset: Corpus, Benchmarks and Metrics for Formality Style Transfer

Sudha Rao
University of Maryland, College Park*
raosudha@cs.umd.edu

Joel Tetreault
Grammarly
joel.tetreault@grammarly.com

Abstract

Style transfer is the task of automatically transforming a piece of text in one particular style into another. A major barrier to progress in this field has been a lack of training and evaluation datasets, as well as benchmarks and automatic metrics. In this work, we create the largest corpus for a particular stylistic transfer (formality) and show that techniques from the machine translation community can serve as strong baselines for future work. We also discuss challenges of using automatic metrics.

1 Introduction

One key aspect of *effective communication* is the accurate expression of the style or tone of some content. For example, writing a more *persuasive email* in a marketing position could lead to increased sales; writing a more *formal email* when applying for a job could lead to an offer; and writing a more *polite note* to your future spouse's parents, may put you in a good light. Hovy (1987) argues that by varying the style of a text, people convey more information than is present in the literal meaning of the words. One particularly important dimension of style is formality (Heylighen and Dewaele, 1999). Automatically changing the style of a given content to make it more formal can be a useful addition to any writing assistance tool.

In the field of style transfer, to date, the only available dataset has been for the transformation of modern English to Shakespeare, and it led to the application of phrase-based machine translation (PBMT) (Xu et al., 2012) and neural machine translation (NMT) (Jhamtani et al., 2017) models to the task. The lack of an equivalent or larger dataset for any other form of style transfer has blocked progress in this field. Moreover, prior work has mainly borrowed metrics from machine translation (MT) and paraphrase communities for evaluating style transfer. However, it is not clear if those metrics are the best ones to use for this task. In this work, we address these issues through the following three contributions:

- **Corpus:** We present Grammarly's Yahoo Answers Formality Corpus (GYAFC), the largest dataset for any style containing a total of 110K informal / formal sentence pairs. Table 1 shows sample sentence pairs.
- **Benchmarks:** We introduce a set of learning models for the task of formality style transfer. Inspired by work in low resource MT, we adapt existing PBMT and NMT approaches for our task and show that they can serve as strong benchmarks for future work.
- **Metrics:** In addition to MT and paraphrase metrics, we evaluate our models along three axes: *formality*, *fluency* and *meaning preservation* using existing automatic metrics. We compare these metrics with their human judgments and show there is much room for further improvement.

Informal:	*I'd say it is punk though.*
Formal:	*However, I do believe it to be punk.*
Informal:	*Gotta see both sides of the story.*
Formal:	*You have to consider both sides of the story.*

Table 1: Informal sentences with formal rewrites.

In this paper, we primarily focus on the *informal* to *formal* direction since we collect our dataset for this direction. However, we evaluate our models on the *formal* to *informal* direction as well.[1] All data, model outputs, and evaluation results have been made public[2] in the hope that they will encourage more research into style transfer.

*This research was performed when the first author was at Grammarly.

[1] Results are in the supplementary material.

[2] https://github.com/raosudha89/GYAFC-corpus

Proceedings of NAACL-HLT 2018, pages 129–140
New Orleans, Louisiana, June 1 - 6, 2018.

In the following two sections we discuss related work and the GYAFC dataset. In §4, we detail our rule-based and MT-based approaches. In §5, we describe our human and automatic metric based evaluation. In §6, we describe the results of our models using both human and automatic evaluation and discuss how well the automatic metrics correlate with human judgments.

2 Related Work

Style Transfer with Parallel Data: Sheikha and Inkpen (2011) collect pairs of formal and informal words and phrases from different sources and use a natural language generation system to generate informal and formal texts by replacing lexical items based on user preferences. Xu et al. (2012) (henceforth XU12) was one of the first works to treat style transfer as a sequence to sequence task. They generate a parallel corpus of 30K sentence pairs by scraping the modern translations of Shakespeare plays and train a PBMT system to translate from modern English to Shakespearean English.[3] More recently, Jhamtani et al. (2017) show that a copy-mechanism enriched sequence-to-sequence neural model outperforms XU12 on the same set. In text simplification, the availability of parallel data extracted from English Wikipedia and Simple Wikipedia (Zhu et al., 2010) led to the application of PBMT (Wubben et al., 2012a) and more recently NMT (Wang et al., 2016) models. We take inspiration from both the PBMT and NMT models and apply several modifications to these approaches for our task of transforming the formality style of the text.

Style Transfer without Parallel Data: Another direction of research directly controls certain attributes of the generated text *without* using parallel data. Hu et al. (2017) control the sentiment and the tense of the generated text by learning a disentangled latent representation in a neural generative model. Ficler and Goldberg (2017) control several linguistic style aspects simultaneously by conditioning a recurrent neural network language model on specific style (professional, personal, length) and content (theme, sentiment) parameters. Under NMT models, Sennrich et al. (2016a) control the politeness of the translated text via side constraints, Niu et al. (2017) control the level of formality of MT output by selecting phrases of a requisite formality level from the k-best list during decoding. In the field of text simplification, more recently, Xu et al. (2016) learn large-scale paraphrase rules using bilingual texts whereas Kajiwara and Komachi (2016) build a monolingual parallel corpus using sentence similarity based on alignment between word embeddings. Our work differs from these methods in that we mainly address the question of how much leverage we can derive by collecting a large amount of informal-formal sentence pairs and build models that learn to transfer style directly using this parallel corpus.

Identifying Formality: There has been previous work on detecting formality of a given text at the lexical level (Brooke et al., 2010; Lahiri et al., 2011; Brooke and Hirst, 2014; Pavlick and Nenkova, 2015), at the sentence level (Pavlick and Tetreault, 2016) and at the document level (Sheikha and Inkpen, 2010; Peterson et al., 2011; Mosquera and Moreda, 2012). In our work, we reproduce the sentence-level formality classifier introduced in Pavlick and Tetreault (2016) (PT16) to extract informal sentences for GYAFC creation and to automatically evaluate system outputs.

Evaluating Style Transfer: The problem of style transfer falls under the category of natural language generation tasks such as machine translation, paraphrasing, etc. Previous work on style transfer (Xu et al., 2012; Jhamtani et al., 2017; Niu et al., 2017; Sennrich et al., 2016a) has re-purposed the MT metric BLEU (Papineni et al., 2002) and the paraphrase metric PINC (Chen and Dolan, 2011) for evaluation. Additionally, XU12 introduce three new automatic style metrics based on cosine similarity, language model and logistic regression that measure the degree to which the output matches the target style. Under human based evaluation, on the other hand, there has been work on a more fine grained evaluation where human judgments were separately collected for adequacy, fluency and style (Xu et al., 2012; Niu et al., 2017). In our work, we conduct a more thorough evaluation where we evaluate model outputs on the three criteria of *formality*, *fluency* and *meaning* using both automatic metrics and human judgments.

[3] https://github.com/cocoxu/Shakespeare

Domain	Total	Informal	Formal
All Yahoo Answers	40M	24M	16M
Entertainment & Music	3.8M	2.7M	700K
Family & Relationships	7.8M	5.6M	1.8M

Table 2: Yahoo Answers corpus statistics

		Informal to Formal		*Formal to Informal*	
	Train	Tune	Test	Tune	Test
E&M	52,595	2,877	1,416	2,356	1,082
F&R	51,967	2,788	1,332	2,247	1,019

Table 3: GYAFC dataset statistics

3 GYAFC **Dataset**

3.1 Creation Process

Yahoo Answers,[4] a question answering forum, contains a large number of informal sentences and allows redistribution of data. Hence, we use the Yahoo Answers L6 corpus[5] to create our GYAFC dataset of informal and formal sentence pairs. In order to ensure a uniform distribution of data, we remove sentences that are questions, contain URLs, and are shorter than 5 words or longer than 25. After these preprocessing steps, 40 million sentences remain. The Yahoo Answers corpus consists of several different domains like *Business, Entertainment & Music, Travel, Food,* etc. PT16 show that the formality level varies significantly across different genres. In order to control for this variation, we work with two specific domains that contain the most informal sentences and show results on training and testing within those categories. We use the formality classifier from PT16 to identify informal sentences. We train this classifier on the *Answers* genre of the PT16 corpus which consists of nearly 5,000 randomly selected sentences from Yahoo Answers manually annotated on a scale of -3 (very informal) to 3 (very formal).[6] We find that the domains of *Entertainment & Music* and *Family & Relationships* contain the most informal sentences and create our GYAFC dataset using these domains. Table 2 shows the number of formal and informal sentences in all of Yahoo Answers corpus and within the two selected domains. Sentences with a score less than 0 are considered as informal and sentences with a score greater than 0 are considered as formal.

Next, we randomly sample a subset of 53,000 informal sentences each from the *Entertainment & Music* (E&M) and *Family & Relationships* (F&R) categories and collect one formal rewrite per sentence using Amazon Mechanical Turk. The workers are presented with detailed instructions, as well as examples. To ensure quality control, four experts, two of which are the authors of this paper, reviewed the rewrites of the workers and rejected those that they felt did not meet the required standards. They also provided the workers with reasons for rejection so that they would not repeat the same mistakes. Any worker who repeatedly performed poorly was eventually blocked from doing the task. We use this train set to train our models for the style transfer tasks in both directions.

Since we want our tune and test sets to be of higher quality compared to the train set, we recruit a set of 85 expert workers for this annotation who had a 100% acceptance rate for our task and who had previously done more than 100 rewrites. Further, we collect multiple references for the tune/test set to adapt PBMT tuning and evaluation techniques to our task. We collect four different rewrites per sentence using our expert workers by randomly assigning sentences to the experts until four rewrites for each sentence are obtained.[7] To create our tune and test sets for the *informal* to *formal* direction, we sample an additional 3,000 informal sentences for our tune set and 1,500 sentences for our test set from each of the two domains.

To create our tune and test sets for the *formal* to *informal* direction, we start with the same tune and test split as the first direction. For each formal rewrite[8] from the first direction, we collect three different informal rewrites using our expert workers as before. These three informal rewrites along with the original informal sentence become our set of four references for this direction of the task. Table 3 shows the exact number of sentences in our train, tune and test sets.

3.2 Analysis

The following quantitative and qualitative analyses are aimed at characterizing the changes between the original informal sentence and its formal

[4] https://answers.yahoo.com/answer

[5] https://webscope.sandbox.yahoo.com/catalog.php?datatype=l

[6] http://www.seas.upenn.edu/~nlp/resources/formality-corpus.tgz

[7] Thus, note that the four rewrites are not from the same four workers for each sentence

[8] Out of four, we pick the one with the most edit distance with the original informal. Rationale explained in Section 3.2

rewrite in the GYAFC train split.[9] We present our analysis here on only the E&M domain data since we observe similar patterns in F&R.

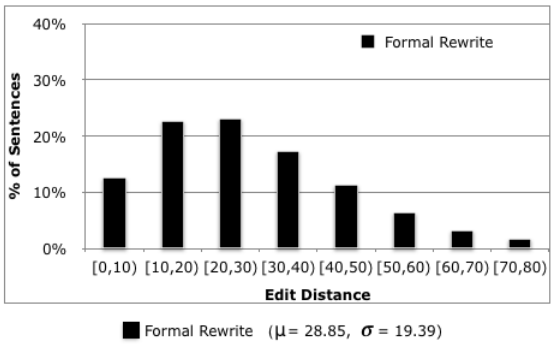


Figure 1: Percentage of sentences binned according to formality score in train set of E&M.

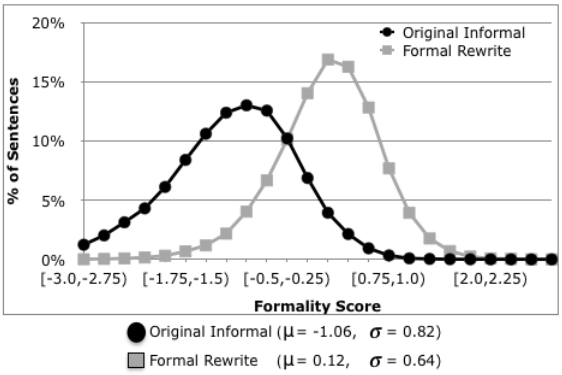


Figure 2: Percentage of sentences binned according to formality score in train set of E&M

Quantitative Analysis: While rewriting sentences more formally, humans tend to make a wide range of lexical/character-level edits. In Figure 1, we plot the distribution of the character-level Levenshtein edit distance between the original informal and the formal rewrites in the train set and observe a standard deviation of $\sigma = 19.39$ with a mean $\mu = 28.85$. Next, we look at the difference in the formality level of the original informal and the formal rewrites in GYAFC. We find that the classifier trained on the *Answers* genre of PT16 dataset correlates poorly (Spearman $\rho = 0.38$) with human judgments when tested on our domain specific datasets. Hence, we collect formality judgments on a scale of -3 to +1, similar to PT16, for an additional 5000 sentences each from both domains and obtain a formality classifier with higher correlation (Spearman $\rho = 0.56$). We use this retrained classifier for our evaluation in §5 as well.

In Figure 2, we plot the distribution of the formality scores on the original informal sentence and their formal rewrites in the train set and observe an increase in the mean formality score as we go from informal (-1.06) to formal rewrites (0.12). As compared to edit distance and formality, we observe a much lower variation in sentence lengths with the mean slightly increasing from informal (11.93) to their formal rewrites (12.56) in the train set.

Qualitative Analysis: To understand what stylistic choices differentiate formal from informal text, we perform an analysis similar to PT16 and look at 50 rewrites from both domains and record the frequency of the types of edits that workers made when creating a more formal sentence.[10] In contrast to PT16, we observe a higher percentage of phrasal paraphrases (47%), edits to punctuations (40%) and expansion of contractions (12%). This is reflective of our sentences coming from very informal domains of Yahoo Answers. Similar to PT16, we also observe capitalization (46%) and normalization (10%).

4 Models

We experiment with three main classes of approaches: a rule-based approach, PBMT and NMT. Inspired by work in low resource machine translation, we apply several modifications to the standard PBMT and NMT models and create a set of strong benchmarks for the style transfer community. We apply these models to both directions of style transfer: *informal* to *formal* and *formal* to *informal*. In our description, we refer to the two styles as *source* and *target*. We summarize the models below and direct the reader to supplementary material for further detail.

4.1 Rule-based Approach

Corresponding to the category of edits described in §3.2, we develop a set of rules to automatically make an informal sentence more formal where we capitalize first word and proper nouns, remove repeated punctuations, handcraft a list of expansion for contractions etc. For the *formal* to *informal* direction, we design a similar set of rules in the opposite direction.

[9] We observe similar patterns on the tune and test set.

[10] Examples of edits in supplementary material.

4.2 Phrase-based Machine Translation

Phrased-based machine translation models have had success in the fields of machine translation, style transfer (XU12) and text simplification (Wubben et al., 2012b; Xu et al., 2016). Inspired by work in low resource machine translation, we use a combination of training regimes to develop our model. We train on the output of the rule-based approach when applied to GYAFC. This is meant to force the PBMT model to learn generalizations *outside* the rules. To increase the data size, we use self-training (Ueffing, 2006), where we use the PBMT model to translate the large number of in-domain sentences from GYAFC belonging to the the source style and use the resultant output to retrain the PBMT model. Using subselection, we only select rewrites that have an Levenshtein edit distance of over 10 characters when compared to the source to encourage the model to be less conservative. Finally, we upweight the rule-based GYAFC data via duplication (Sennrich et al., 2016b). For our experiments, we use Moses (Koehn et al., 2007). We train a 5-gram language model using KenLM (Heafield et al., 2013), and use target style sentences from GYAFC and the sub-sampled target style sentences from out-of-domain Yahoo Answers, as in Moore and Lewis (2010), to create a large language model.

4.3 Neural Machine Translation

While encoder-decoder based neural network models have become quite successful for MT(Sutskever et al., 2014; Bahdanau et al., 2014; Cho et al., 2014), the field of style transfer, has not yet been able to fully take advantage of these advances owing to the lack of availability of large parallel data. With GYAFC we can now show how well NMT techniques fare for style transfer. We experiment with three NMT models:

NMT baseline: Our baseline model is a bi-directional LSTM (Hochreiter and Schmidhuber, 1997) encoder-decoder model with attention (Bahdanau et al., 2014).[11] We pretrain the input word embeddings on Yahoo Answers using GloVE (Pennington et al., 2014). As in our PBMT based approach, we train our NMT baseline model on the output of the rule-based approach when applied to GYAFC.

NMT Copy: Jhamtani et al., (2017) introduce a copy-enriched NMT model for style transfer to better handle stretches of text which should not be changed. We incorporate this mechanism into our NMT Baseline.

NMT Combined: The size of our parallel data is smaller than the size typically used to train NMT models. Motivated by this fact, we propose several variants to the baseline models that we find helps minimize this issue. We augment the data used to train NMT Copy via two techniques: 1) we run the PBMT model on additional source data, and 2) we use back-translation (Sennrich et al., 2016c) of the PBMT model to translate the large number of in-domain target style sentences from GYAFC. To balance the over one million artificially generated pairs from the respective techniques, we upweight the rule-based GYAFC data via duplication.[12]

5 Evaluation

As discussed earlier, there has been very little research into best practices for style transfer evaluation. Only a few works have included a human evaluation (Xu et al., 2012; Jhamtani et al., 2017), and automatic evaluations have employed BLEU or PINC (Xu et al., 2012; Chen and Dolan, 2011), which have been borrowed from other fields and not vetted for this task. In our work, we conduct a more thorough and detailed evaluation using both humans and automatic metrics to assess transformations. Inspired by work in the paraphrase community (Callison-Burch, 2008), we solicit ratings on how formal, how fluent and how meaning-preserving a rewrite is. Additionally, we look at the correlation between the human judgments and the automatic metrics.

5.1 Human-based Evaluation

We perform human-based evaluation to assess model outputs on the four criteria: *formality*, *fluency*, *meaning* and *overall*. For a subset of 500 sentences from the test sets of both *Entertainment & Music* and *Family & Relationship* domains, we collect five human judgments per sentence per criteria using Amazon Mechanical Turk as follows:

[11] Details are in the supplementary material.

[12] Training data sizes for different methods are summarized in the supplementary material.

Formality: Following PT16, workers rate the formality of the source style sentence, the target style reference rewrite and the target style model outputs on a discrete scale of -3 to +3 described as: *-3: Very Informal, -2: Informal, -1: Somewhat Informal, 0: Neutral, 1: Somewhat Formal, 2: Formal and 3: Very Formal*.

Fluency: Following Heilman et al. (2014), workers rate the fluency of the source style sentence, the target style reference rewrite and the target style model outputs on a discrete scale of 1 to 5 described as: *5: Perfect, 4: Comprehensible, 3: Somewhat Comprehensible, 2: Incomprehensible*. We additionally provide an option of *1: Other* for sentences that are incomplete or just a fragment.

Meaning Preservation: Following the annotation scheme developed for the Semantic Textual Similarity (STS) dataset (Agirre et al., 2016), given two sentences i.e. the source style sentence and the target style reference rewrite or the target style model output, workers rate the meaning similarity of the two sentences on a scale of 1 to 6 described as: *6: Completely equivalent, 5: Mostly equivalent, 4: Roughly equivalent, 3: Not equivalent but share some details, 2: Not equivalent but on same topic, 1: Completely dissimilar*.

Overall Ranking: In addition to the fine-grained human judgments, we collect judgments to assess the overall ranking of the systems. Given the original source style sentence, the target style reference rewrite and the target style model outputs, we ask workers to rank the rewrites in the order of their overall formality, taking into account both fluency and meaning preservation. We then rank the model using the equation below:

$$rank(model) = \frac{1}{|S|} \sum_{s \in S} \frac{1}{|J|} \sum_{j \in J} rank(s_{model}, j) \quad (1)$$

where, $model$ is the one of our models, S is a subset of 500 test set sentences, J is the set of five judgments, s_{model} is the model rewrite for sentence s, and $rank(s_{model}, j)$ is the rank of s_{model} in judgment j.

The two authors of the paper reviewed these human judgments and found that in majority of the cases the annotations looked correct. But as is common in any such crowdsourced data collection process, there were some errors, especially in the overall ranking of the systems.

5.2 Automatic Metrics

We cover each of the human evaluations with a corresponding automatic metric:

Formality: We use the formality classifier described in PT16. We find that the classifier trained on the *answers* genre of PT16 dataset does not perform well when tested on our datasets. Hence, we collect formality judgments for an additional 5000 sentences and use the formality classifier re-trained on this in-domain data.

Fluency: We use the reimplementation[13] of Heilman et al. (2014) (H14 in Table 4) which is a statistical model for predicting the grammaticality of a sentence on a scale of 0 to 4 previously shown to be effective for other generation tasks like grammatical error correction (Napoles et al., 2016).

Meaning Preservation: Modeling semantic similarity at a sentence level is a fundamental language processing task, and one that is a wide open field of research. Recently, He et al., (2015) (HE15 in Table 4) developed a convolutional neural network based sentence similarity measure. We use their off-the-shelf implementation[14] to train a model on the STS and use it to measure the meaning similarity between the original source style sentence and its target style rewrite (both reference and model outputs).

Overall Ranking: We experiment with BLEU (Papineni et al., 2002) and PINC (Chen and Dolan, 2011) as both were used in prior style evaluations, as well as TERp (Snover et al., 2009).

6 Results

In this section, we discuss how well the five models perform in the *informal* to *formal* style transfer task using human judgments (§6.1) and automatic metrics (§6.2), the correlation of the automatic metrics and human judgments to determine the ef-

[13] https://github.com/cnap/grammaticality-metrics/tree/master/heilman-et-al

[14] https://github.com/castorini/MP-CNN-Torch

Model	Formality		Fluency		Meaning		Combined		Overall		
	Human	PT16	Human	H14	Human	HE15	Human	Auto	BLEU	TERp	PINC
Original Informal	*-1.23*	*-1.00*	*3.90*	*2.89*	–	–	–	–	*50.69*	*0.35*	*0.00*
Formal Reference	0.38	0.17	4.45	3.32	4.57	3.64	5.68	4.67	100.0	0.37	69.79
Rule-based	-0.59	-0.34	4.00	3.09	**4.85**	**4.41**	5.24	4.69	61.38	0.27	26.05
PBMT	-0.19*	0.00*	3.96	3.28*	4.64*	4.19*	5.27	4.82*	67.26*	**0.26**	44.94*
NMT Baseline	**0.05***	0.07*	4.05	**3.52***	3.55*	3.89*	4.96*	**4.84***	56.61	0.38*	**56.92***
NMT Copy	0.02*	**0.10***	4.07	3.45*	3.48*	3.87*	4.93*	4.81*	58.01	0.38*	56.39*
NMT Combined	-0.16*	0.00*	**4.09***	3.27*	4.46*	4.20*	**5.32***	4.82*	**67.67***	**0.26**	43.54*

Table 4: Results of models on 500 test sentences from E&M for *informal* to *formal* task evaluated using human judgments and automatic metrics for three criteria of evaluation: formality, fluency and meaning preservation. Scores marked with * are significantly different from the rule-based scores with $p < 0.001$.

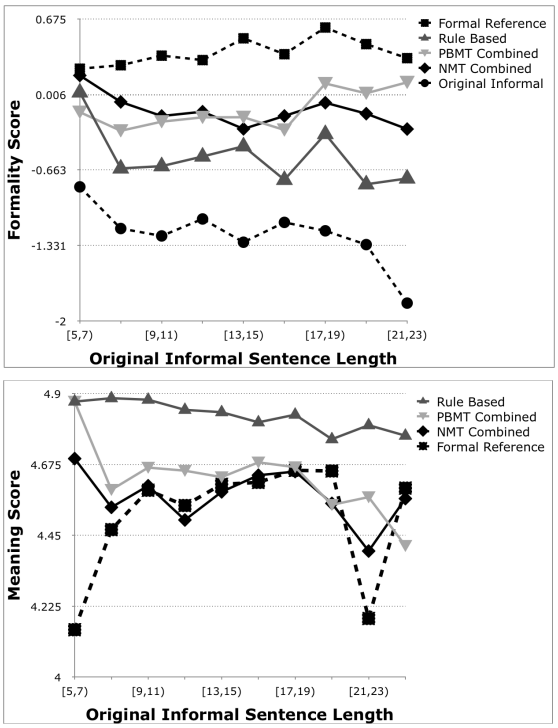


Figure 3: For varying sentence lengths of the original informal sentence the *formality* and the *meaning* scores from human judgments on different model outputs and on the original informal and the formal reference sentences.

ficacy of the metrics (§6.3) and present a manual analysis (§6.4). We randomly select 500 sentences from each test set and run all five models. We use the entire train and tune split for training and tuning. We discuss results only on the E&M domain and list results on the F&R domain in the supplementary material.

Table 4 shows the results for human §6.1 and automatic §6.2 evaluation of model rewrites. For all metrics except *TERp*, a higher score is better. For each of the automatic metrics, we evaluate against four human references. The row *'Original Informal'* contains the scores when the original informal sentence is compared with the four formal reference rewrites. Comparing the model scores to this score helps us understand how closer are the model outputs to the formal reference rewrites compared to initial distance between the informal and the formal reference rewrite.

6.1 Results using Human Judgments

The columns marked *'Human'* in Table 4 show the human judgments for the models on the three separate criteria of *formality*, *fluency* and *meaning* collected using the process described in Section 5.1.[15] The NMT Baseline and Copy models beat others on the formality axis by a significant margin. Only the NMT Combined model achieves a statistically higher fluency score when compared to the rule-based baseline model. As expected, the rule-based model is the most meaning preserving since it is the most conservative. Figure 3 shows the trend in the four leading models along *formality* and *meaning* for varying lengths of the source sentence. NMT Combined beats PBMT on *formality* for shorter lengths whereas the trend reverses as the length increases. PBMT generally preserves meaning more than the NMT Combined. We find that the fluency scores for all models decreases as the sentence length increases which is similar to the trend generally observed with machine translation based approaches.

Since a good style transfer model is the one that attains a balanced score across all the three axes, we evaluate the models on a combination of these metrics[16] shown under the column *'Combined'* in Table 4. NMT Combined is the only model having a combined score statistically greater than the rule-based approach.

[15] Out of the four reference rewrites, we pick one at random to show to Turkers.

[16] We recalibrate the scores to normalize for different ranges.

Finally, Table 5 shows the overall rankings of the models from best to worst in both domains. PBMT and NMT Combined models beat the rule-based model although not significantly in the E&M domain but significantly in the F&R domain. Interestingly, the rule-based approach attains third place with a score significantly higher than NMT Copy and NMT Baseline models. It is important to note here that while such a rule-based approach is relatively easy to craft for the formality style transfer task, the same may not be true for other styles like politeness or persuasiveness.

E&M	F&R
(2.03*) Reference	(2.13*) Reference
(2.47) PBMT	(2.38*) PBMT
(2.48) NMT Combined	(2.38*) NMT Combined
(2.54) Rule-based	(2.56) Rule-based
(3.03*) NMT Copy	(2.72*) NMT Copy
(3.03*) NMT Baseline	(2.79*) NMT Baseline

Table 5: Ranking of different models on the *informal* to *formal* style transfer task. Rankings marked with * are significantly different from the rule-based ranking with $p < 0.001$.

Automatic	**Human**	**E&M**	**F&R**
Formality	*Formality*	0.47	0.45
Fluency	*Fluency*	0.48	0.46
Meaning	*Meaning*	0.33	0.30
BLEU	*Overall*	-0.48	-0.43
TERp	*Overall*	0.31	0.30
PINC	*Overall*	0.11	0.08

Table 6: Spearman rank correlation between automatic metrics and human judgments. The first three metrics are correlated with their respective human judgments and the last three metrics are correlated with the *overall ranking* human judgments. All correlations are statistically significant with $p < 0.001$.

6.2 Results with Automatic Metrics

Under automatic metrics, the formality and meaning scores align with the human judgments with the NMT Baseline and NMT Copy winning on formality and rule-based winning on meaning. The fluency score of the NMT Baseline is the highest in contrast to human judgments where the NMT Combined wins. This discrepancy could be due to H14 being trained on *essays* which contains sentences of a more formal genre compared to Yahoo Answers. In fact, the fluency classifier scores the formal reference quite low as well. Under overall metrics, PBMT and NMT Combined models beat other models as per BLEU (significantly) and TERp (not significantly). NMT Baseline and NMT copy win over other models as per PINC which can be explained by the fact that PINC measures lexical dissimilarity with the source and NMT models tend towards making more changes. Although such an analysis is useful, for a more thorough understanding of these metrics, we next look at their correlation with human judgments.

6.3 Metric Correlation

We report the spearman rank correlation coefficient between automatic metrics and human judgments in Table 6. For *formality*, *fluency* and *meaning*, the correlation is with their respective human judgments whereas for BLEU, TERp and PINC, the correlation is with the overall ranking.

We see that the formality and the fluency metrics correlate moderately well while the meaning metric correlates comparatively poorly. To be fair, the HE15 classifier was trained on the STS dataset which contains more formal writing than informal. BLEU correlates moderately well (better than what XU12 observed for the Shakespeare task) whereas the correlation drops for TERp. PINC, on the other hand, correlates very poorly with a positive correlation with rank when it should have a negative correlation with rank, just like BLEU. This sheds light on the fact that PINC, on its own, is not a good metric for style transfer since it prefers lexical edits at the cost of meaning changes. In the Shakespeare task, XU12 did observe a higher correlation with PINC (0.41) although the correlation was not with overall system ranking but rather only on the style metric. Moreover, in the Shakespeare task, changing the text is more favorable than in formality.

6.4 Manual Analysis

The prior evaluations reveal the relative performance differences between approaches. Here, we identify trends per and between approaches. We sample 50 informal sentences total from both domains and then analyze the outputs from each model. We present sample sentences in Table 7.

The NMT Baseline and NMT Copy tend to have the most variance in their performance. This is likely due to the fact that they are trained on only 50K sentence pairs, whereas the other models are trained on much more data. For shorter sentences, these models make some nice formal transformations like from '*very dumb*' to '*very foolish*'. However, for longer sentences, these models make drastic meaning changes and drop some content altogether (see examples in Table 7). On the

Entertainment & Music	
Original Informal	Wow , I am very dumb in my observation skills .....
Reference Formal	I do not have good observation skills .
Rule-based	Wow , I am very dumb in my observation skills .
PBMT	Wow , I am very dumb in my observation skills .
NMT Baseline	I am very foolish in my observation skills .
NMT Copy	Wow , I am very foolish in my observation skills .
NMT Combined	I am very unintelligent in my observation skills .
Family & Relationship	
Original Informal	i hardly everrr see him in school either usually i see hima t my brothers basketball games .
Reference Formal	I hardly ever see him in school . I usually see him with my brothers playing basketball .
Rule-based	I hardly everrr see him in school either usually I see hima t my brothers basketball games .
PBMT	I hardly see him in school as well, but my brothers basketball games .
NMT	I rarely see him in school , either I see him at my brother 's basketball games .
NMT Copy	I hardly see him in school either , usually I see him at my brother 's basketball games .
NMT Combined	I rarely see him in school either usually I see him at my brothers basketball games .

Table 7: Sample model outputs with references from both E&M and F&R domains on the *informal* to *formal* task

other hand, the PBMT and NMT Combined models have lower variance in their performance. They make changes more conservatively but when they do, they are usually correct. Thus, most of the outputs from these two models are usually meaning preserving but at the expense of a lower formality score improvement.

In most examples, all models are good at removing very informal words like '*stupid*', '*idiot*' and '*hell*', with PBMT and NMT Combined models doing slightly better. All models struggle when the original sentence is very informal or disfluent. They all also struggle with sentence completions that humans seem to be very good at. This might be because humans assume a context when absent, whereas the models do not. Unknown tokens, either real words or misspelled words, tend to wreak havoc on all approaches. In most cases, the models simply did not transform that section of the sentence, or remove the unknown tokens. Most models are effective at low-level changes such as writing out numbers, inserting commas, and removing common informal phrases.

7 Conclusions and Future Work

The goal of this paper was to move the field of style transfer forward by creating a large training and evaluation corpus to be made public, showing that adapting MT techniques to this task can serve as strong baselines for future work, and analyzing the usefulness of existing metrics for overall style transfer as well as three specific criteria of automatic style transfer evaluation. We view this work as rigorously expanding on the foundation set by XU12 five years earlier. It is our hope that with a common test set, the field can finally benchmark approaches which do not require parallel data.

We found that while the NMT systems perform well given automatic metrics, humans had a slight preference for the PBMT approach. That being said, two of the neural approaches (NMT Baseline and Copy) often made successful changes and larger rewrites that the other models could not. However, this often came at the expense of a meaning change.

We also introduced new metrics and vetted all metrics using comparison with human judgments. We found that previously-used metrics did not correlate well with human judgments, and thus should be avoided in system development or final evaluation. The formality and fluency metrics correlated best and we believe that some combination of these metrics with others would be the best next step in the development of style transfer metrics. Such a metric could then in turn be used to optimize MT models. Finally, in this work we focused on one particular style, formality. The long term goal is to generalize the methods and metrics to any style.

Acknowledgments

The authors would like to thank Yahoo Research for making their data available. The authors would also like to thank Junchao Zheng and Claudia Leacock for their help in the data creation process, Courtney Napoles for providing the fluency scores, Marcin Junczys-Dowmunt, Rico Sennrich, Ellie Pavlick, Maksym Bezva, Dimitrios Alikaniotis and Kyunghyun Cho for helpful discussion and the three anonymous reviewers for their useful comments and suggestions.

References

Eneko Agirre, Carmen Banea, Daniel M Cer, Mona T Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *SemEval@ NAACL-HLT*. pages 497–511.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Julian Brooke and Graeme Hirst. 2014. Supervised ranking of co-occurrence profiles for acquisition of continuous lexical attributes. In *COLING*. pages 2172–2183.

Julian Brooke, Tong Wang, and Graeme Hirst. 2010. Automatic acquisition of lexical formality. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. Association for Computational Linguistics, pages 90–98.

Chris Callison-Burch. 2008. Syntactic constraints on paraphrases extracted from parallel corpora. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Honolulu, Hawaii, pages 196–205. http://www.aclweb.org/anthology/D08-1021.

David L Chen and William B Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pages 190–200.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. *Syntax, Semantics and Structure in Statistical Translation* page 103.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. *Proceedings of the Workshop on Stylistic Variation, EMNLP 2017* .

Hua He, Kevin Gimpel, and Jimmy J Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *EMNLP*. pages 1576–1586.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Sofia, Bulgaria, pages 690–696. https://kheafield.com/papers/edinburgh/estimate_paper.pdf.

Michael Heilman, Aoife Cahill, Nitin Madnani, Melissa Lopez, Matthew Mulholland, and Joel Tetreault. 2014. Predicting grammaticality on an ordinal scale. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 174–180. http://www.aclweb.org/anthology/P14-2029.

Francis Heylighen and Jean-Marc Dewaele. 1999. Formality of language: definition, measurement and behavioral determinants. *Interner Bericht, Center Leo Apostel, Vrije Universiteit Brüssel* .

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

Eduard Hovy. 1987. Generating natural language under pragmatic constraints. *Journal of Pragmatics* 11(6):689–719.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *International Conference on Machine Learning*. pages 1587–1596.

Harsh Jhamtani, Varun Gangal, Eduard Hovy, and Eric Nyberg. 2017. Shakespearizing modern language using copy-enriched sequence-to-sequence models. *Proceedings of the Workshop on Stylistic Variation, EMNLP 2017* pages 10–19.

Tomoyuki Kajiwara and Mamoru Komachi. 2016. Building a monolingual parallel corpus for text simplification using sentence similarity based on alignment between word embeddings. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. pages 1147–1158.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*. Association for Computational Linguistics, pages 177–180.

Shibamouli Lahiri, Prasenjit Mitra, and Xiaofei Lu. 2011. Informality judgment at sentence level and experiments with formality score. In *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer, pages 446–457.

Robert C Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the ACL 2010 conference short papers*. Association for Computational Linguistics, pages 220–224.

Alejandro Mosquera and Paloma Moreda. 2012. Smile: An informality classification tool for helping to assess quality and credibility in web 2.0 texts. In *Proceedings of the ICWSM workshop: Real-Time Analysis and Mining of Social Streams (RAMSS)*.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2016. There's no comparison: Reference-less evaluation metrics in grammatical error correction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 2109–2115. https://aclweb.org/anthology/D16-1228.

Xing Niu, Marianna Martindale, and Marine Carpuat. 2017. A study of style in machine translation: Controlling the formality of machine translation output. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 2804–2809.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 311–318.

Ellie Pavlick and Ani Nenkova. 2015. Inducing lexical style properties for paraphrase and genre differentiation. In *HLT-NAACL*. pages 218–224.

Ellie Pavlick and Joel Tetreault. 2016. An empirical analysis of formality in online communication. *Transactions of the Association for Computational Linguistics* 4:61–74.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pages 1532–1543.

Kelly Peterson, Matt Hohensee, and Fei Xia. 2011. Email formality in the workplace: A case study on the enron corpus. In *Proceedings of the Workshop on Languages in Social Media*. Association for Computational Linguistics, pages 86–95.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Controlling politeness in neural machine translation via side constraints. In *HLT-NAACL*. pages 35–40.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Edinburgh neural machine translation systems for wmt 16. In *Proceedings of the First Conference on Machine Translation*. Association for Computational Linguistics, Berlin, Germany, pages 371–376. http://www.aclweb.org/anthology/W16-2323.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016c. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 86–96. https://doi.org/10.18653/v1/P16-1009.

Fadi Abu Sheikha and Diana Inkpen. 2010. Automatic classification of documents by formality. In *Natural Language Processing and Knowledge Engineering (NLP-KE), 2010 International Conference on*. IEEE, pages 1–5.

Fadi Abu Sheikha and Diana Inkpen. 2011. Generation of formal and informal sentences. In *Proceedings of the 13th European Workshop on Natural Language Generation*. Association for Computational Linguistics, pages 187–193.

Matthew Snover, Nitin Madnani, Bonnie J Dorr, and Richard Schwartz. 2009. Fluency, adequacy, or hter?: exploring different human judgments with a tunable mt metric. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, pages 259–268.

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

Nicola Ueffing. 2006. Self-training for machine translation. In *NIPS workshop on Machine Learning for Multilingual Information Access*.

Tong Wang, Ping Chen, John Rochford, and Jipeng Qiang. 2016. Text simplification using neural machine translation. In *AAAI*. pages 4270–4271.

Sander Wubben, Antal Van Den Bosch, and Emiel Krahmer. 2012a. Sentence simplification by monolingual machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics, pages 1015–1024.

Sander Wubben, Antal van den Bosch, and Emiel Krahmer. 2012b. Sentence simplification by monolingual machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Jeju Island, Korea, pages 1015–1024. http://www.aclweb.org/anthology/P12-1107.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics* 4:401–415.

Wei Xu, Alan Ritter, Bill Dolan, Ralph Grishman, and Colin Cherry. 2012. Paraphrasing for style. *Proceedings of COLING 2012* pages 2899–2914.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *Proceedings of the 23rd international conference on computational linguistics*. Association for Computational Linguistics, pages 1353–1361.

Improving Implicit Discourse Relation Classification by Modeling Inter-dependencies of Discourse Units in a Paragraph

Zeyu Dai, Ruihong Huang
Department of Computer Science and Engineering
Texas A&M University
{jzdaizeyu, huangrh}@tamu.edu

Abstract

We argue that semantic meanings of a sentence or clause can not be interpreted independently from the rest of a paragraph, or independently from all discourse relations and the overall paragraph-level discourse structure. With the goal of improving implicit discourse relation classification, we introduce a paragraph-level neural networks that model inter-dependencies between discourse units as well as discourse relation continuity and patterns, and predict a sequence of discourse relations in a paragraph. Experimental results show that our model outperforms the previous state-of-the-art systems on the benchmark corpus of PDTB.

1 Introduction

PDTB-style discourse relations, mostly defined between two adjacent text spans (i.e., discourse units, either clauses or sentences), specify how two discourse units are logically connected (e.g., causal, contrast). Recognizing discourse relations is one crucial step in discourse analysis and can be beneficial for many downstream NLP applications such as information extraction, machine translation and natural language generation.

Commonly, explicit discourse relations were distinguished from implicit ones, depending on whether a discourse connective (e.g., "because" and "after") appears between two discourse units (Prasad et al., 2008a). While explicit discourse relation detection can be framed as a discourse connective disambiguation problem (Pitler and Nenkova, 2009; Lin et al., 2014) and has achieved reasonable performance (F1 score $>$ 90%), implicit discourse relations have no discourse connective and are especially difficult to identify (Lin et al., 2009, 2014; Xue et al., 2015). To fill the gap, implicit discourse relation prediction has drawn significant research interest recently and progress has been made (Chen et al., 2016; Liu and Li, 2016) by modeling compositional meanings of two discourse units and exploiting word interactions between discourse units using neural tensor networks or attention mechanisms in neural nets. However, most of existing approaches ignore wider paragraph-level contexts beyond the two discourse units that are examined for predicting a discourse relation in between.

To further improve implicit discourse relation prediction, we aim to improve discourse unit representations by positioning a discourse unit (DU) in its wider context of a paragraph. The key observation is that semantic meaning of a DU can not be interpreted independently from the rest of the paragraph that contains it, or independently from the overall paragraph-level discourse structure that involve the DU. Considering the following paragraph with four discourse relations, one relation between each two adjacent DUs:

(1): *[The Butler, Wis., manufacturer went public at $15.75 a share in August 1987,]*$_{DU1}$ *and* ***(Explicit-Expansion)*** *[Mr. Sim's goal then was a $29 per-share price by 1992.]*$_{DU2}$ ***(Implicit-Expansion)*** *[Strong earnings growth helped achieve that price far ahead of schedule, in August 1988.]*$_{DU3}$ ***(Implicit-Comparison)*** *[The stock has since softened, trading around $25 a share last week and closing yesterday at $23 in national over-the-counter trading.]*$_{DU4}$ *But* ***(Explicit-Comparison)*** *[Mr. Sim has set a fresh target of $50 a share by the end of reaching that goal.]*$_{DU5}$

Clearly, each DU is an integral part of the paragraph and not independent from other units. *First*, predicting a discourse relation may require understanding wider paragraph-level contexts beyond two relevant DUs and the overall discourse structure of a paragraph. For example, the implicit "Comparison" discourse relation between DU3 and DU4 is difficult to identify without the back-

Proceedings of NAACL-HLT 2018, pages 141–151
New Orleans, Louisiana, June 1 - 6, 2018.

ground information (the history of per-share price) introduced in DU1 and DU2. *Second*, a DU may be involved in multiple discourse relations (e.g., DU4 is connected with both DU3 and DU5 with a "Comparison" relation), therefore the pragmatic meaning representation of a DU should reflect all the discourse relations the unit was involved in. *Third*, implicit discourse relation prediction should benefit from modeling discourse relation continuity and patterns in a paragraph that involve easy-to-identify explicit discourse relations (e.g., "Implicit-Comparison" relation is followed by "Explicit-Comparison" in the above example).

Following these observations, we construct a neural net model to process a paragraph each time and jointly build meaning representations for all DUs in the paragraph. The learned DU representations are used to predict a sequence of discourse relations in the paragraph, including both implicit and explicit relations. Although explicit relations are not our focus, predicting an explicit relation will help to reveal the pragmatic roles of its two DUs and reconstruct their representations, which will facilitate predicting neighboring implicit discourse relations that involve one of the DUs.

In addition, we introduce two novel designs to further improve discourse relation classification performance of our paragraph-level neural net model. First, previous work has indicated that recognizing explicit and implicit discourse relations requires different strategies, we therefore untie parameters in the discourse relation prediction layer of the neural networks and train two separate classifiers for predicting explicit and implicit discourse relations respectively. This unique design has improved both implicit and explicit discourse relation identification performance. Second, we add a CRF layer on top of the discourse relation prediction layer to fine-tune a sequence of predicted discourse relations by modeling discourse relation continuity and patterns in a paragraph.

Experimental results show that the intuitive paragraph-level discourse relation prediction model achieves improved performance on PDTB for both implicit discourse relation classification and explicit discourse relation classification.

2 Related Work

2.1 Implicit Discourse Relation Recognition

Since the PDTB (Prasad et al., 2008b) corpus was created, a surge of studies (Pitler et al., 2009; Lin et al., 2009; Liu et al., 2016; Rutherford and Xue, 2016) have been conducted for predicting discourse relations, primarily focusing on the challenging task of implicit discourse relation classification when no explicit discourse connective phrase was presented. Early studies (Pitler et al., 2008; Lin et al., 2009, 2014; Rutherford and Xue, 2015) focused on extracting linguistic and semantic features from two discourse units. Recent research (Zhang et al., 2015; Rutherford et al., 2016; Ji and Eisenstein, 2015; Ji et al., 2016) tried to model compositional meanings of two discourse units by exploiting interactions between words in two units with more and more complicated neural network models, including the ones using neural tensor (Chen et al., 2016; Qin et al., 2016; Lei et al., 2017) and attention mechanisms (Liu and Li, 2016; Lan et al., 2017; Zhou et al., 2016). Another trend is to alleviate the shortage of annotated data by leveraging related external data, such as explicit discourse relations in PDTB (Liu et al., 2016; Lan et al., 2017; Qin et al., 2017) and unlabeled data obtained elsewhere (Rutherford and Xue, 2015; Lan et al., 2017), often in a multi-task joint learning framework.

However, nearly all the previous works assume that a pair of discourse units is independent from its wider paragraph-level contexts and build their discourse relation prediction models based on *only* two relevant discourse units. In contrast, we model inter-dependencies of discourse units in a paragraph when building discourse unit representations; in addition, we model global continuity and patterns in a sequence of discourse relations, including both implicit and explicit relations.

Hierarchical neural network models (Liu and Lapata, 2017; Li et al., 2016) have been applied to RST-style discourse parsing (Carlson et al., 2003) mainly for the purpose of generating text-level hierarchical discourse structures. In contrast, we use hierarchical neural network models to build context-aware sentence representations in order to improve implicit discourse relation prediction.

2.2 Paragraph Encoding

Abstracting latent representations from a long sequence of words, such as a paragraph, is a challenging task. While several novel neural network models (Zhang et al., 2017b,a) have been introduced in recent years for encoding a paragraph, Recurrent Neural Network (RNN)-based

methods remain the most effective approaches. RNNs, especially the long-short term memory (LSTM) (Hochreiter and Schmidhuber, 1997) models, have been widely used to encode a paragraph for machine translation (Sutskever et al., 2014), dialogue systems (Serban et al., 2016) and text summarization (Nallapati et al., 2016) because of its ability in modeling long-distance dependencies between words. In addition, among four typical pooling methods (sum, mean, last and max) for calculating sentence representations from RNN-encoded hidden states for individual words, max-pooling along with bidirectional LSTM (Bi-LSTM) (Schuster and Paliwal, 1997) yields the current best universal sentence representation method (Conneau et al., 2017). We adopted a similar neural network architecture for paragraph encoding.

3 The Neural Network Model for Paragraph-level Discourse Relation Recognition

3.1 The Basic Model Architecture

Figure 1 illustrates the overall architecture of the discourse-level neural network model that consists of two Bi-LSTM layers, one max-pooling layer in between and one softmax prediction layer. The input of the neural network model is a paragraph containing a sequence of discourse units, while the output is a sequence of discourse relations with one relation between each pair of adjacent discourse units[1].

Given the words sequence of one paragraph as input, the lower Bi-LSTM layer will read the whole paragraph and calculate hidden states as word representations, and a max-pooling layer will be applied to abstract the representation of each discourse unit based on individual word representations. Then another Bi-LSTM layer will run over the sequence of discourse unit representations and compute new representations by further modeling semantic dependencies between discourse units within paragraph. The final softmax prediction layer will concatenate representations of two adjacent discourse units and predict the discourse relation between them.

Word Vectors as Input: The input of the paragraph-level discourse relation prediction model is a sequence of word vectors, one vector per word in the paragraph. In this work, we used the pre-trained 300-dimension Google English word2vec embeddings[2]. For each word that is not in the vocabulary of Google word2vec, we will randomly initialize a vector with each dimension sampled from the range $[-0.25, 0.25]$. In addition, recognizing key entities and discourse connective phrases is important for discourse relation recognition, therefore, we concatenate the raw word embeddings with extra linguistic features, specifically one-hot Part-Of-Speech tag embeddings and one-hot named entity tag embeddings[3].

Building Discourse Unit Representations: We aim to build discourse unit (DU) representations that sufficiently leverage cues for discourse relation prediction from paragraph-wide contexts, including the preceding and following discourse units in a paragraph. To process long paragraph-wide contexts, we take a bottom-up two-level abstraction approach and progressively generate a compositional representation of each word first (low level) and then generate a compositional representation of each discourse unit (high level), with a max-pooling operation in between. At both word-level and DU-level, we choose Bi-LSTM as our basic component for generating compositional representations, mainly considering its capability to capture long-distance dependencies between words (discourse units) and to incorporate influences of context words (discourse units) in each side.

Given a variable-length words sequence $X = (x_1, x_2, ..., x_L)$ in a paragraph, the word-level Bi-LSTM will process the input sequence by using two separate LSTMs, one process the word sequence from the left to right while the other follows the reversed direction. Therefore, at each word position t, we obtain two hidden states $\overrightarrow{h_t}, \overleftarrow{h_t}$. We concatenate them to get the word representation $h_t = [\overrightarrow{h_t}, \overleftarrow{h_t}]$. Then we apply max-pooling over the sequence of word representations for words in a discourse unit in order to get the discourse unit embedding:

[1] In PDTB, most of discourse relations were annotated between two adjacent sentences or two adjacent clauses. For exceptional cases, we applied heuristics to convert them.

[2] Downloaded from https://docs.google.com/uc?id=0B7XkCwpI5KDYNlNUTTlSS21pQmM

[3] Our feature-rich word embeddings are of dimension 343, including 300 dimensions for word2vec embeddings + 36 dimensions for Part-Of-Speech (POS) tags + 7 dimensions for named entity tags. We used the Stanford CoreNLP to generate POS tags and named entity tags.

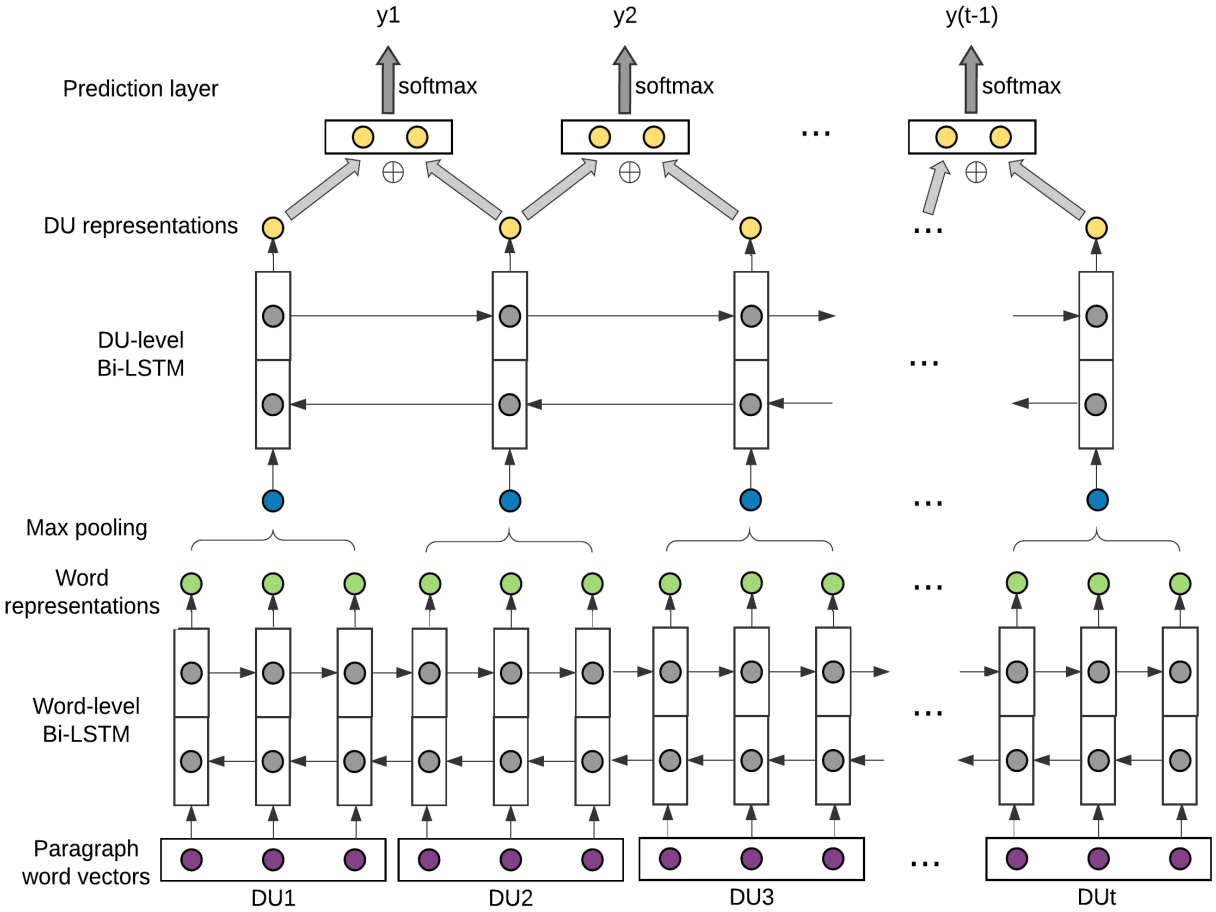


Figure 1: The Basic Model Architecture for Paragraph-level Discourse Relations Sequence Prediction.

$$MP_{DU}[j] = \max_{i=DU\_start}^{DU\_end} h_i[j] \quad (1)$$

$$where, 1 \leq j \leq hidden\_node\_size \quad (2)$$

Next, the DU-level Bi-LSTM will process the sequence of discourse unit embeddings in a paragraph and generate two hidden states $\overrightarrow{hDU_t}$ and $\overleftarrow{hDU_t}$ at each discourse unit position. We concatenate them to get the discourse unit representation $hDU_t = [\overrightarrow{hDU_t}, \overleftarrow{hDU_t}]$.

The Softmax Prediction Layer: Finally, we concatenate two adjacent discourse unit representations hDU_{t-1} and hDU_t and predict the discourse relation between them using a softmax function:

$$y_{t-1} = softmax(W_y * [hDU_{t-1}, hDU_t] + b_y) \quad (3)$$

3.2 Untie Parameters in the Softmax Prediction Layer (Implicit vs. Explicit)

Previous work (Pitler and Nenkova, 2009; Lin et al., 2014; Rutherford and Xue, 2016) has revealed that recognizing explicit vs. implicit discourse relations requires different strategies. Note that in the PDTB dataset, explicit discourse relations were distinguished from implicit ones, depending on whether a discourse connective exists between two discourse units. Therefore, explicit discourse relation detection can be simplified as a discourse connective phrase disambiguation problem (Pitler and Nenkova, 2009; Lin et al., 2014). On the contrary, predicting an implicit discourse relation should rely on understanding the overall

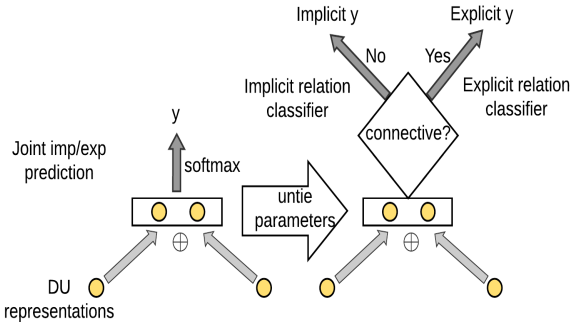


Figure 2: Untie Parameters in the Prediction Layer

contents of its two discourse units (Lin et al., 2014; Rutherford and Xue, 2016).

Considering the different natures of explicit vs. implicit discourse relation prediction, we decide to untie parameters at the final discourse relation prediction layer and train two softmax classifiers, as illustrated in Figure 2. The two classifiers have different sets of parameters, with one classifier for *only* implicit discourse relations and the other for *only* explicit discourse relations.

$$y_{t-1} = \begin{cases} softmax(W_{exp}[hDU_{t-1}, hDU_t] + b_{exp}), & exp \\ softmax(W_{imp}[hDU_{t-1}, hDU_t] + b_{imp}), & imp \end{cases} \quad (4)$$

The loss function used for the neural network training considers loss induced by both implicit relation prediction and explicit relation prediction:

$$Loss = Loss_{imp} + \alpha * Loss_{exp} \quad (5)$$

The α, in the full system, is set to be 1, which means that minimizing the loss in predicting either type of discourse relations is equally important. In the evaluation, we will also evaluate a system variant, where we will set $\alpha = 0$, which means that the neural network will not attempt to predict explicit discourse relations and implicit discourse relation prediction will not be influenced by predicting neighboring explicit discourse relations.

3.3 Fine-tune Discourse Relation Predictions Using a CRF Layer

Data analysis and many linguistic studies (Pitler et al., 2008; Asr and Demberg, 2012; Lascarides and Asher, 1993; Hobbs, 1985) have repeatedly shown that discourse relations feature continuity and patterns (e.g., a temporal relation is likely to be followed by another temporal relation). Especially, Pitler et al. (2008) firstly reported that patterns exist between implicit discourse relations and their neighboring explicit discourse relations.

Motivated by these observations, we aim to improve implicit discourse relation detection by making use of easily identifiable explicit discourse relations and taking into account global patterns of discourse relation distributions. Specifically, we add an extra CRF layer at the top of the softmax prediction layer (shown in figure 3) to fine-tune predicted discourse relations by considering their inter-dependencies.

The Conditional Random Fields (Lafferty et al., 2001) (CRF) layer updates a state transition matrix, which can effectively adjust the current label depending on proceeding and following labels. Both training and decoding of the CRF layer can be solved efficiently by using the Viterbi algorithm. With the CRF layer, the model jointly assigns a sequence of discourse relations between each two adjacent discourse units in a paragraph, including both implicit and explicit relations, by considering relevant discourse unit representations as well as global discourse relation patterns.

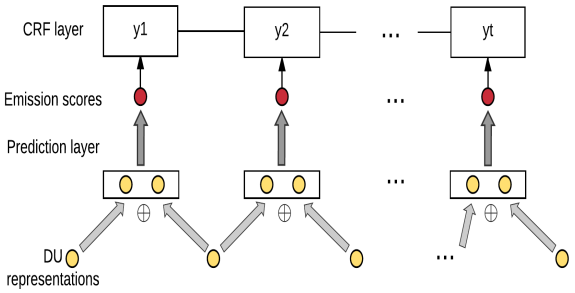


Figure 3: Fine-tune Discourse Relations with a CRF layer.

4 Evaluation

4.1 Dataset and Preprocessing

The Penn Discourse Treebank (PDTB): We experimented with PDTB v2.0 (Prasad et al., 2008b) which is the largest annotated corpus containing 36k discourse relations in 2,159 Wall Street Journal (WSJ) articles. In this work, we focus on the top-level[4] discourse relation senses which are consist of four major semantic classes: Comparison (Comp), Contingency (Cont), Expansion (Exp) and Temporal (Temp). We followed the same PDTB section partition (Rutherford and Xue, 2015) as previous work and used sections 2-20 as training set, sections 21-22 as test set, and sections 0-1 as development set. Table 1 presents the data distributions we collected from PDTB.

Preprocessing: The PDTB dataset documents its annotations as a list of discourse relations, with each relation associated with its two discourse units. To recover the paragraph context for a discourse relation, we match contents of its two annotated discourse units with all paragraphs in corresponding raw WSJ article. When all the matching was completed, each paragraph was split into a sequence of discourse units, with one discourse relation (implicit or explicit) between each two ad-

[4]In PDTB, the sense label of discourse relation was annotated hierarchically with three levels.

Type	Class	Train	Dev	Test	Total
Implicit	Comp	1942	197	152	2291
	Cont	3339	292	279	3910
	Exp	7003	671	574	8248
	Temp	760	64	85	909
Explicit	Comp	4184	422	364	4970
	Cont	2837	286	213	3336
	Exp	4612	481	424	5517
	Temp	2742	254	297	3293

Table 1: Distributions of Four Top-level Discourse Relations in PDTB.

# of DUs	2	3	4	5	>5
ratio	44%	25%	15%	7.3%	8.7%

Table 2: Distributions of Paragraphs.

jacent discourse units[5]. Following this method, we obtained 14,309 paragraphs in total, each contains 3.2 discourse units on average. Table 2 shows the distribution of paragraphs based on the number of discourse units in a paragraph.

4.2 Parameter Settings and Model Training

We tuned the parameters based on the best performance on the development set. We fixed the weights of word embeddings during training. All the LSTMs in our neural network use the hidden state size of 300. To avoid overfitting, we applied dropout (Hinton et al., 2012) with dropout ratio of 0.5 to both input and output of LSTM layers. To prevent the exploding gradient problem in training LSTMs, we adopt gradient clipping with gradient L2-norm threshold of 5.0. These parameters remain the same for all our proposed models as well as our own baseline models.

We chose the standard cross-entropy loss function for training our neural network model and adopted Adam (Kingma and Ba, 2014) optimizer with the initial learning rate of 5e-4 and a mini-batch size of 128[6]. If one instance is annotated with two labels (4% of all instances), we use both of them in loss calculation and regard the prediction as correct if model predicts one of the annotated labels. All the proposed models were implemented with Pytorch[7] and converged to the best performance within 20-40 epochs.

To alleviate the influence of randomness in neural network model training and obtain stable experimental results, we ran each of the proposed models and our own baseline models ten times and report the average performance of each model instead of the best performance as reported in many previous works.

4.3 Baseline Models and Systems

We compare the performance of our neural network model with several recent discourse relation recognition systems that only consider two relevant discourse units.

- (Rutherford and Xue, 2015): improves implicit discourse relation prediction by creating more training instances from the Gigaword corpus utilizing explicitly mentioned discourse connective phrases.
- (Chen et al., 2016): a gated relevance network (GRN) model with tensors to capture semantic interactions between words from two discourse units.
- (Liu et al., 2016): a convolutional neural network model that leverages relations between different styles of discourse relations annotations (PDTB and RST (Carlson et al., 2003)) in a multi-task joint learning framework.
- (Liu and Li, 2016): a multi-level attention-over-attention model to dynamically exploit features from two discourse units for recognizing an implicit discourse relation.
- (Qin et al., 2017): a novel pipelined adversarial framework to enable an adaptive imitation competition between the implicit network and a rival feature discriminator with access to connectives.
- (Lei et al., 2017): a Simple Word Interaction Model (SWIM) with tensors that captures both linear and quadratic relations between words from two discourse units.
- (Lan et al., 2017): an attention-based LSTM neural network that leverages explicit discourse relations in PDTB and unannotated external data in a multi-task joint learning framework.

[5] In several hundred discourse relations, one discourse unit is complex and can be further separated into two elementary discourse units, which can be illustrated as [DU1-DU2]-DU3. We simplify such cases to be a relation between DU2 and DU3.

[6] Counted as the number of discourse relations rather than paragraph instances.

[7] http://pytorch.org/

	Implicit						Explicit	
Model	Macro	Acc	Comp	Cont	Exp	Temp	Macro	Acc
(Rutherford and Xue, 2015)	40.50	57.10	-	-	-	-	-	-
(Liu et al., 2016)	44.98	57.27	-	-	-	-	-	-
(Liu and Li, 2016)	46.29	57.57	-	-	-	-	-	-
(Lei et al., 2017)	46.46	-	-	-	-	-	-	-
(Lan et al., 2017)	47.80	57.39	-	-	-	-	-	-
DU-pair level Discourse Relation Recognition (Our Own Baselines)								
Bi-LSTM	40.01	53.50	30.52	42.06	65.52	21.96	-	-
+ tensors	45.36	57.18	36.88	44.85	68.70	30.74	-	-
Paragraph level Discourse Relation Recognition								
Basic System Variant ($\alpha = 0$)	47.56	56.88	37.12	46.47	67.72	38.92	-	-
Basic System ($\alpha = 1$)	48.10	57.52	37.33	47.89	68.39	38.80	91.93	92.89
+ Untie Parameters	48.69	**58.20**	37.68	49.19	**68.86**	39.04	**93.70**	**94.46**
+ the CRF Layer	**48.82**	57.44	**37.72**	**49.39**	67.45	**40.70**	93.21	93.98

Table 3: Multi-class Classification Results on PDTB. We report accuracy (Acc) and macro-average F1-scores for both explicit and implicit discourse relation predictions. We also report class-wise F1 scores.

4.4 Evaluation Settings

On the PDTB corpus, both binary classification and multi-way classification settings are commonly used to evaluate the implicit discourse relation recognition performance. We noticed that all the recent works report class-wise implicit relation prediction performance in the binary classification setting, while none of them report detailed performance in the multi-way classification setting. In the binary classification setting, separate "one-versus-all" binary classifiers were trained, and each classifier is to identify one class of discourse relations. Although separate classifiers are generally more flexible in combating with imbalanced distributions of discourse relation classes and obtain higher class-wise prediction performance, one pair of discourse units may be tagged with all four discourse relations without proper conflict resolution. Therefore, the multi-way classification setting is more appropriate and natural in evaluating a practical end-to-end discourse parser, and we mainly evaluate our proposed models using the four-way multi-class classification setting.

Since none of the recent previous work reported class-wise implicit relation classification performance in the multi-way classification setting, for better comparisons, we re-implemented the neural tensor network architecture (so-called SWIM in (Lei et al., 2017)) which is essentially a Bi-LSTM model with tensors and report its detailed evaluation result in the multi-way classification setting. As another baseline, we report the performance of a Bi-LSTM model without tensors as well. Both baseline models take two relevant discourse units as the only input.

For additional comparisons, We also report the performance of our proposed models in the binary classification setting.

4.5 Experimental Results

Multi-way Classification: The first section of table 3 shows macro average F1-scores and accuracies of previous works. The second section of table 3 shows the multi-class classification results of our implemented baseline systems. Consistent with results of previous works, neural tensors, when applied to Bi-LSTMs, improved implicit discourse relation prediction performance. However, the performance on the three small classes (Comp, Cont and Temp) remains low.

The third section of table 3 shows the multi-class classification results of our proposed paragraph-level neural network models that capture inter-dependencies among discourse units. The first row shows the performance of a variant of our basic model, where we only identify implicit relations and ignore identifying explicit relations by setting the α in equation (5) to be 0. Compared with the baseline Bi-LSTM model, the only difference is that this model considers paragraph-wide contexts and model inter-dependencies among discourse units when building representation for individual DU. We can see that this model has greatly improved implicit relation classification perfor-

Model	Comp	Cont	Exp	Temp
(Chen et al., 2016)	40.17	54.76	-	31.32
(Liu et al., 2016)	37.91	55.88	69.97	37.17
(Liu and Li, 2016)	36.70	54.48	70.43	38.84
(Qin et al., 2017)	40.87	54.56	72.38	36.20
(Lei et al., 2017)	40.47	55.36	69.50	35.34
(Lan et al., 2017)	40.73	**58.96**	**72.47**	38.50
Paragraph level Discourse Relation Recognition				
Basic System ($\alpha = 1$)	42.68	55.17	68.94	41.03
+ Untie Parameters	**46.79**	57.09	70.41	**45.61**

Table 4: Binary Classification Results on PDTB. We report F1-scores for implicit discourse relations.

	Implicit		Explicit	
Model	Macro	Acc	Macro	Acc
Basic System ($\alpha = 1$)	49.92	59.08	93.05	93.83
+ Untie Parameters	50.47	**59.85**	93.95	94.74
+ the CRF Layer	**51.84**	59.75	**94.17**	**94.82**

Table 5: Multi-class Classification Results of Ensemble Models on PDTB.

mance across all the four relations and improved the macro-average F1-score by over 7 percents. In addition, compared with the baseline Bi-LSTM model with tensor, this model improved implicit relation classification performance across the three small classes, with clear performance gains of around 2 and 8 percents on contingency and temporal relations respectively, and overall improved the macro-average F1-score by 2.2 percents.

The second row shows the performance of our basic paragraph-level model which predicts both implicit and explicit discourse relations in a paragraph. Compared to the variant system (the first row), the basic model further improved the classification performance on the first three implicit relations. Especially on the contingency relation, the classification performance was improved by another 1.42 percents. Moreover, the basic model yields good performance for recognizing explicit discourse relations as well, which is comparable with previous best result (92.05% macro F1-score and 93.09% accuracy as reported in (Pitler et al., 2008)).

After untying parameters in the softmax prediction layer, implicit discourse relation classification performance was improved across all four relations, meanwhile, the explicit discourse relation classification performance was also improved. The CRF layer further improved implicit discourse relation recognition performance on the three small classes. In summary, our full paragraph-level neural network model achieves the best macro-average F1-score of 48.82% in predicting implicit discourse relations, which outperforms previous neural tensor network models (e.g., (Lei et al., 2017)) by more than 2 percents and outperforms the best previous system (Lan et al., 2017) by 1 percent.

Binary Classification: From table 4, we can see that compared against the best previous systems, our paragraph-level model with untied parameters in the prediction layer achieves F1-score improvements of 6 points on Comparison and 7 points on Temporal, which demonstrates that paragraph-wide contexts are important in detecting minority discourse relations. Note that the CRF layer of the model is not suitable for binary classification.

4.6 Ensemble Model

As we explained in section 4.2, we ran our models for 10 times to obtain stable average performance. Then we also created ensemble models by applying majority voting to combine results of ten runs. From table 5, each ensemble model obtains performance improvements compared with single model. The full model achieves performance boosting of (51.84 - 48.82 = 3.02) and (94.17 - 93.21 = 0.96) in macro F1-scores for predicting implicit and explicit discourse relations respectively. Furthermore, the ensemble model achieves the best performance for predicting both implicit

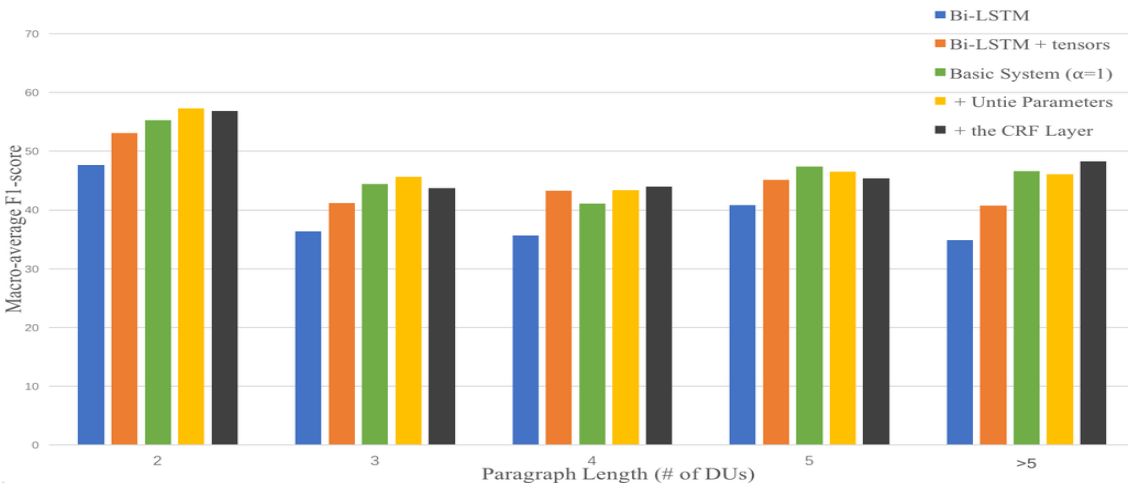


Figure 4: Impact of Paragraph Length. We plot the macro-average F1-score of implicit discourse relation classification on instances with different paragraph length.

and explicit discourse relations simultaneously.

4.7 Impact of Paragraph Length

To understand the influence of paragraph lengths to our paragraph-level models, we divide paragraphs in the PDTB test set into several subsets based on the number of DUs in a paragraph, and then evaluate our proposed models on each subset separately. From Figure 4, we can see that our paragraph-level models (the latter three) overall outperform DU-pair baselines across all the subsets. As expected, the paragraph-level models achieve clear performance gains on long paragraphs (with more than 5 DUs) by extensively modeling mutual influences of DUs in a paragraph. But somewhat surprisingly, the paragraph-level models achieve noticeable performance gains on short paragraphs (with 2 or 3 DUs) as well. We hypothesize that by learning more appropriate discourse-aware DU representations in long paragraphs, our paragraph-level models reduce bias of using DU representations in predicting discourse relations, which benefits discourse relation prediction in short paragraphs as well.

4.8 Example Analysis

For the example (1), the baseline neural tensor model predicted both implicit relations wrongly ("Implicit-Contingency" between DU2 and DU3; "Implicit-Expansion" between DU3 and DU4), while our paragraph-level model predicted all the four discourse relations correctly, which indicates that paragraph-wide contexts play a key role in implicit discourse relation prediction.

For another example:

(2): *[Marshall came clanking in like Marley's ghost dragging those chains of brigades and air wings and links with Arab despots.]*$_{DU1}$ ***(Implicit-Temporal)*** *[He wouldn't leave]*$_{DU2}$ *until* ***(Explicit-Temporal)*** *[Mr. Cheney promised to do whatever the Pentagon systems analysts told him.]*$_{DU3}$

Our basic paragraph-level model wrongly predicted the implicit discourse relation between DU1 and DU2 to be "Implicit-Comparison", without being able to effectively use the succeeding "Explicit-Temporal" relation. On the contrary, the full model corrected this mistake by modeling discourse relation patterns with the CRF layer.

5 Conclusion

We have presented a paragraph-level neural network model that takes a sequence of discourse units as input, models inter-dependencies between discourse units as well as discourse relation continuity and patterns, and predicts a sequence of discourse relations in a paragraph. By building wider-context informed discourse unit representations and capturing the overall discourse structure, the paragraph-level neural network model outperforms the best previous models for implicit discourse relation recognition on the PDTB dataset.

Acknowledgments

We acknowledge the support of NVIDIA Corporation for their donation of one GeForce GTX TITAN X GPU used for this research.

References

Fatemeh Torabi Asr and Vera Demberg. 2012. Implicitness of discourse relations. In *Coling*. pages 2669–2684.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2003. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Current and new directions in discourse and dialogue*, Springer, pages 85–112.

Jifan Chen, Qi Zhang, Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Implicit discourse relation detection via a deep architecture with gated relevance network. In *ACL 2016*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. pages 681–691. http://aclanthology.info/papers/D17-1071/d17-1071.

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

Jerry R Hobbs. 1985. *On the coherence and structure of discourse*. CSLI.

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

Yangfeng Ji and Jacob Eisenstein. 2015. One vector is not enough: Entity-augmented distributed semantics for discourse relations. *TACL* 3:329–344. https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/536.

Yangfeng Ji, Gholamreza Haffari, and Jacob Eisenstein. 2016. A latent variable recurrent neural network for discourse relation language models. In *Proceedings of NAACL-HLT*. pages 332–342.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*. volume 951, pages 282–289.

Man Lan, Jianxiang Wang, Yuanbin Wu, Zheng-Yu Niu, and Haifeng Wang. 2017. Multi-task attention-based neural networks for implicit discourse relationship representation and identification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 1310–1319.

Alex Lascarides and Nicholas Asher. 1993. Temporal interpretation, discourse relations and commonsense entailment. *Linguistics and philosophy* 16(5):437–493.

Wenqiang Lei, Xuancong Wang, Meichun Liu, Ilija Ilievski, Xiangnan He, and Min-Yen Kan. 2017. Swim: A simple word interaction model for implicit discourse relation recognition. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*. pages 4026–4032. https://doi.org/10.24963/ijcai.2017/562.

Qi Li, Tianshi Li, and Baobao Chang. 2016. Discourse parsing with attention-based hierarchical neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 362–371.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing implicit discourse relations in the penn discourse treebank. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP '09, pages 343–351. http://dl.acm.org/citation.cfm?id=1699510.1699555.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2014. A pdtb-styled end-to-end discourse parser. *Natural Language Engineering* 20(2):151–184.

Yang Liu and Mirella Lapata. 2017. Learning contextually informed representations for linear-time discourse parsing. In *EMNLP*.

Yang Liu and Sujian Li. 2016. Recognizing implicit discourse relations via repeated reading: Neural networks with multi-level attention. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. pages 1224–1233. http://aclweb.org/anthology/D/D16/D16-1130.pdf.

Yang Liu, Sujian Li, Xiaodong Zhang, and Zhifang Sui. 2016. Implicit discourse relation classification via multi-task neural networks. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*. pages 2750–2756. http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/11831.

Ramesh Nallapati, Bowen Zhou, Cicero Nogueira dos santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *CoNLL 2016* page 280.

Emily Pitler, Annie Louis, and Ani Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '09, pages 683–691. http://dl.acm.org/citation.cfm?id=1690219.1690241.

Emily Pitler and Ani Nenkova. 2009. Using syntax to disambiguate explicit discourse connectives in text. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*. Association for Computational Linguistics, pages 13–16.

Emily Pitler, Mridhula Raghupathy, Hena Mehta, Ani Nenkova, Alan Lee, and Aravind Joshi. 2008. Easily identifiable discourse relations. In *In Proceedings of the 22nd International Conference on Computational Linguistics (COLING 2008) Short Papers*.

R. Prasad, N. Dinesh, Lee A., E. Miltsakaki, L. Robaldo, Joshi A., and B. Webber. 2008a. The Penn Discourse Treebank 2.0. In *lrec2008*.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008b. The penn discourse treebank 2.0. In *In Proceedings of LREC*.

Lianhui Qin, Zhisong Zhang, and Hai Zhao. 2016. A stacking gated neural architecture for implicit discourse relation classification. In *EMNLP*. pages 2263–2270.

Lianhui Qin, Zhisong Zhang, Hai Zhao, Zhiting Hu, and Eric P. Xing. 2017. Adversarial connective-exploiting networks for implicit discourse relation classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. pages 1006–1017. https://doi.org/10.18653/v1/P17-1093.

Attapol Rutherford and Nianwen Xue. 2015. Improving the inference of implicit discourse relations via classifying explicit discourse connectives. In *HLT-NAACL*. pages 799–808.

Attapol T Rutherford, Vera Demberg, and Nianwen Xue. 2016. Neural network models for implicit discourse relation classification in english and chinese without surface features. *arXiv preprint arXiv:1606.01990* .

Attapol T Rutherford and Nianwen Xue. 2016. Robust non-explicit neural discourse parser in english and chinese. *ACL 2016* page 55.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45(11):2673–2681.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*. pages 3776–3784.

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

Nianwen Xue, Hwee Tou Ng, Sameer Pradhan, Rashmi PrasadO Christopher Bryant, and Attapol T Rutherford. 2015. The conll-2015 shared task on shallow discourse parsing. *CoNLL 2015* page 1.

Biao Zhang, Jinsong Su, Deyi Xiong, Yaojie Lu, Hong Duan, and Junfeng Yao. 2015. Shallow convolutional neural network for implicit discourse relation recognition. In *EMNLP*. The Association for Computational Linguistics, pages 2230–2235.

Ruqing Zhang, Jiafeng Guo, Yanyan Lan, Jun Xu, and Xueqi Cheng. 2017a. Spherical paragraph model. *arXiv preprint arXiv:1707.05635* .

Yizhe Zhang, Dinghan Shen, Guoyin Wang, Zhe Gan, Ricardo Henao, and Lawrence Carin. 2017b. Deconvolutional paragraph representation learning. In *Advances in Neural Information Processing Systems*. pages 4170–4180.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. volume 2, pages 207–212.

A Deep Ensemble Model with Slot Alignment for Sequence-to-Sequence Natural Language Generation

Juraj Juraska, Panagiotis Karagiannis, Kevin K. Bowden and **Marilyn A. Walker**
Natural Language and Dialogue Systems Lab
University of California, Santa Cruz
{jjuraska,pkaragia,kkbowden,mawalker}@ucsc.edu

Abstract

Natural language generation lies at the core of generative dialogue systems and conversational agents. We describe an ensemble neural language generator, and present several novel methods for data representation and augmentation that yield improved results in our model. We test the model on three datasets in the restaurant, TV and laptop domains, and report both objective and subjective evaluations of our best model. Using a range of automatic metrics, as well as human evaluators, we show that our approach achieves better results than state-of-the-art models on the same datasets.

1 Introduction

There has recently been a substantial amount of research in natural language processing (NLP) in the context of personal assistants, such as Cortana or Alexa. The capabilities of these conversational agents are still fairly limited and lacking in various aspects, one of the most challenging of which is the ability to produce utterances with human-like coherence and naturalness for many different kinds of content. This is the responsibility of the natural language generation (NLG) component.

Our work focuses on language generators whose inputs are structured *meaning representations* (MRs). An MR describes a single dialogue act with a list of key concepts which need to be conveyed to the human user during the dialogue. Each piece of information is represented by a slot-value pair, where the *slot* identifies the type of information and the *value* is the corresponding content. *Dialogue act* (DA) types vary depending on the dialogue manager, ranging from simple ones, such as a *goodbye* DA with no slots at all, to complex ones, such as an *inform* DA containing multiple slots with various types of values (see example in Table 1).

MR	*inform* (name [**The Golden Curry**], food [**Japanese**], priceRange [**moderate**], familyFriendly [**yes**], near [**The Bakers**])
Utt.	Located near **The Bakers**, **kid-friendly** restaurant, **The Golden Curry**, offers **Japanese** cuisine with a **moderate** price range.

Table 1: An example of an MR and a corresponding reference utterance.

A natural language generator must produce a syntactically and semantically correct utterance from a given MR. The utterance should express all the information contained in the MR, in a natural and conversational way. In traditional language generator architectures, the assembling of an utterance from an MR is performed in two stages: *sentence planning*, which enforces semantic correctness and determines the structure of the utterance, and *surface realization*, which enforces syntactic correctness and produces the final utterance form.

Earlier work on statistical NLG approaches were typically hybrids of a handcrafted component and a statistical training method (Langkilde and Knight, 1998; Stent et al., 2004; Rieser and Lemon, 2010). The handcrafted aspects, however, lead to decreased portability and potentially limit the variability of the outputs. New corpus-based approaches emerged that used semantically aligned data to train language models that output utterances directly from their MRs (Mairesse et al., 2010; Mairesse and Young, 2014). The alignment provides valuable information during training, but the semantic annotation is costly.

The most recent methods do not require aligned data and use an *end-to-end* approach to training, performing sentence planning and surface realization simultaneously (Konstas and Lapata, 2013). The most successful systems trained on unaligned data use recurrent neural networks (RNNs) paired with an encoder-decoder system design (Mei et al.,

Proceedings of NAACL-HLT 2018, pages 152–162
New Orleans, Louisiana, June 1 - 6, 2018.

2016; Dušek and Jurčíček, 2016), but also other concepts, such as imitation learning (Lampouras and Vlachos, 2016). These NLG models, however, typically require greater amount of data for training due to the lack of semantic alignment, and they still have problems producing syntactically and semantically correct output, as well as being limited in naturalness (Nayak et al., 2017).

Here we present a neural ensemble natural language generator, which we train and test on three large unaligned datasets in the restaurant, television, and laptop domains. We explore novel ways to represent the MR inputs, including novel methods for delexicalizing slots and their values, automatic slot alignment, as well as the use of a semantic reranker. We use automatic evaluation metrics to show that these methods appreciably improve the performance of our model. On the largest of the datasets, the E2E dataset (Novikova et al., 2017b) with nearly 50K samples, we also demonstrate that our model significantly outperforms the baseline *E2E NLG Challenge*[1] system in human evaluation. Finally, after augmenting our model with stylistic data selection, subjective evaluations reveal that it can still produce overall better results despite a significantly reduced training set.

2 Related Work

NLG is closely related to machine translation and has similarly benefited from recent rapid development of deep learning methods. State-of-the-art NLG systems build thus on deep neural *sequence-to-sequence* models (Sutskever et al., 2014) with an *encoder-decoder* architecture (Cho et al., 2014) equipped with an *attention* mechanism (Bahdanau et al., 2015). They typically also rely on slot *delexicalization* (Mairesse et al., 2010; Henderson et al., 2014), which allows the model to better generalize to unseen inputs, as exemplified by TGen (Dušek and Jurčíček, 2016). However, Nayak et al. (2017) point out that there are frequent scenarios where delexicalization behaves inadequately (see Section 5.1 for more details), and Agarwal and Dymetman (2017) show that a *character-level* approach to NLG may avoid the need for delexicalization, at the potential cost of making more semantic omission errors.

The end-to-end approach to NLG typically requires a mechanism for aligning slots on the output utterances: this allows the model to generate utterances with fewer missing or redundant slots. Cuayáhuitl et al. (2014) perform automatic slot labeling using a Bayesian network trained on a labeled dataset, and show that a method using spectral clustering can be extended to unlabeled data with high accuracy. In one of the first successful neural approaches to language generation, Wen et al. (2015a) augment the generator's inputs with a control vector indicating which slots still need to be realized at each step. Wen et al. (2015b) take the idea further by embedding a new sigmoid gate into their LSTM cells, which directly conditions the generator on the DA. More recently, Dušek and Jurčíček (2016) supplement their encoder-decoder model with a trainable classifier which they use to rerank the beam search candidates based on missing and redundant slot mentions.

Our work builds upon the successful attentional encoder-decoder framework for sequence-to-sequence learning and expands it through ensembling. We explore the feasibility of a domain-independent slot aligner that could be applied to any dataset, regardless of its size, and beyond the reranking task. We also tackle some challenges caused by delexicalization in order to improve the quality of surface realizations, while retaining the ability of the neural model to generalize.

	E2E	TV	Laptop
|training set|	42061	4221	7944
|validation set|	4672	1407	2649
|test set|	630	1407	2649
total	47363	7035	13242
DA types	1	14	14
slot types	8	16	20

Table 2: Overview of the number of samples, as well as different DA and slot types, in each dataset .

3 Datasets

We evaluated the models on three datasets from different domains. The primary one is the recently released E2E restaurant dataset (Novikova et al., 2017b) with 48K samples. For benchmarking we use the TV dataset and the Laptop dataset (Wen et al., 2016) with 7K and 13K samples, respectively. Table 2 summarizes the proportions of the training, validation, and test sets for each dataset.

3.1 E2E Dataset

The E2E dataset is by far the largest one available for task-oriented language generation in the restaurant domain. The human references were

[1] http://www.macs.hw.ac.uk/InteractionLab/E2E/

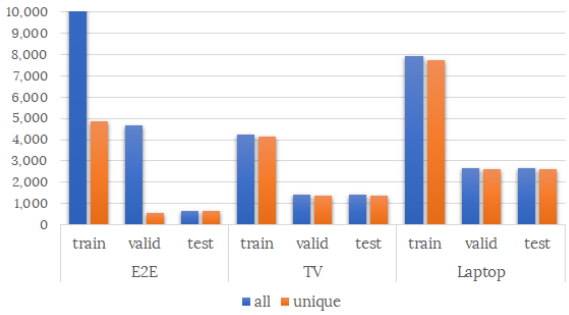


Figure 1: Proportion of unique MRs in the datasets. Note that the number of MRs in the E2E dataset was cut off at 10K for the sake of visibility of the small differences between other column pairs.

collected using pictures as the source of information, which was shown to inspire more informative and natural utterances (Novikova et al., 2016). With nearly 50K samples, it offers almost 10 times more data than the San Francisco restaurant dataset introduced in Wen et al. (2015b), which has frequently been used for benchmarks. The reference utterances in the E2E dataset exhibit superior lexical richness and syntactic variation, including more complex discourse phenomena. It aims to provide higher-quality training data for end-to-end NLG systems to learn to produce more naturally sounding utterances. The dataset was released as a part of the E2E NLG Challenge.

Although the E2E dataset contains a large number of samples, each MR is associated on average with 8.65 different reference utterances, effectively offering less than 5K unique MRs in the training set (Fig. 1). Explicitly providing the model with multiple ground truths, it offers multiple alternative utterance structures the model can learn to apply for the same type of MR. The delexicalization, as detailed later in Section 5.1, improves the ability of the model to share the concepts across different MRs.

The dataset contains only 8 different slot types, which are fairly equally distributed. The number of slots in each MR ranges between 3 and 8, but the majority of MRs consist of 5 or 6 slots. Even though most of the MRs contain many slots, the majority of the corresponding human utterances, however, consist of one or two sentences only (Table 3), suggesting a reasonably high level of sentence complexity in the references.

3.2 TV and Laptop Datasets

The reference utterances in the TV and the Laptop datasets were collected using Amazon Mechanical Turk (AMT), one utterance per MR. These two datasets are similar in structure, both using the same 14 DA types.[2] The Laptop dataset, however, is almost twice as large and contains 25% more slot types.

slots	3	4	5	6	7	8
sent.	1.09	1.23	1.41	1.65	1.84	1.92
prop.	5%	18%	32%	28%	14%	3%

Table 3: Average number of sentences in the reference utterance for a given number of slots in the corresponding MR, along with the proportion of MRs with specific slot counts.

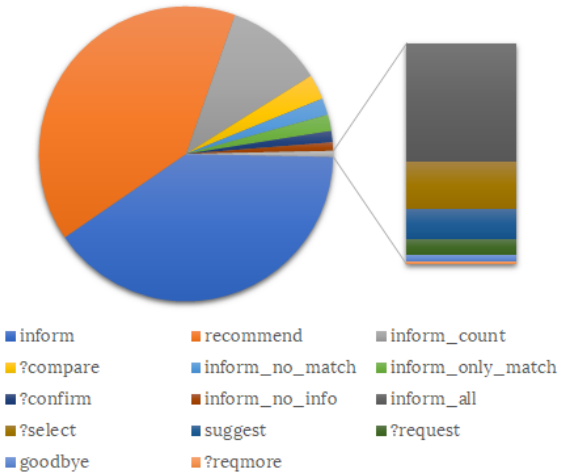


Figure 2: Proportion of DAs in the Laptop dataset.

Although both of these datasets contain more than a dozen different DA types, the vast majority (68% and 80% respectively) of the MRs describe a DA of either type `inform` or `recommend` (Fig. 2), which in most cases have very similarly structured realizations, comparable to those in the E2E dataset. DAs such as `suggest`, `?request`, or `goodbye` are represented by less than a dozen samples, but are significantly easier to learn to generate an utterance from because the corresponding MRs contain three slots at the most.

4 Ensemble Neural Language Generator

4.1 Encoder-Decoder with Attention

Our model uses the standard encoder-decoder architecture with attention, as defined in Bahdanau et al. (2015). Encoding the input into a sequence of context vectors instead of a single vector enables the decoder to learn what specific parts of the

[2] We noticed the MRs with the `?request` DA type in the TV dataset have no slots provided, as opposed to the Laptop dataset, so we imputed these in order to obtain valid MRs.

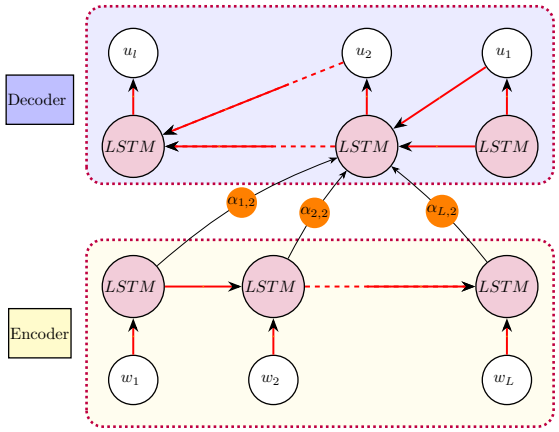


Figure 3: Standard architecture of a single-layer encoder-decoder LSTM model with attention. For each time step t in the output sequence, the attention scores $\alpha_{t,1}, \ldots, \alpha_{t,L}$ are calculated. This diagram shows the attention scores only for $t = 2$.

input sequence to pay attention to, given the output generated so far. In this attentional encoder-decoder architecture, the probability of the output at each time step t of the decoder depends on a distinct context vector q_t in the following way:

$$P(u_t|u_1, \ldots, u_{t-1}, \mathbf{w}) = g(u_{t-1}, s_t, q_t) \ ,$$

where in the place of function g we use the softmax function over the size of the vocabulary, and s_t is a hidden state of the decoder RNN at time step t, calculated as:

$$s_t = f(s_{t-1}, u_{t-1}, q_t) \ .$$

The context vector q_t is obtained as a weighted sum of all the hidden states $h_1, \ldots, h_L$ of the encoder:

$$q_t = \sum_{i=1}^{L} \alpha_{t,i} h_i \ ,$$

where $\alpha_{t,i}$ corresponds to the attention score the t-th word in the target sentence assigns to the i-th item in the input MR.

We compute the attention score $\alpha_{t,i}$ using a multi-layer perceptron (MLP) jointly trained with the entire system (Bahdanau et al., 2015). The encoder's and decoder's hidden states at time i and t, respectively, are concatenated and used as the input to the MLP, namely:

$$\alpha_{t,i} = softmax\left(\mathbf{w}^T tanh\left(W[h_i; s_t]\right)\right) \ ,$$

where W and $\mathbf{w}$ are the weight matrix and the vector of the first and the second layer of the MLP, respectively. The learned weights indicate the level of influence of the individual words in the input sequence on the prediction of the word at time step t of the decoder. The model thus learns a soft alignment between the source and the target sequence.

4.2 Ensembling

In order to enhance the quality of the predicted utterances, we create three neural models with different encoders. Two of the models use a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) encoder, whereas the third model has a CNN (LeCun et al., 1998) encoder. We train these models individually for a different number of epochs and then combine their predictions.

Initially, we attempted to combine the predictions of the models by averaging the log-probability at each time step and then selecting the word with the maximum log-probability. We noticed that the quality, as well as the BLEU score of our utterances, decreased significantly. We believe that this is due to the fact that different models learn different sentence structures and, hence, combining predictions at the probability level results in incoherent utterances.

Therefore, instead of combining the models at the log-probability level, we accumulate the top 10 predicted utterances from each model type using beam search and allow the reranker (see Section 4.4) to rank all candidate utterances taking the proportion of slots they successfully realized into consideration. Finally, our system predicts the utterance that received the highest score.

4.3 Slot Alignment

Our training data is inherently unaligned, meaning our model is not certain which sentence in a multi-sentence utterance contains a given slot, which limits the model's robustness. To accommodate this, we create a heuristic-based slot aligner which automatically preprocesses the data. Its primary goal is to align chunks of text from the reference utterances with an expected value from the MR. Applications of our slot aligner are described in subsequent sections and in Table 4.

In our task, we have a finite set of slot mentions which must be detected in the corresponding utterance. Moreover, from our training data we can see that most slots are realized by inserting a specific set of phrases into an utterance. Using this insight, we construct a gazetteer, which primarily searches for overlapping content between the MR and each

sentence in an utterance, by associating all possible slot realizations with their appropriate slot type. We additionally augment the gazetteer using a small set of handcrafted rules which capture cases not easily encapsulated by the above process, for example, associating the `priceRange` slot with a chunk of text using currency symbols or relevant lexemes, such as "cheap" or "high-end". While handcrafted, these rules are transferable across domains, as they target the slots, not the domains, and mostly serve to counteract the noise in the E2E dataset. Finally, we use WordNet (Fellbaum, 1998) to further augment the size of our gazetteer by accounting for synonyms and other semantic relationships, such as associating "pasta" with the `food[Italian]` slot.

4.4 Reranker

As discussed in Section 4.2, our model uses beam search to produce a pool of the most likely utterances for a given MR. While these results have a probability score provided by the model, we found that relying entirely on this score often results in the system picking a candidate which is objectively worse than a lower scoring utterance (i.e. one missing more slots and/or realizing slots incorrectly). We therefore augment that score by multiplying it by the following score which takes the slot alignment into consideration:

$$s_{\text{align}} = \frac{N}{(N_u + 1) \cdot (N_o + 1)},$$

where N is the number of all slots in the given MR, and N_u and N_o represent the number of unaligned slots (those not observed by our slot aligner) and over-generated slots (those which have been realized but were not present in the original MR), respectively.

5 Data Preprocessing

5.1 Delexicalization

We enhance the ability of our model to generalize the learned concepts to unseen MRs by delexicalizing the training data. Moreover, it reduces the amount of data required to train the model. We identify the categorical slots whose values always propagate verbatim to the utterance, and replace the corresponding values in the utterance with placeholder tokens. The placeholders are eventually replaced in the output utterance in post-processing by copying the values from the input MR. Examples of such slots would be `name` or `near` in the E2E dataset, and `screensize` or `processor` in the TV and the Laptop dataset.

Previous work identifies categorical slots as good delexicalization candidates that improve the performance of the model (Wen et al., 2015b; Nayak et al., 2017). However, we chose not to delexicalize those categorical slots whose values can be expressed in alternative ways, such as "less than $20" and "cheap", or "on the riverside" and "by the river". Excluding these from delexicalization may lead to an increased number of incorrect realizations, but it encourages diversity of the model's outputs by giving it a freedom to choose among alternative ways of expressing a slot-value in different contexts. This, however, assumes that the training set contains a sufficient number of samples displaying this type of alternation so that the model can learn that certain phrases are synonymous. With its multiple human references for each MR, the E2E dataset has this property.

As Nayak et al. (2017) point out, delexicalization affects the sentence planning and the lexical choice around the delexicalized slot value. For example, the realization of the slot `food[`*`Italian`*`]` in the phrase "serves *Italian* food" is valid, while the realization of `food[`*`fast food`*`]` in "serves *fast food* food" is clearly undesired. Similarly, a naive delexicalization can result in "a Italian restaurant", whereas the article should be "an". Another problem with articles is singular versus plural nouns in the slot value. For example, the slot `accessories` in the TV dataset, can take on values such as "remote control", as well as "3D glasses", where only the former requires an article before the value.

We tackle this issue by defining different placeholder tokens for values requiring different treatment in the realization. For instance, the value "Italian" of the `food` slot is replaced by `slot_vow_cuisine_food`, indicating that the value starts with a vowel and represents a cuisine, while "fast food" is replaced by `slot_con_food`, indicating that the value starts with a consonant and cannot be used as a term for cuisine. The model thus learns to generate "a" before `slot_con_food` and "an" before `slot_vow_cuisine_food` when appropriate, as well as to avoid generating the word "food" after `food`-slot placeholders that do not contain the word "cuisine". All these rules are general and

can automatically be applied across different slots and domains.

5.2 Data Expansion

Slot Permutation

In our initial experiments, we tried expanding the training set by permuting the slot ordering in the MRs as suggested in Nayak et al. (2017). From different slot orderings of every MR we sampled five random permutations (in addition to the original MR), and created new pseudo-samples with the same reference utterance. The training set thus increased six times in size.

Using such an augmented training set might add to the model's robustness, nevertheless it did not prove to be helpful with the E2E dataset. In this dataset, we observed the slot order to be fixed across all the MRs, both in the training and the test set. As a result, for the majority of the time, the model was training on MRs with slot orders it would never encounter in the test set, which ultimately led to a decreased performance in prediction on the test set.

Utterance/MR Splitting

Taking a more utterance-oriented approach, we augment the training set with single-sentence utterances paired with their corresponding MRs. These new pseudo-samples are generated by splitting the existing reference utterances into single sentences and using the slot aligner introduced in Section 4.3 to identify the slots that correspond to each sentence. The MRs of the new samples are created as the corresponding subsets of slots and, whenever the sentence contains the name (of the restaurant/TV/etc.) or a pronoun referring to it (such as "it" or "its"), the `name` slot is included too. Finally, a new `position` slot is appended to every new MR, indicating whether it represents the first sentence or a subsequent sentence in the original utterance. An example of this splitting technique can be seen in Table 4. The training set almost doubled in size through this process.

Since the slot aligner works heuristically, not all utterances are successfully aligned with the MR. The vast majority of such cases, however, is caused by reference utterances in the datasets having incorrect or entirely missing slot mentions. There is a noticeable proportion of those, so we leave them in the training set with the unaligned slots removed from the MR so as to avoid confusing the model when learning from such samples.

MR	name [The Waterman], food [English], priceRange [cheap], customer rating [average], area [city centre], familyFriendly [yes]
Utt.	There is a family-friendly, cheap restaurant in the city centre, called The Waterman. It serves English food and has an average rating by customers.
New MR #1	name [The Waterman], priceRange [cheap], area [city centre], familyFriendly [yes], *position [outer]*
New MR #2	name [The Waterman], food [English], customer rating [average], *position [inner]*

Table 4: An example of the utterance/MR splitting.

MR	name [Wildwood], eatType [coffee shop], food [English], priceRange [moderate], customer rating [1 out of 5], near [Ranch]
Simple utt.	Wildwood provides English food for a moderate price. It has a low customer rating and is located near Ranch. It is a coffee shop.
Elegant utt.	A low-rated English style coffee shop around Ranch, called Wildwood, has moderately priced food.

Table 5: Contrastive example of a simple and a more elegant reference utterance style for the same MR in the E2E dataset.

5.3 Sentence Planning via Data Selection

The quality of the training data inherently imposes an upper bound on the quality of the predictions of our model. Therefore, in order to bring our model to produce more sophisticated utterances, we experimented with filtering the training data to contain only the most natural sounding and structurally complex utterances for each MR. For instance, we prefer having an elegant, single-sentence utterance with an apposition as the reference for an MR, rather than an utterance composed of three simple sentences, two of which begin with "it" (see the examples in Table 5).

We assess the complexity and naturalness of each utterance by the use of discourse phenomena, such as contrastive cues, subordinate clauses, or aggregation. We identify these in the utterance's parse-tree produced by the Stanford CoreNLP toolkit (Manning et al., 2014) by defining a set of rules for extracting the discourse phenomena. Furthermore, we consider the number of sentences used to convey all the information in the corresponding MR, as longer sentences tend to exhibit more advanced discourse phenomena. Penalizing utterances for too many sentences contributes to reducing the proportion of generic reference utter-

ances, such as the "simple" example in the above table, in the filtered training set.

6 Evaluation

Researchers in NLG have generally used both automatic and human evaluation. Our results report the standard automatic evaluation metrics: BLEU (Papineni et al., 2002), NIST (Przybocki et al., 2009), METEOR (Lavie and Agarwal, 2007), and ROUGE-L (Lin, 2004). For the E2E dataset experiments, we additionally report the results of the human evaluation carried out on the CrowdFlower platform as a part of the E2E NLG Challenge.

6.1 Experimental Setup

We built our ensemble model using the seq2seq framework (Britz et al., 2017) for TensorFlow. Our individual LSTM models use a bidirectional LSTM encoder with 512 cells per layer, and the CNN models use a pooling encoder as in Gehring et al. (2017). The decoder in all models was a 4-layer RNN decoder with 512 LSTM cells per layer and with attention. The hyperparameters were determined empirically. After experimenting with different beam search parameters, we settled on the beam width of 10. Moreover, we employed the length normalization of the beams as defined in Wu et al. (2016), in order to encourage the decoder to favor longer sequences. The length penalty providing the best results on the E2E dataset was 0.6, whereas for the TV and Laptop datasets it was 0.9 and 1.0, respectively.

6.2 Experiments on the E2E Dataset

We start by evaluating our system on the E2E dataset. Since the reference utterances in the test set were kept secret for the E2E NLG Challenge, we carried out the metric evaluation using the validation set. This was necessary to narrow down the models that perform well compared to the baseline. The final model selection was done based on a human evaluation of the models' outputs on the test set.

6.2.1 Automatic Metric Evaluation

In the first experiment, we assess what effect the augmenting of the training set via utterance splitting has on the performance of different models. The results in Table 6 show that both the LSTM and the CNN models clearly benefit from additional pseudo-samples in the training set. This can likely be attributed to the model having access to more granular information about which parts of the utterance correspond to which slots in the MR. This may assist the model in sentence planning and building a stronger association between parts of the utterance and certain slots, such as that "it" is a substitute for the name.

		BLEU	NIST	METEOR	ROUGE
LSTM	$\bar{\mathbf{s}}$	0.6664	8.0150	0.4420	0.7062
	$\mathbf{s}$	0.6930‡	8.4198	0.4379	0.7099
CNN	$\bar{\mathbf{s}}$	0.6599	7.8520	0.4333	0.7018
	$\mathbf{s}$	0.6760†	8.0440	0.4448	0.7055

Table 6: Automatic metric scores of different models tested on the E2E dataset, both unmodified ($\bar{\mathbf{s}}$) and augmented ($\mathbf{s}$) through the utterance splitting. The symbols † and ‡ indicate statistically significant improvement over the $\bar{\mathbf{s}}$ counterpart with $p < 0.05$ and $p < 0.01$, respectively, based on the paired t-test.

Testing our ensembling approach reveals that reranking predictions pooled from different models produces an ensemble model that is overall more robust than the individual submodels. The submodels fail to perform well in all four metrics at once, whereas the ensembling creates a new model that is more consistent across the different metric types (Table 7).[3] While the ensemble model decreases the proportion of incorrectly realized slots compared to its individual submodels on the validation set, on the test set it only outperforms two of the submodels in this aspect (Table 8). Analyzing the outputs, we also observed that the CNN model surpassed the two LSTM models in the ability to realize the "fast food" and "pub" values reliably, both of which were hardly present in the validation set but very frequent in the test set. On the official E2E test set, our ensemble model performs comparably to the baseline model, TGen (Dušek and Jurčíček, 2016), in terms of automatic metrics (Table 9).

6.2.2 Human Evaluation

It is known that automatic metrics function only as a general and vague indication of the quality of an utterance in a dialogue (Liu et al., 2016; Novikova et al., 2017a). Systems which score similarly according to these metrics could produce utterances that are significantly different because automatic

[3] The scores here correspond to the model submitted to the E2E NLG Challenge. Subsequently, we found better performing models according to some metrics: see Table 6.

	BLEU	NIST	METEOR	ROUGE
LSTM1	0.6661	8.1626	0.4644	0.7018
LSTM2	0.6493	7.9996	0.4649	0.6995
CNN	0.6636	7.9617	0.4700	0.7107
Ensem.	0.6576	8.0761	0.4675	0.7029

Table 7: Automatic metric scores of three different models and their ensemble, tested on the *validation set* of the E2E dataset. LSTM2 differs from LSTM1 in that it was trained longer.

	Validation set	Test set
LSTM1	0.116%	0.988%
LSTM2	0.145%	1.241%
CNN	0.232%	0.253%
Ensem.	0.087%	0.965%

Table 8: Error rate of the ensemble model compared to its individual submodels.

metrics fail to capture many of the characteristics of natural sounding utterances. Therefore, to better assess the structural complexity of the predictions of our model, we present the results of a human evaluation of the models' outputs in terms of both naturalness and quality, carried out by the E2E NLG Challenge organizers.

Quality examines the grammatical correctness and adequacy of an utterance given an MR, whereas *naturalness* assesses whether a predicted utterance could have been produced by a native speaker, irrespective of the MR. To obtain these scores, crowd workers ranked the outputs of 5 randomly selected systems from worst to best. The final scores were produced using the TrueSkill algorithm (Sakaguchi et al., 2014) through pairwise comparisons of the human evaluation scores among the 20 competing systems.

Our system, trained on the E2E dataset without stylistic selection (Section 5.3), achieved the highest quality score in the E2E NLG Challenge, and was ranked second in naturalness.[4] The system's performance in quality (the primary metric) was significantly better than the competition according to the TrueSkill evaluation, which used bootstrap resampling with a p-level of $p \leq 0.05$. Comparing these results with the scores achieved by the baseline model in quality and naturalness (5th and 6th place, respectively) reinforces our belief that models that perform similarly on the automatic metrics (Table 9) can exhibit vast differences in the structural complexity of their generated utterances.

[4]The system that surpassed ours in naturalness was ranked the last according to the quality metric.

	BLEU	NIST	METEOR	ROUGE
TGen	0.6593	8.6094	*0.4483*	*0.6850*
Ensem.	*0.6619*	*8.6130*	0.4454	0.6772

Table 9: Automatic metric scores of our ensemble model compared against TGen (the baseline model), tested on the *test set* of the E2E dataset.

Ex. #1	The Cricketers is a cheap Chinese restaurant near All Bar One in the riverside area, **but it has an average customer rating and is not family friendly**.
Ex. #2	**If you are looking for** a coffee shop near The Rice Boat, **try** Giraffe.

Table 10: Examples of generated utterances that contain more advanced discourse phenomena.

6.2.3 Experiments with Data Selection

After filtering the E2E training set as described in Section 5.3, the new training set consisted of approximately 20K pairs of MRs and utterances. Interestingly, despite this drastic reduction in training samples, the model was able to learn more complex utterances that contained the natural variations of the human language. The generated utterances exhibited discourse phenomena such as contrastive cues (see Example #1 in Table 10), as well as a more conversational style (Example #2). Nevertheless, the model also failed to realize slots more frequently.

In order to observe the effect of stylistic data selection, we conducted a human evaluation where we assessed the utterances based on *error rate* and *naturalness*. The error rate is calculated as the percentage of slots the model failed to realize divided by the total number of slots present among all samples. The annotators ranked samples of utterance triples – corresponding to three different ensemble models – by naturalness from 1 to 3 (3 being the most natural, with possible ties). The *conservative* model combines three submodels all trained on the full training set, the *progressive* one combines submodels solely trained on the filtered dataset, and finally, the *hybrid* is an ensemble of three models only one of which is trained on the full training set, so as to serve as a fallback.

The impact of the reduction of the number of

Ensemble model	Error rate	Naturalness
Conservative	*0.40%*	2.196
Progressive	1.60%	2.118
Hybrid	*0.40%*	*2.435*

Table 11: Average error rate and naturalness metrics obtained from six annotators for different ensemble models.

training samples becomes evident by looking at the score of the progressive model (Table 11), where this model trained solely on the reduced dataset had the highest error rate. We observe, however, that a hybrid ensemble model manages to perform the best in terms of the error rate, as well as the naturalness.

These results suggest that filtering the dataset through careful data selection can help to achieve better and more natural sounding utterances. It significantly improves the model's ability to produce more elegant utterances beyond the "[name] is... It is/has..." format, which is only too common in neural language generators in this domain.

6.3 Experiments on TV and Laptop Datasets

In order to provide a better frame of reference for the performance of our proposed model, we utilize the RNNLG benchmark toolkit[5] to evaluate our system on two additional, widely used datasets in NLG, and compare our results with those of a state-of-the-art model, SCLSTM (Wen et al., 2015b). As Table 12 shows, our ensemble model performs competitively with the baseline on the TV dataset, and it outperforms it on the Laptop dataset by a wide margin. We believe the higher error rate of our model can be explained by the significantly less aggressive slot delexicalization than the one used in SCLSTM. That, however, gives our model a greater lexical freedom and, with it, the ability to produce more natural utterances.

The model trained on the Laptop dataset is also a prime example of how an ensemble model is capable of extracting the best learned concepts from each individual submodel. By combining their knowledge and compensating thus for each other's weaknesses, the ensemble model can achieve a lower error rate, as well as a better overall quality, than any of the submodels individually.

[5]https://github.com/shawnwun/RNNLG

	TV		Laptop	
	BLEU	**ERR**	**BLEU**	**ERR**
SCLSTM	0.5265	2.31%	0.5116	0.79%
LSTM	0.5012	3.86%	0.5083	4.43%
CNN	0.5287	1.87%	0.5231	2.25%
Ensem.	0.5226	1.67%	0.5238	1.55%

Table 12: Automatic metric scores of our ensemble model evaluated on the test sets of the TV and Laptop datasets, and compared against SCLSTM. The ERR column indicates the slot error rate, as computed by the RNNLG toolkit (for our models calculated in post-processing).

7 Conclusion and Future Work

In this paper we presented our ensemble attentional encoder-decoder model for generating natural utterances from MRs. Moreover, we presented novel methods of representing the MRs to improve performance. Our results indicate that the proposed utterance splitting applied to the training set greatly improves the neural model's accuracy and ability to generalize. The ensembling method paired with the reranking based on slot alignment also contributed to the increase in quality of the generated utterances, while minimizing the number of slots that are not realized during the generation. This also enables the use of a less aggressive delexicalization, which in turn stimulates diversity in the produced utterances.

We showed that automatic slot alignment can be utilized for expanding the training data, as well as for utterance reranking. Our alignment currently relies in part on empirically observed heuristics, and a more robust aligner would allow for more flexible expansion into new domains. Since the stylistic data selection noticeably improved the diversity of our system's outputs, we believe this is a method with future potential, which we intend to further explore. Finally, it is clear that current automatic evaluation metrics in NLG are only sufficient for providing a vague idea as to the system's performance; we postulate that leveraging the reference data to train a classifier will result in a more conclusive automatic evaluation metric.

Acknowledgements

This research was partially supported by NSF Robust Intelligence #IIS-1302668-002.

References

Shubham Agarwal and Marc Dymetman. 2017. A surprisingly effective out-of-the-box char2char model on the e2e nlg challenge dataset. In *SIGDIAL Conference*.

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

Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc V. Le. 2017. Massive exploration of neural machine translation architectures. *CoRR* abs/1703.03906.

Kyunghyun Cho, Bart van Merriënboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*.

Heriberto Cuayáhuitl, Nina Dethlefs, Helen Hastie, and Xingkun Liu. 2014. Training a statistical surface realiser from automatic slot labelling. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*. IEEE, pages 112–117.

Ondřej Dušek and Filip Jurčíček. 2016. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings .

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. 2017. Convolutional sequence to sequence learning. In *ICML*.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014. Robust dialog state tracking using delexicalised recurrent neural networks and unsupervised adaptation. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*. IEEE, pages 360–365.

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

Ioannis Konstas and Mirella Lapata. 2013. A global model for concept-to-text generation. *J. Artif. Intell. Res.(JAIR)* 48:305–346.

Gerasimos Lampouras and Andreas Vlachos. 2016. Imitation learning for language generation from unaligned data. In *COLING*.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *COLING-ACL*.

Alon Lavie and Abhaya Agarwal. 2007. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*. Association for Computational Linguistics, pages 228–231.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out* .

Chia-Wei Liu, Ryan Joseph Lowe, Iulian Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *EMNLP*.

François Mairesse, Milica Gašić, Filip Jurčíček, Simon Keizer, Blaise Thomson, Kai Yu, and Steve Young. 2010. Phrase-based statistical language generation using graphical models and active learning. In *ACL*.

François Mairesse and Steve Young. 2014. Stochastic language generation in dialogue using factored language models. *Computational Linguistics* 40:763–799.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL*.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. *NAACL* .

Neha Nayak, Dilek Hakkani-Tur, Marilyn Walker, and Larry Heck. 2017. To plan or not to plan? discourse planning in slot-value informed sequence to sequence models for language generation. In *INTERSPEECH*.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017a. Why we need new evaluation metrics for nlg. In *EMNLP*.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017b. The E2E NLG shared task .

Jekaterina Novikova, Oliver Lemon, and Verena Rieser. 2016. Crowd-sourcing nlg data: Pictures elicit better data. *INLG* .

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Mark Przybocki, Kay Peterson, Sébastien Bronsart, and Gregory Sanders. 2009. The nist 2008 metrics for machine translation challenge – overview, methodology, metrics, and results. *Machine Translation* 23(2-3):71–103.

Verena Rieser and Oliver Lemon. 2010. Natural language generation as planning under uncertainty for spoken dialogue systems. In *Empirical methods in natural language generation*, Springer, pages 105–120.

Keisuke Sakaguchi, Matt Post, and Benjamin Van Durme. 2014. Efficient elicitation of annotations for human evaluation of machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, pages 1–11.

Amanda Stent, Rashmi Prasad, and Marilyn Walker. 2004. Trainable sentence planning for complex information presentation in spoken dialog systems. In *Proceedings of the 42nd annual meeting on association for computational linguistics*. Association for Computational Linguistics, page 79.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Tsung-Hsien Wen, Milica Gašić, Dongho Kim, Nikola Mrkšić, Pei hao Su, David Vandyke, and Steve Young. 2015a. Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. In *SIGDIAL Conference*.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, and Steve Young. 2016. Multi-domain neural network language generation for spoken dialogue systems. In *NAACL*.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015b. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *EMNLP*.

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

A Melody-conditioned Lyrics Language Model

Kento Watanabe[1], **Yuichiroh Matsubayashi**[1],
Satoru Fukayama[2], **Masataka Goto**[2], **Kentaro Inui**[1,3], **Tomoyasu Nakano**[2]
[1]Graduate School of Information Sciences, Tohoku University,
[2]National Institute of Advanced Industrial Science and Technology (AIST),
[3]RIKEN Center for Advanced Intelligence Project
{kento.w, y-matsu, inui}@ecei.tohoku.ac.jp,
{s.fukayama, m.goto, t.nakano}@aist.go.jp

Abstract

This paper presents a novel, data-driven language model that produces entire lyrics for a given input melody. Previously proposed models for lyrics generation suffer from the inability of capturing the relationship between lyrics and melody partly due to the unavailability of lyrics-melody aligned data. In this study, we first propose a new practical method for creating a large collection of lyrics-melody aligned data and then create a collection of 1,000 lyrics-melody pairs augmented with precise syllable-note alignments and word/sentence/paragraph boundaries. We then provide a quantitative analysis of the correlation between word/sentence/paragraph boundaries in lyrics and melodies. We then propose an RNN-based lyrics language model conditioned on a featurized melody. Experimental results show that the proposed model generates fluent lyrics while maintaining the compatibility between boundaries of lyrics and melody structures.

1 Introduction

Writing lyrics for a given melody is a challenging task. Unlike prose text, writing lyrics requires both knowledge and consideration of music-specific properties such as the structure of melody, rhythms, etc. (Austin et al., 2010; Ueda, 2010). A simple example is the correlation between word boundaries in lyrics and the rests in a melody. As shown in Figure 1, a single word spanning beyond a long melody rest can sound unnatural. When writing lyrics, a lyricist must consider such constraints in content and lexical selection, which can impose extra cognitive loads.

This consideration when writing lyrics has motivated a wide-range of studies for the task of computer-assisted lyrics writing (Barbieri et al., 2012; Abe and Ito, 2012; Potash et al., 2015; Watanabe et al., 2017). Such studies aim to model the

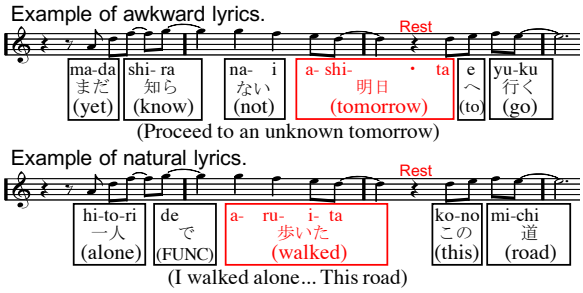


Figure 1: Examples of awkward and natural lyrics. FUNC indicates a function word. The song is from the RWC Music Database (RWC-MDB-P-2001 No.20) (Goto et al., 2002).

language in lyrics and to design a computer system for assisting lyricists in writing. They propose to constrain their models to generate only lyrics that satisfy given conditions on syllable counts, rhyme positions, etc. However, such constraints are assumed to be manually provided by a human user, which requires the user to interpret a source melody and transform their interpretation to a set of constraints. To assist users with transforming a melody to constraints, a language model that automatically captures the relationship between lyrics and melody is required.

Some studies (Oliveira et al., 2007; Oliveira, 2015; Nichols et al., 2009) have quantitatively analyzed the correlations between melody and phonological aspects of lyrics (e.g., the relationship between a beat and a syllable stress). However, these studies do not address the relationship between melody and the *discourse structure* of lyrics. Lyrics are not just a sequence of syllables but a meaningful sequence of words. Therefore, it is desirable that the sentence/paragraph boundaries are determined based on both melody rests and context words.

Considering such line/paragraph structure of lyrics, we present a novel language model that gen-

Proceedings of NAACL-HLT 2018, pages 163–172
New Orleans, Louisiana, June 1 - 6, 2018.

erates lyrics whose word, sentence, and paragraph boundaries are appropriate for a given melody, without manually transforming the melody to syllable constraints. This direction of research has received less attention because it requires a large dataset consisting of aligned pairs of melody and segment boundaries of lyrics which has yet to exist.

To address this issue, we leverage a publicly-available collection of digital music scores and create a dataset of digital music scores each of which specifics a melody score augmented with syllable information for each melody note. We collected 1,000 Japanese songs from an online forum where many amateur music composers upload their music scores. We then automatically aligned each music score with the raw text data of the corresponding lyrics in order to augment it with the word, sentence, and paragraph boundaries.

The availability of such aligned, parallel data opens a new area of research where one can conduct a broad range of data-oriented research for investigating and modeling correlations between melodies and discourse structure of lyrics. In this paper, with our melody-lyrics aligned songs, we investigate the phenomena that (i) words, sentences, and paragraphs rarely span beyond a long melody rest and (ii) the boundaries of larger components (i.e., paragraphs) tend to coincide more with longer rests. To the best of our knowledge, there is no previous work that provides any quantitative analysis of this phenomenon with this size of data (see Section 7).

Following this analysis, we build a novel, data-driven language model that generates fluent lyrics whose sentence and paragraph boundaries fit an input melody. We extend a Recurrent Neural Network Language Model (RNNLM) (Mikolov et al., 2010) so that its output can be conditioned on a featurized melody. Both our quantitative and qualitative evaluations show that our model captures the consistency between melody and boundaries of lyrics while maintaining word fluency.

2 Melody-lyric alignment data

Our goal is to create a melody-conditioned language model that captures the correlations between melody patterns and discourse segments of lyrics. The data we need for this purpose is a collection of melody-lyrics pairs where the melody and lyrics are aligned at the level of not only note-syllable alignment but also discourse components

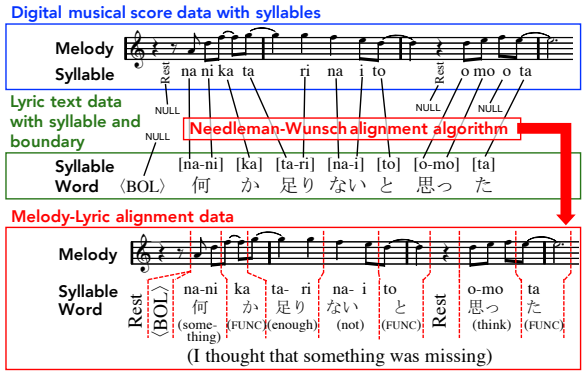


Figure 2: Melody-lyrics alignment using the Needleman Wunsch algorithm. BOL denotes a line boundary.

(i.e., word/sentence/paragraph boundaries) of a lyric, as illustrated in the bottom of Figure 2. We create such a dataset by automatically combining two types of data available from online forum sites: digital music score data (the top of Figure 2) and raw lyrics data (the middle).

A digital music score specifies a melody score augmented with syllable information for each melody note (see the top of Figure 2). Score data augmented in this way is sufficient for analyzing the relationship between the phonological aspects of lyrics and melody, but it is insufficient for our goal since the structural information of the lyrics is not included. We thus augment score data further with boundaries of sentences, and paragraphs, where we assume that sentences and paragraphs of lyrics are approximately captured by *lines* and *blocks*,[1] respectively, of the lyrics in the raw text.

The integration of music scores and raw lyrics is achieved by (1) applying a morphological analyzer[2] to raw lyrics for word segmentation and Chinese character pronunciation prediction and (2) aligning music score with raw lyrics at the syllable level as illustrated in Figure 2. For this alignment, we employ the Needleman-Wunsch algorithm (Needleman and Wunsch, 1970). This alignment process is reasonably accurate because it fails in principle only when the morphological analysis fails in Chinese character pronunciation prediction, which occurs for only less than 1% of the words in the data set.

With this procedure, we obtained 54,181 Japanese raw lyrics and 1,000 digital musical

[1] Blocks are assumed to be segmented by empty lines.

[2] To extract word boundaries and syllable information for Japanese lyrics, we apply MeCab parser (Kudo et al., 2004).

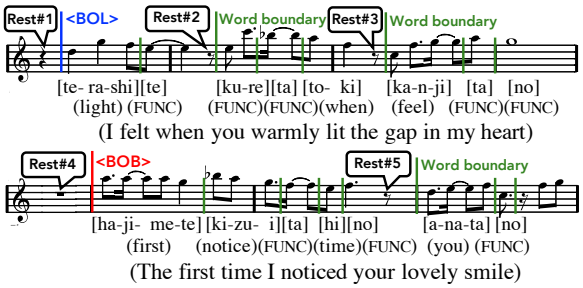


Figure 3: Example boundaries appearing immediately after a rest. BOB indicates a block boundary.

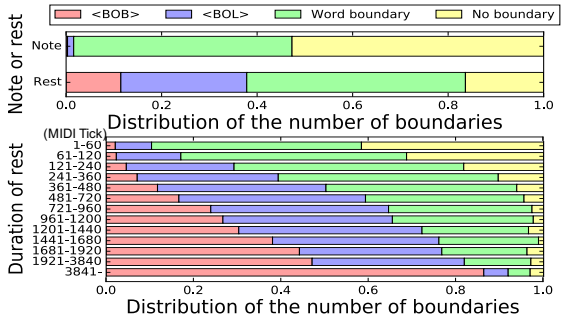


Figure 4: Distribution of the number of boundaries in the melody-lyrics alignment data.

scores from online forum sites[3]; we thus created 1,000 melody-lyrics pairs. We refer to these 1,000 melody-lyrics pairs as a *melody-lyrics alignment* data[4] and refer to the remaining 53,181 lyrics without melody as a *raw lyrics* data.

We randomly split the 1,000 melody-lyrics alignments into two sets: 90% for analyzing/training and the remaining 10% for testing. From those, we use 20,000 of the most frequent words whose syllable counts are equal to or less than 10, and converted others to a special symbol ⟨unknown⟩. All of the digital music score data we collected were distributed in the UST format, a common file format designed specifically for recently emerging computer vocal synthesizers. While we focus on Japanese music in this study, our method for data creation is general enough to be applied to other language formats such as MusicXML and ABC, because transferring such data formats to UST is straightforward.

[3]For selecting the 1,000 songs, we chose only frequently downloaded or highly popular songs to ensure the quality of the resulting dataset.

[4]We publicly release all source URLs of the 1,000 songs (https://github.com/KentoW/melody-lyrics).

3 Correlations between melody and lyric

In this section, we examine two phenomena related to boundaries of lyrics: (1) the positions of lyrics segment boundaries are biased to melody rest positions, and (2) the probability of boundary occurrence depends on the duration of a rest, i.e., a shorter rest tends to be a word boundary and a longer rest tends to be a block boundary, as shown in Figure 3. All analyses were performed on the training split of the melody-lyrics alignment data, which is described in Section 2.

For the first phenomenon, we first calculated the distribution of boundary appearances at the positions of melody notes and rests. Here, by the *boundary of a line* (or block), we refer to the position of the beginning of the line (or block).[5] In Figure 3, we say, for example, that the boundary of the first block beginning "*te-ra-shi te*" coincides with Rest#1. The result, shown at the top of Figure 4, indicates that line and block boundaries are strongly biased to rest positions and are far less likely to appear at note positions. Words, lines, and blocks rarely span beyond a long melody rest.

The bottom of Figure 4 shows the detailed distributions of boundary occurrences for different durations of melody rests, where durations of 480 and 1920 correspond to a quarter rest and a whole rest, respectively. The results exhibit a clear, strong tendency that the boundaries of larger segments tend to coincide more with longer rests. To the best of our knowledge, this is the first study that has ever provided such strong empirical evidence for the phenomena related to the correlations between lyrics segments and melody rests. It is also important to note that the choice of segment boundaries looks like a probabilistic process (i.e., there is a long rest without a block boundary). This observation suggests the difficulty of describing the correlations of lyrics and melody in a rule-based fashion and motivates our probabilistic approach as we present in the next section.

4 Melody-conditioned language model

Our goal is to build a language model that generates fluent lyrics whose discourse segment fit a given melody in the sense that generated segment boundaries follow the distribution observed in Section 3. We propose to pursue this goal by conditioning a

[5]The beginning of a line/block and the end of a line/block are equivalent since there is no melody between the end and beginning of a line/block.

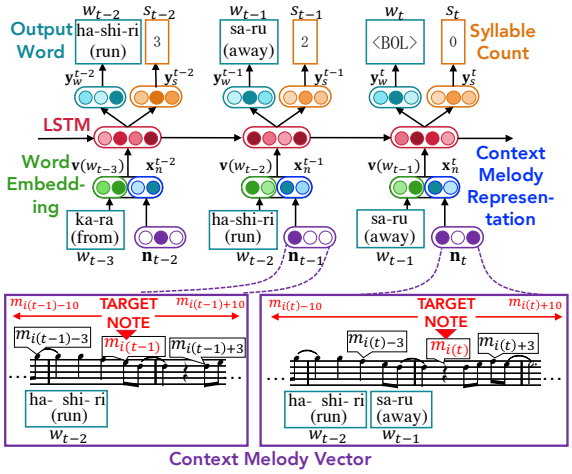


Figure 5: Melody-conditioned RNNLM.

standard RNNLM with a featurized input melody. We call this model a *Melody-conditioned RNNLM*.

The network structure of the model is illustrated in Figure 5. Formally, we are given a melody $\mathbf{m} = m_1,...,m_i,...,m_I$ that is a sequence of notes and rests, where m includes a pitch and a duration information. Our model generates lyrics $\mathbf{w} = w_1,...,w_t,...,w_T$ that is a sequence of words and segment boundary symbols: ⟨BOL⟩ and ⟨BOB⟩, special symbols denoting a line and a block boundary, respectively. For each time step t, the model outputs a single word or boundary symbol taking a pair of the previously generated word w_{t-1} and the musical feature vector $\mathbf{n}_t$ for the current word position which includes context window-based features that we describe in the following section. In this model, we assume that the syllables of the generated words and the notes in the input melody have a one-to-one correspondence. Therefore, the position of the incoming note/rest for a word position t (referred to as a target note for t) is uniquely determined by the syllable counts of the previously generated words.[6] The target note for t is denoted as $m_{i(t)}$ by defining a function $i(\cdot)$ which maps time step t to the index of the next note in t.

Here, the challenging issue with this model is training. Generally, language models require a large amount of text data to learn well. Moreover, this is also the case for learning correlation between rest positions and *syllable counts*. As shown in Figure 4, most words are supposed to not overlap a long rest. This means, for example, that when the incoming melody sequence for a next word position is *note, note, (long) rest, note, note*, as the sequence following to $m_{i(t-1)}$ in Figure 5, it is desirable to select a word whose syllable count is two or less so that the generated word does not overlap the long rest. If there is sufficient data available, this tendency may be learned directly from the correlation between rests and words without explicitly considering the syllable count of a word. However, our melody-lyrics alignments for 1,000 songs are insufficient for this purpose.

We take two approaches to address this data sparsity problem. First, we propose two training strategies that increase the number of training examples using raw lyrics that can be obtained in greater quantities. Second, we construct a model that predicts the number of syllables in each word, as well as words themselves, to explicitly supervise the correspondence between rest positions and syllable counts.

In the following sections, we first describe the details of the proposed model and then present the training strategies used to obtain better models with our melody-lyrics alignment data.

4.1 Model construction

The proposed model is based on a standard RNNLM (Mikolov et al., 2010):

$$P(\mathbf{w}) = \prod_{t=1}^{T} P(w_t|w_0, ..., w_{t-1}), \quad (1)$$

where context words are encoded using LSTM (Hochreiter and Schmidhuber, 1997) and the probabilities over words are calculated by a softmax function. $w_0 = \langle \mathrm{B} \rangle$ is a symbol denoting the beginning of lyrics. We extend this model such that each output is conditioned by the context melody vectors $\mathbf{n}_1, ..., \mathbf{n}_t$, as well as previous words:

$$P(\mathbf{w}|\mathbf{m}) = \prod_{t=1}^{T} P(w_t|w_0, ..., w_{t-1}, \mathbf{n}_1, ..., \mathbf{n}_t). \quad (2)$$

The model simultaneously predicts the syllable counts of words by sharing the parameters of LSTM with the above word prediction model in order to learn the correspondence between the melody segments and syllable counts:

$$P(\mathbf{s}|\mathbf{m}) = \prod_{t=1}^{T} P(s_t|w_0, ..., w_{t-1}, \mathbf{n}_1, ..., \mathbf{n}_t), \quad (3)$$

where $\mathbf{s} = s_1, ..., s_T$ is a sequence of syllable counts, which corresponds to $\mathbf{w}$.

[6] Note that our melody-lyrics alignment data used in training does not make this assumption, but we can still uniquely identify the positions of target notes based on the obtained melody-word alignment.

For each time step t, the model outputs a word distribution $\mathbf{y}_w^t \in \mathbb{R}^V$ and a distribution of syllable count $\mathbf{y}_s^t \in \mathbb{R}^S$ using a softmax function:

$$\mathbf{y}_w^t = \text{softmax}(\text{BN}(\mathbf{W}_w \mathbf{z}_t)), \quad (4)$$

$$\mathbf{y}_s^t = \text{softmax}(\text{BN}(\mathbf{W}_s \mathbf{z}_t)), \quad (5)$$

where $\mathbf{z}_t$ is the output of the LSTM for each time step. V is the vocabulary size and S is the syllable count threshold.[7] $\mathbf{W}_w$ and $\mathbf{W}_s$ are weight matrices. BN denotes batch normalization (Ioffe and Szegedy, 2015).

The input to the LSTM in each time step t is a concatenation of the embedding vector of the previous word $\mathbf{v}(w_{t-1})$ and the context melody representation $\mathbf{x}_n^t$, which is a nonlinear transformation of the context melody vector $\mathbf{n}_t$:

$$\mathbf{x}^t = [\mathbf{v}(w_{t-1}), \mathbf{x}_n^t], \quad (6)$$

$$\mathbf{x}_n^t = \text{ReLU}(\mathbf{W}_n \mathbf{n}_t + \mathbf{b}_n), \quad (7)$$

where $\mathbf{W}_n$ is a weight matrix and $\mathbf{b}_n$ is a bias.

To generate lyrics, the model searches for the word sequence with the greatest probability (Eq. 2) using beam search. The model stops generating lyrics when the syllable count of the lyrics reaches the number of notes in the input melody.

Note that our model is not specific to the language of lyrics. The model only requires the sequences of melody, words, and syllable counts and does not use any language-specific features.

4.2 Context melody vector

In Section 3, we indicated that the positions of rests and their durations are important factors for modeling boundaries of lyrics. Thus, we collect a sequence of notes and rests around the current word position (i.e., time step t) and encode their information into context melody vector $\mathbf{n}_t$ (see the bottom of Figure 5).

The context melody vector $\mathbf{n}_t$ is a binary feature vector that includes a musical notation type (i.e., note or rest), a duration[8], and a pitch for each note/rest in the context window. We collect notes and rests around the target note $m_{i(t)}$ for the current word position t with a window size of 10 (i.e., $m_{i(t)-10}, ..., m_{i(t)}, ..., m_{i(t)+10}$).

For pitch information, we use a gap (pitch interval) between a target note $m_{i(t)}$ and its previous note $m_{i(t-1)}$. Here, the pitch is represented by a MIDI note number in the range 0 to 127. For example, the target and its previous notes are 68 and 65, respectively, and the gap is $+3$.

[7] The syllable counts of the ⟨BOL⟩ and ⟨BOB⟩ are zero.

[8] We rounded each duration to one of the values 60, 120, 240, 360, 480, 720, 960, 1200, 1440, 1680, 1920, and 3840 and use one-hot encoding for each rounded duration.

Algorithm 1 Pseudo melody generation

```
for each syllable in the input-lyrics do
    b ← get boundary type next to the syllable
    sample note pitch p ~ P(p_i | p_{i-2}, p_{i-1})
    sample note duration d_note ~ P(d_note | b)
    assign note with (p, d_note) to the syllable
    sample binary variable r ~ P(r | b)
    if r = 1 then
        insert rest with duration d_rest ~ P(d_rest | b)
    end if
end for
```

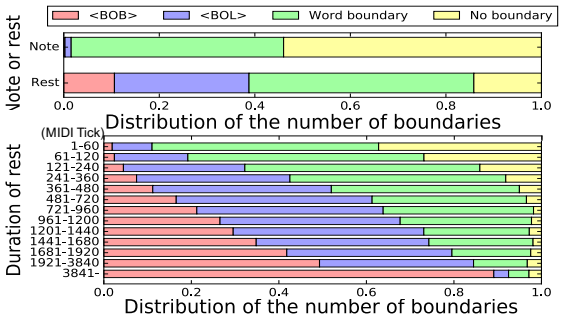


Figure 6: Distribution of the number of boundaries in pseudo-data.

4.3 Training strategies

Pretraining The size of our melody-lyrics alignment data is limited. However, we can obtain a large amount of raw lyrics. We, therefore, pretrain the model with 53,181 raw lyrics and then fine-tune it with the melody-lyrics alignment data. In pretraining, all context melody vectors $\mathbf{n}_t$ are zero vectors. We refer to these pretrained and fine-tuned models as *Lyrics-only* and *Fine-tuned* models, respectively.

Learning with pseudo-melody We propose a method to increase the melody-lyrics alignment data by attaching *pseudo melodies* to the obtained 53,181 raw lyrics. We refer to the model that uses this data as the *Pseudo-melody* model.

Algorithm 1 shows the details of pseudo-melody generation. For each syllable in the lyrics, we first assign a note to the syllable by sampling the probability distributions. The pitch of each note is generated based on the trigram probability. Then, we determine whether to generate a rest next to it. Since we established the correlations between rests and boundaries of lyrics in Section 3, the probability for a rest and its duration is conditioned by a boundary

type next to the target syllable. All probabilities are calculated using the training split of the melody-lyrics alignment data.

Figure 6 shows the distributions of the number of boundaries in the pseudo data. The distributions closely resemble those of gold data in Figure 4.

5 Quantitative evaluation

We evaluate the proposed Melody-conditioned RNNLMs quantitatively based on two evaluation metrics: (1) a test set perplexity for measuring the fluency; (2) a line/block boundary replication task for measuring the consistency between the melody and boundaries in the generated lyrics.

5.1 Experimental setup

In our model, we chose the dimensions of the word embedding vectors and context melody representation vectors to 512 and 256, respectively, and the dimension of the LSTM hidden state was 768. We used a categorical cross-entropy loss for outputs $\mathbf{y}_w^t$ and $\mathbf{y}_s^t$, Adam (Kingma and Ba, 2014) with an initial learning rate of 0.001 for parameter optimization, and a mini-batch size of 32. We applied an early-stopping strategy with a maximum epoch number of 100, and training was terminated after five epochs of unimproved loss on the validation set. For lyrics generation, we used a beam search with a width of 10. An example of the generated lyrics is shown in the supplemental material.

5.2 Evaluation metrics

Perplexity Test-set perplexity (PPL) is a standard evaluation measure for language models. PPL measures the predictability of wording in original lyrics, where a lower PPL value indicates that the model can generate fluent lyrics. We used PPL and its variant PPL-W, which excludes line/block boundaries, to investigate the predictability of words.

Accuracy of boundary replication Under the assumption that the line and block boundaries of the original lyrics are placed at appropriate positions in the melody, we evaluated consistency between the melody and boundaries in the generated lyrics by measuring the reproducibility of the boundaries in the original lyrics. Here the metric we used was F_1-measure of the boundary positions. We also asked a person to place line and block boundaries at plausible positions for randomly selected 10 input melodies that the evaluator has never heard. This person is not a professional musician but an experienced performer educated on musicology. The bottom part of Table 1 represents the human performance.

Model	Perplexity PPL	Perplexity PPL-W	F_1-measure BOB	F_1-measure BOL	F_1-measure UB
Lyrics-only	138.0	225.0	0.121	0.061	0.106
Full-data	135.9	222.1	0.122	0.063	0.108
Alignment-only	173.3	314.8	0.298	0.287	0.477
Heuristic	175.8	284.7	**0.373**	0.239	0.402
Fine-tuned	152.2	275.5	0.260	**0.302**	**0.479**
Pseudo-melody	**115.7**	**197.5**	0.318	0.241	0.406
(w/o $\mathbf{y}_s$)					
Fine-tuned	155.1	278.1	0.318	0.241	0.366
Pseudo-melody	118.0	201.5	0.312	0.250	0.406
Human	-	-	0.717	0.671	0.751

Table 1: Results of the quantitative evaluation. "UB" denotes the score for unlabeled matching of line/block boundaries. "w/o $\mathbf{y}_s$" denotes the exclusion of the syllable-count output layer.

5.3 Effect of Melody-conditioned RNNLM

To investigate the effect of our language models, we compared the following six models. The first one is (1) a *Lyrics-only* model, a standard RNNLM trained with 54,081 song lyrics without melody information. The second and third ones are baseline Melody-conditioned RNNLMs where the proposed training strategies are not applied: (2) a *Full-data* model trained with mixed data (54,081 song lyrics and 900 melody-lyrics alignments of those), and (3) an *Alignment-only* model trained with only 900 melody-lyrics alignment data. The fourth one is a strong baseline to evaluate the performance of the proposed approaches: (4) a *Heuristic* model that (i) assigns a line/block boundary to a rest based on its duration with the same probability, as reported in Figure 4, and (ii) fills the space between any two boundaries with lyrics of the appropriate syllable counts. This *Heuristic* model computes the following word probability:

$$P(w_t|w_0,...,w_{t-1},\mathbf{m}) = \quad (8)$$

$$\begin{cases} Q(\langle \mathrm{BOB}\rangle|m_{i(t+1)}) & (\text{if } w_t = \langle \mathrm{BOB}\rangle) \\ Q(\langle \mathrm{BOL}\rangle|m_{i(t+1)}) & (\text{if } w_t = \langle \mathrm{BOL}\rangle) \\ (1 - Q(\langle \mathrm{BOB}\rangle|m_{i(t+1)}) - Q(\langle \mathrm{BOL}\rangle|m_{i(t+1)}))\times & \\ \frac{P_{\mathrm{LSTM}}(w_t|w_0,...,w_{t-1})}{1-P_{\mathrm{LSTM}}(\langle \mathrm{BOL}\rangle|w_0,...,w_{t-1})-P_{\mathrm{LSTM}}(\langle \mathrm{BOB}\rangle|w_0,...,w_{t-1})} & \\ \qquad (\text{otherwise}) \end{cases}$$

where Q is the same probability as reported in Figure 4. P_{LSTM} is the word probability calculated by a standard LSTM language model. The remaining two are Melody-conditioned RNNLMs with the proposed learning strategies: (5) *Fine-tuned* and (6) *Pseudo-melody* models.

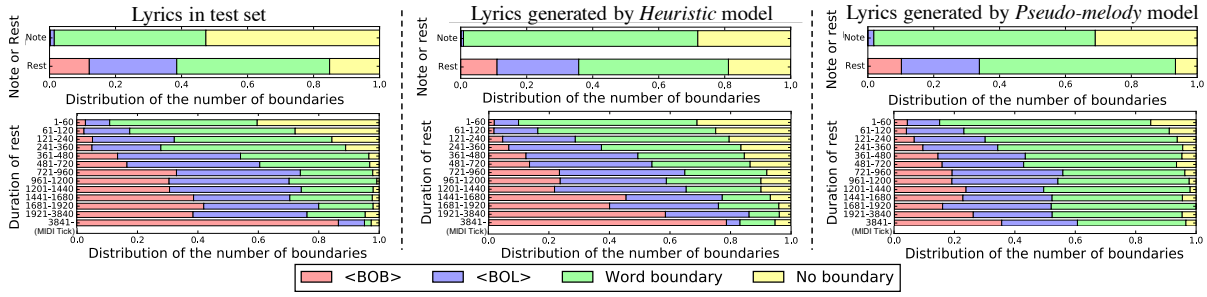


Figure 7: Distribution of the number of boundaries in the test set and lyrics generated by the *Heuristic* and *Pseudo-melody* models.

The top part of Table 1 summarizes the performance of these models. Regarding the boundary replication, the *Heuristic*, *Alignment-only*, *Fine-tuned*, and *Pseudo-melody* models achieved higher performance than the *Lyrics-only* model for unlabeled matching of line/block boundaries (i.e., UB). This result indicates that our Melody-conditioned RNNLMs successfully capture the consistency between melody and boundaries of lyrics. The results of the *Full-data* model is low (as expected) because the size of the melody-lyrics alignment data is far smaller than that of the raw lyrics data and this harms the learning process of the dependency between melody and lyrics. For the block boundary, the *Heuristic* model achieved the best performances. For the line boundary, the *Fine-tuned* model achieved the best performances.

Regarding PPL and PPL-W, the *Lyrics-only*, *Full-data*, and *Pseudo-melody* models show better results than the other models. The *Fine-tuned* model shows reduced performance compared with the *Lyrics-only* model because fine-tuning with a small amount of data causes overfitting in the language model. Also, the training size of the *Alignment-only* model is insufficient for learning a language model of lyrics. Interestingly, the *Pseudo-melody* model achieved better performance than the *Full-data* model and overall achieved the best score. This result indicates that the *Pseudo-melody* model uses the information of a given melody to make a better prediction of its lyrics word sequence. On the other hand, the *Heuristic* model had the worst performance, despite training with a large amount of raw lyrics. We analyze the reason for such performance and describe our results in Section 5.5. It is not necessarily clear which to choose, either the *Fine-tuned* or *Pseudo-melody* model, which may depend also on the size and diversity of the training and test data. However, one can conclude at least that combining a limited-scale collection of melody-lyrics alignment data with a far larger collection of lyrics-alone data boosts the model's capability of generating a fluent lyrics which structurally fits well the input melody.

5.4 Effect of predicting syllable-counts

To investigate the effect of predicting syllable-counts, we compared the performance of the proposed models to models that exclude the syllable-count output layer $\mathbf{y}_s$. The middle part of Table 1 summarizes the results. For the pretraining strategy, the use of $\mathbf{y}_s$ successfully alleviates data sparsity when learning the correlation between syllable counts and melodies from only words themselves. As can be seen, the model without $\mathbf{y}_s$ shows reduced performance relative to both PPLs and the boundary replication. On the other hand, for the pseudo-melody strategy, the two models are competitive in both measures. This means that the *Pseudo-melody* model obtained a sufficient amount of word-melody input pairs to learn the correlation.

5.5 Analysis of melody and generated lyrics

To examine whether the models can capture correlations between rests and boundaries of lyrics, we calculate the proportion of the word, line, and block boundaries in the original lyrics and in the lyrics generated by the *Heuristic* and *Pseudo-melody* model for the test set (Figure 7). The proportion of ⟨BOL⟩ and ⟨BOB⟩ generated by the *Heuristic* model are almost equivalent to those of the original lyrics. On the other hand, for the *Pseudo-melody* model, the proportion of line/block boundary types for the longer rests are smaller than that of the original lyrics.

Although the *Heuristic* model reproduces the proportion of the original line/block boundaries, the model had a low performance in terms of PPL,

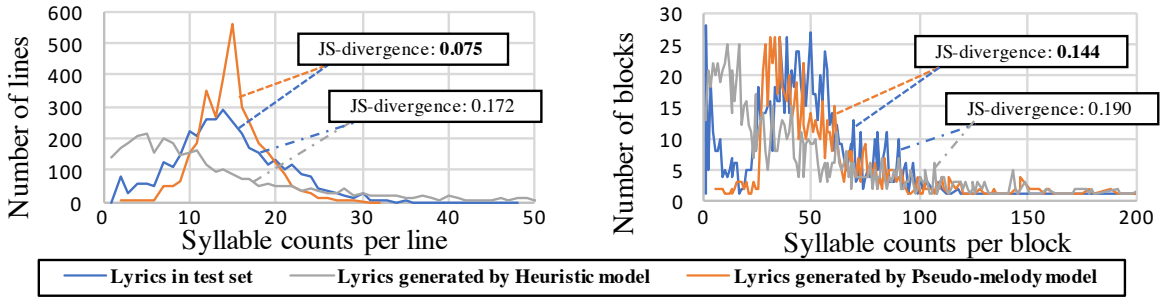


Figure 8: Distribution of the syllable count of the generated lines/blocks

	Heuristic		*Lyrics-only*		*Fine-tuned*		*Pseudo-melody*		*Human* (Upper-bound)	
Measure	Means ± SD	Median	Means ± SD	Median	Means ± SD	Median	Means ± SD	Median	Means ± SD	Median
L	2.06±1.08	2	2.33±1.23	2	**2.85**±1.20	3	**2.93**±1.14	3	3.56±1.33	4
G	2.28±1.07	2	**2.81**±1.16	3	2.79±1.06	3	**2.97**±1.08	3	3.50±1.25	4
LM	2.34±1.07	2	**2.91**±1.15	3	2.70±1.13	3	**2.96**±1.09	3	3.49±1.35	4
DM	2.33±1.10	2	**2.80**±1.06	3	2.59±1.11	3	**2.89**±1.07	3	3.49±1.30	4
OQ	2.01±1.01	2	2.59±1.15	3	2.42±1.08	2	**2.65**±1.01	3	3.32±1.19	4

Table 2: Results of the qualitative evaluation.

as shown in Section 5.3. By investigating the lyrics generated by the *Heuristic* model, we found that the model tends to generate line/block boundaries after the melody rest, even if the two rests are quite close. Figure 8 shows the distributions of the syllable per line / block frequency and the distributions of the Jensen-Shannon divergence. While the Heuristic model tends to generate short lines/blocks, our model generates the lyrics so that lines/blocks do not become too short. This result supports that (i) our model is trained using melody and lyric contexts and (ii) the heuristic approach, which simply generates line/block boundaries based on the distribution in Figure 4, cannot generate fluent lyrics with well-formed line/block lengths.

6 Qualitative evaluation

To asses the quality of the generated lyrics, inspired by (Oliveira, 2015), we asked 50 Yahoo crowdsourcing workers to answer the following five questions using a five-point Likert scale:

Listenability (L) When listening to melody and lyrics, are the positions of words, lines, and segments natural? (1=*Poor* to 5=*Perfect*)

Grammaticality (G) Are the lyrics grammatically correct? (1=*Poor* to 5=*Perfect*)

Line-level meaning (LM) Is each line in the lyrics meaningful? (1=*Unclear* to 5=*Clear*)

Document-level meaning (DM) Are the entire lyrics meaningful? (1=*Unclear* to 5=*Clear*)

Overall quality (OQ) What is the overall quality of the lyrics? (1=*Terrible* to 5=*Great*)

For the evaluation sets, we randomly selected four melodies from the RWC Music Database (Goto et al., 2002). For each melody, we prepared four lyrics generated by the *Heuristic*, *Lyrics-only*, *Fine-tuned*, and *Pseudo-melody* models. Moreover, to obtain an upper bound for this evaluation, we used the lyrics created by amateur writers: we asked four native Japanese speakers to write lyrics on the evaluation melody. One writer was a junior high school teacher of music who had experience in music composition and writing lyrics. Three writers were graduate students with different levels of musical expertise. Two of the three writers had experience with music composition, but none of them had experience with writing lyrics.[9] As a result, we obtained 50 (workers) × 4 (melodies) × 5 (lyrics) samples in total. We note that workers did not know whether lyrics were created by a human or generated by a computer.

Table 2 shows the average scores, standard deviations, and medians for each measure. Regarding the "Listenability" evaluation, workers gave high scores to the *Fine-tuned* and *Pseudo-melody* models that are trained using both the melody and lyrics. This result is consistent with the perplexity evaluation result. On the other hand, regarding the "Grammaticality" and "Meaning" evaluation, workers gave high scores to the *Lyrics-only* and *Pseudo-melody* models that are well-trained on a large amount of text data. This result is consistent with the result of

[9]We release lyrics and audio files used in the qualitative evaluation on the Web (https://github.com/KentoW/deep-lyrics-examples).

the boundary replication task. Regarding the "Overall quality" evaluation, the *Pseudo-melody* model outperformed all other models. These results indicate our pseudo data learning strategy contributes to generating high-quality lyrics. However, the quality of lyrics automatically generated is still worse than the quality of lyrics that humans produce, and it still remains an open challenge for future research to develop computational models that generate high-quality lyrics.

7 Related work

In the literature, a broad range of research efforts has been reported for computationally modeling lyrics-specific properties such as meter, rhythm, rhyme, stress, and accent Greene et al. (2010); Reddy and Knight (2011); Watanabe et al. (2014, 2016). While these studies provide insightful findings on the properties of lyrics, none of those takes the approach of using melody-lyrics parallel data for modeling correlations of lyrics and melody structures. One exception is the work of Nichols et al. (2009), who used melody-lyrics parallel data to investigate, for example, the correlation between syllable stress and pitch; however, their exploration covers only correlations at the prosody level but not structural correlations.

The same trend can be seen also in the literature of automatic lyrics generation, where most studies utilize only lyrics data. Barbieri et al. (2012) and Abe and Ito (2012) propose a model for generating lyrics under a range of constraints provided in terms of rhyme, rhythm, part-of-speech, etc. Potash et al. (2015) proposes an RNNLM that generates rhymed lyrics under the assumption that rhymes tend to coincide with the end of lines. In those studies, the melody is considered only indirectly; namely, input prosodic/linguistic constraints/preferences on lyrics are assumed to be manually provided by a human user because the proposed models are not capable of interpreting and transforming a given melody to constraints/preferences.

For generating lyrics for a given melody, we have so far found in the literature two studies which propose a method. Oliveira et al. (2007) and Oliveira (2015) manually analyze correlations among melodies, beats, and syllables using 42 Portuguese songs and propose a set of heuristic rules for lyrics generation. Ramakrishnan A et al. (2009) attempt to induce a statistical model for generating melodic Tamil lyrics from melody-lyrics parallel data using only ten songs. However, the former captures only phonological aspects of melody-lyrics correlations and can generate a small fragment of lyrics (not an entire lyrics) for a given piece of melody. The latter suffers from the severe shortage of data and fails to conduct empirical experiments.

8 Conclusion and future work

This paper has presented a novel data-driven approach for building a melody-conditioned lyrics language model. We created a 1,000-song melody-lyrics alignment dataset and conducted a quantitative investigation into the correlations between melodies and segment boundaries of lyrics. No prior work has ever conducted such a quantitative analysis of melody-lyrics correlations with this size of data. We have also proposed a RNN-based, melody-conditioned language model that generates fluent lyrics whose word/line/block boundaries fit a given input melody. Our experimental results have shown that: (1) our Melody-conditioned RNNLMs capture the consistency between melody and boundaries of lyrics while maintaining word fluency; (2) combining a limited-scale collection of melody-lyrics alignment data with a far larger collection of lyrics-alone data for training the model boosts the model's competence; (3) we have also produced positive empirical evidence for the effect of applying a multi-task learning schema where the model is trained for syllable count prediction as well as for word prediction; and (4) the human judgments collected via crowdsourcing showed that our model improves the quality of generated lyrics.

For future directions, we plan to further extend the proposed model for capturing other aspects of lyrics/melody discourse structure such as repetitions, verse-bridge-chorus structure, and topical coherence of discourse segment. The proposed method for creating melody-lyrics alignment data enables us to explore such a broad range of aspects of melody-lyrics correlations.

Acknowledgments

This study utilized the RWC Music Database (Popular Music). This work was partially supported by a Grant-in-Aid for JSPS Research Fellow Grant Number JP16J05945, JSPS KAKENHI Grant Numbers JP15H01702, and JST ACCEL Grant Number JPMJAC1602. The authors would like to thank Dr. Paul Reisert for the English language review.

References

Chihiro Abe and Akinori Ito. 2012. A Japanese lyrics writing support system for amateur songwriters. In *Proceedings of Asia-Pacific Signal & Information Processing Association Annual Summit and Conference 2012 (APSIPA ASC 2012)*. pages 1–4.

Dave Austin, Jim Peterik, and Cathy Lynn Austin. 2010. *Songwriting for Dummies*. Wileys.

Gabriele Barbieri, François Pachet, Pierre Roy, and Mirko Degli Esposti. 2012. Markov constraints for generating lyrics with style. In *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI 2012)*. pages 115–120.

Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. 2002. RWC Music Database: Popular, classical and jazz music databases. In *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR 2002)*. volume 2, pages 287–288.

Erica Greene, Tugba Bodrumlu, and Kevin Knight. 2010. Automatic analysis of rhythmic poetry with applications to generation and translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP 2010)*. pages 524–533.

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

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*. volume 37, pages 448–456.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint* arXiv:1412.6980.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to Japanese morphological analysis. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*. pages 230–237.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of Interspeech 2010*. pages 1045–1048.

Saul B. Needleman and Christian D. Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology* 48(3):443–453.

Eric Nichols, Dan Morris, Sumit Basu, and Christopher Raphael. 2009. Relationships between lyrics and melody in popular music. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*. pages 471–476.

Hugo Gonçalo Oliveira. 2015. Tra-la-lyrics 2.0: Automatic generation of song lyrics on a semantic domain. *Journal of Artificial General Intelligence* 6(1):87–110.

Hugo R. Gonçalo Oliveira, F. Amialcar Cardoso, and Francisco C. Pereira. 2007. Tra-la-lyrics: An approach to generate text based on rhythm. In *Proceedings of the 4th International Joint Workshop on Computational Creativity*. pages 47–55.

Peter Potash, Alexey Romanov, and Anna Rumshisky. 2015. GhostWriter: Using an LSTM for automatic Rap lyric generation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*. pages 1919–1924.

Ananth Ramakrishnan A, Sankar Kuppan, and Sobha Lalitha Devi. 2009. Automatic generation of Tamil lyrics for melodies. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*. pages 40–46.

Sravana Reddy and Kevin Knight. 2011. Unsupervised discovery of rhyme schemes. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2011)*. pages 77–82.

Tatsuji Ueda. 2010. よくわかる作詞の教科書 *The writing lyrics textbook which is easy to understand (in Japanese)*. YAMAHA music media corporation.

Kento Watanabe, Yuichiroh Matsubayashi, Kentaro Inui, and Masataka Goto. 2014. Modeling structural topic transitions for automatic lyrics generation. In *Proceedings of the 28th Pacific Asia Conference on Language, Information and Computation (PACLIC 2014)*. pages 422–431.

Kento Watanabe, Yuichiroh Matsubayashi, Kentaro Inui, Tomoyasu Nakano, Satoru Fukayama, and Masataka Goto. 2017. LyriSys: An Interactive support system for writing lyrics based on topic transition. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces (ACM IUI 2017)*. pages 559–563.

Kento Watanabe, Yuichiroh Matsubayashi, Naho Orita, Naoaki Okazaki, Kentaro Inui, Satoru Fukayama, Tomoyasu Nakano, Jordan Smith, and Masataka Goto. 2016. Modeling discourse segments in lyrics using repeated patterns. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING 2016)*. pages 1959–1969.

Discourse-Aware Neural Rewards for Coherent Text Generation

Antoine Bosselut[1]*, **Asli Celikyilmaz**[2], **Xiaodong He**[3],
Jianfeng Gao[2], **Po-Sen Huang**[2] and **Yejin Choi**[1,4]
[1]Paul G. Allen School of Computer Science & Engineering, University of Washington
[2]Microsoft Research
[3]JD AI Research
[4]Allen Institute for Artificial Intelligence
{antoineb,yejin}@cs.washington.edu {xiaodong.he}@jd.com
{aslicel,jfgao,pshuang}@microsoft.com

Abstract

In this paper, we investigate the use of discourse-aware rewards with reinforcement learning to guide a model to generate long, coherent text. In particular, we propose to learn neural rewards to model cross-sentence ordering as a means to approximate desired discourse structure. Empirical results demonstrate that a generator trained with the learned reward produces more coherent and less repetitive text than models trained with cross-entropy or with reinforcement learning with commonly used scores as rewards.

1 Introduction

Defining an ideal loss for training text generation models remains an open research question. Many existing approaches based on variants of recurrent neural networks (Hochreiter and Schmidhuber, 1997; Cho et al., 2014) are trained using cross-entropy loss (Bahdanau et al., 2015; Vinyals et al., 2015; Xu et al., 2015; Rush et al., 2015), often augmented with additional terms for topic coverage or task-specific supervision (Kiddon et al., 2016; Yang et al., 2017).

Training with cross-entropy, however, does not always correlate well with achieving high scores on commonly used evaluation measures such as ROUGE (Lin, 2004), BLEU (Papineni et al., 2002), or CIDEr (Vedantam et al., 2015). Another current line of research therefore explores training generation models that directly optimize the target evaluation measure (Wu et al., 2016; Ranzato et al., 2015; Paulus et al., 2018; Rennie et al., 2017) using reinforcement learning methods such as the REINFORCE algorithm (Williams, 1992).

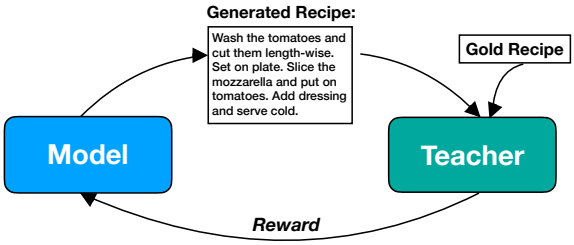


Figure 1: The generator is rewarded for imitating the discourse structure of the gold sequence.

Importantly, most automatic measures are based on local n-gram patterns, providing only a limited and myopic perspective of overall text quality. As a result, while models trained to directly optimize these measures can yield improvements on the same measures, they may not lead to better quality in terms of overall coherence or discourse structure. Indeed, recent studies have reported cases where commonly used measures do not align well with desired aspects of generation quality (Rennie et al., 2017; Li et al., 2016).

The challenge, however, is to define a global score that can measure the complex aspects of text quality beyond local n-gram patterns. In this paper, we investigate *learning* neural rewards and their use in a reinforcement learning regime with a specific focus on learning more discourse-aware and coherent text generation. Our approach shares the spirit of the work of Lowe et al. (2017), where neural scores were learned to approximate human judgments of dialogue quality. The key difference is that our rewards can be fully automatically constructed without requiring human judgments and can be trained in an unsupervised manner.

More specifically, we propose a neural reward learning scheme that is trained to capture cross-sentence ordering structure as a means to approximate the desired discourse structure in documents. The learned *teacher* computes rewards for the

* Work done while author was at Microsoft Research

Proceedings of NAACL-HLT 2018, pages 173–184
New Orleans, Louisiana, June 1 - 6, 2018.

underlying text generator (see Figure 1), which is trained using self-critical reinforcement learning (Rennie et al., 2017). We also present a new method for distributing sentence-level rewards for more accurate credit assignment.

We test our approach on the task of generating cooking recipes, and evaluate using automatic overlap metrics that measure discourse structure. We also provide human judgments that yield comprehensive insights into the model behavior induced by the learned neural rewards. Empirical results demonstrate that a generator trained with the discourse-aware rewards produces text that is more coherent and less repetitive than models trained with cross-entropy or reinforcement learning with other commonly used scores.

2 Neural Teachers

Recent work in image captioning (Rennie et al., 2017), machine translation (Wu et al., 2016), and summarization (Paulus et al., 2018) has investigated using policy gradient methods to fine-tune neural generation models using automatic measures such as CIDEr as the reward. However, because most existing automatic measures focus on local n-gram patterns, fine-tuning on those measures may yield deteriorated text despite increased automatic scores, especially for tasks that require long coherent generation (§6.1).

Since writing out a scoring term that quantifies the quality of discourse coherence is an open research question, we take inspiration from previous research that *learns* the overall ordering structure of a document as an approximation of the discourse structure (Barzilay and Lapata, 2005, 2008; Barzilay and Lee, 2004; Li and Hovy, 2014), and propose two neural teachers that can learn to score an ordered sequence of sentences. The scores from these neural teachers are then used to formulate rewards (§4.2) that guide coherent long text generation systems in a policy gradient reinforcement learning setup. Notably, the neural teachers are trained offline on gold sequences in an unsupervised manner prior to training the generator. They are not trained jointly with the generator and their parameters are fixed during policy learning.

2.1 Notation

We define a document of n sentences as $S = \{s_0, ..., s_n\}$ where each sentence s_j has L_j words.

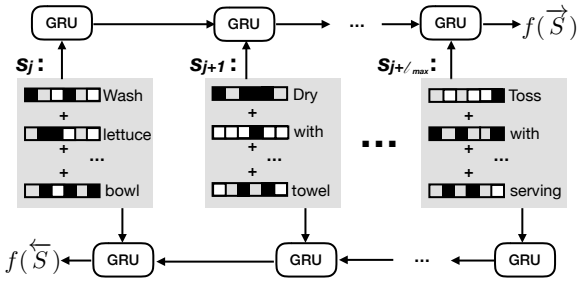


Figure 2: The teacher encodes the sentences of the document in the forward and reverse order.

2.2 Absolute Order Teacher

The first teacher explored is motivated by work on deep semantic similarity models (Huang et al., 2013), which approximated the similarity between queries and documents in information retrieval tasks. We extend this approach to modeling temporal patterns by training a sentence encoder to minimize the similarity between a sequence encoded in its forward order, and the same sequence encoded in the reverse order (see Figure 2).

To focus the teacher on discourse structure, we design the encoder to capture *sentence order*, instead of *word order*. Words in each sentence s_j are encoded using a bag of words:

$$\mathbf{s}_j = \sum_{i=1}^{L_j} x_{ij} \tag{1}$$

where x_{ij} is a word embedding and $\mathbf{s}_j$ is a sentence embedding. Each $\mathbf{s}_j$ is passed to a gated recurrent unit (GRU) and the final output of the hidden unit is used as the representation for the full document:

$$h_j = \text{GRU}(\mathbf{s}_j, h_{j-1}) \tag{2}$$

$$f(S) = h_n \tag{3}$$

where $f(S)$ is the representation of the sentences of the document and h_n is the final output vector of the GRU. To capture properties of temporal coherence among document sentences, the teacher is trained to minimize $\mathcal{L}_{abs}$, the cosine similarity between the sentence embedding from reading the sentences in the forward order, $\overrightarrow{S}$ and from reading the sentences in the reverse order, $\overleftarrow{S}$:

$$\mathcal{L}_{abs} = \frac{\langle f(\overrightarrow{S}), f(\overleftarrow{S}) \rangle}{\|f(\overrightarrow{S})\| \|f(\overleftarrow{S})\|} \tag{4}$$

Intuitively, by parametrizing only relations between *sentences* (with the GRU layer) and not those between *words*, the teacher only captures sentence ordering properties. When training the neural generator (§4), we use this learned teacher to generate a reward that judges the generated sequence's ordering similarity to the gold sequence.

2.3 Relative Order Teacher

While the absolute ordering teacher evaluates the temporal coherence of the entire generation, we may want our teacher to be able to judge finer-grained patterns between sentences. In recipes, for example, where sentences correspond to process steps, the teacher should capture implicit script knowledge (Schank and Abelson, 1975) among groups of sentences. Consequently, the teacher should reward sentences individually for how they fit with surrounding sentences.

In many current approaches for using policy gradient methods to optimize a model with respect to a global score, each sentence receives the same reward. This framework assumes each sentence is equally responsible for the reward gathered by the full sequence, allowing potentially appropriate subsequences to be incorrectly penalized. We design the relative order teacher to address this issue.

The relative order teacher is trained in the same way as the absolute order model. A bag of words embedding is computed for each sentence in the gold sequence. Subsequences of the gold document that have ℓ sentences are selected where $\ell \in (\ell_{min}, \ell_{max})$. For a subsequence beginning at sentence j, the model computes:

$$f(S_{j:j+\ell}) = \text{GRU}(\mathbf{s}_{j+\ell}, h_{j+\ell-1}) \tag{5}$$

where $f(S_{j:j+\ell})$ is the encoded representation of sentences $\{s_j, ...s_{j+\ell}\}$ and h_{j-1} would be initialized as a vector of zeros. The relative ordering teacher is trained to minimize $\mathcal{L}_{rel}$, the cosine similarity between gold orders of subsequences:

$$\mathcal{L}_{rel} = \frac{\langle f(\overrightarrow{S}_{j:j+\ell}), f(\overleftarrow{S}_{j:j+\ell})\rangle}{\|f(\overrightarrow{S}_{j:j+\ell})\|\|f(\overleftarrow{S}_{j:j+\ell})\|} \tag{6}$$

where the arrow above S signifies the order in which the sentences are processed. The relative ordering teacher learns to identify local sentence patterns among ordered sentences, thereby learning how to reward sequences that are temporally coherent.

3 Generator Architecture

In the task of recipe generation, the model is given a title of a recipe such as *"Cheese Sandwich"* and a list of ingredients (e.g., cheese, bread, etc.) and must generate the full multi-sentence recipe text. Similar to data to document generation tasks, the model must generate a full long-form text from sparse input signal, filling in missing information on its own (Wiseman et al., 2017).

3.1 Notation

Using the same notation as Kiddon et al. (2016), we are given a set of recipe title words $\{g_1, ..., g_n\}$ (e.g., { "cheese", "sandwich" }) and a list of ingredients $E = \{\mathbf{i}_1, ..., \mathbf{i}_{|E|}\}$ where each $\mathbf{i}$ can be a single- or multi-word ingredient phrase (e.g., "onions" or "onions, chopped"). In the following paragraphs, all W variables are projections matrices and all b variables are bias vectors.

3.2 Encoder

We use a modification of the baseline encoder of Kiddon et al. (2016). First, the title words are encoded as a bag of embeddings, $\mathbf{g}$. Second, each ingredient phrase $\mathbf{i}$ is encoded as a bag of embeddings vector, $\mathbf{e_i}$. The ingredient embeddings are inputs to a bidirectional gated recurrent unit, which yields an output vector $\mathbf{e}$. The final encoder output is the concatenation of these two representations, $\mathbf{h}^e = [\mathbf{g}, \mathbf{e}]$.

3.3 Decoder

The decoder is a separate gated recurrent unit that receives $\mathbf{h}^e$ from the encoder to initialize its hidden state $\mathbf{h}^d_0$ and must generate a full recipe word by word. At each time step, the model receives an input token embedding, x_t, as well as the output from the encoder $\mathbf{h}^e$:

$$a_t = \sigma(W_1 h^d_{t-1} + W_2 x_t + b_1) \tag{7}$$

$$\mathbf{z}_t = a_t \mathbf{h}^e \tag{8}$$

$$\tilde{x}_t = [x_t, \mathbf{z}_t] \tag{9}$$

where $\tilde{x}_t$ is the input to the recurrent unit at every time step. The recipe generator is pretrained to minimize the negative loglikelihood of predicting the next token in the recipe:

$$L_{mle} = -\sum_{t=1}^{T} \log P(x_t | x_0, ..., x_{t-1}, \mathbf{h}^e) \tag{10}$$

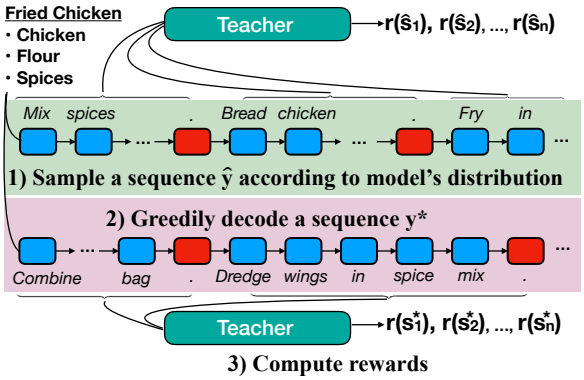


Figure 3: The model generates a recipe by sampling from its output vocabulary distribution and greedily decodes a baseline recipe. The generated sentences are passed to the teacher, which yields a reward for each sentence in each recipe.

where $\mathbf{h}^e$ is the encoded representation of the title and ingredients from Section 3.2 and T is the number of words in the gold recipe.

4 Policy Learning

Training a recipe generation model using maximum likelihood estimation produces generations that are locally coherent, but lack understanding of domain knowledge. By using a teacher that rewards the model for capturing cooking recipe discourse semantics, the model learns a policy that produces generations that better model the underlying recipe process. We learn a policy using the self-critical approach of Rennie et al. (2017).

4.1 Self-critical sequence training

In self-critical sequence training, outlined in Figure 3, the model learns by being rewarded for sampling sequences that receive more reward than a greedily decoded sequence. For each training example, a sequence $\hat{y}$ is generated by sampling from the model's distribution $P(\hat{y}_t|\hat{y}_0, ..., \hat{y}_{t-1}, \mathbf{h}^e)$ at each time step t. Once the sequence is generated, the teacher produces a reward $r(\hat{y}_t)$ for each token in the sequence. A second sequence y^* is generated by argmax decoding from $P(y^*_t|y^*_0, ..., y^*_{t-1}, \mathbf{h}^e)$ at each time step t. The model is trained to minimize:

$$\mathcal{L}_{rl} = -\sum_{t=1}^{T}(r(\hat{y}_t) - r(y^*_t)) \log P(\hat{y}_t|\hat{y}_0, ..., \hat{y}_{t-1}, \mathbf{h}^e) \tag{11}$$

where $r(y^*_t)$ is the reward produced by the teacher for tokens of the greedily decoded sequence. Because $r(y^*)$ can be viewed as a *baseline* reward that sampled sequences should receive more than, the model learns to generate sequences that receive more reward from the teacher than the best sequence that can be greedily decoded from the current policy. This approach allows the model to *explore* sequences that yield higher reward than the current best policy.

4.2 Rewards

As we decode a sequence $y = \{y_0..., y_t\}$, we track a sentence index that is the number of sentence delimiter tokens (e.g., ".") generated by the model. The model then implicitly decodes a set of generated sentences, $S' = \{s_0, ..., s_n\}$. These sentences are provided to the teachers defined in Section 2, which compute a score for the generated sequence. We explain the procedure for producing a token reward $r(y_t)$ from these scores below.

Absolute Order Once a sequence has been generated, the absolute order teacher computes a reward for y in the following way:

$$r_{abs}(y) = \frac{\langle f(S'), f(\overrightarrow{S})\rangle}{\|f(S')\|\|f(\overrightarrow{S})\|} - \frac{\langle f(S'), f(\overleftarrow{S})\rangle}{\|f(S')\|\|f(\overleftarrow{S})\|} \tag{12}$$

where $\overrightarrow{S}$ is the forward-ordered corresponding gold sequence and $\overleftarrow{S}$ is the reverse-ordered gold sequence. Both terms in the reward computation are variations of the loss function on which the absolute order teacher was trained (Equation (4)). This reward compares the generated sequence to both sentence orders of the gold sequence, and rewards generations that are more similar to the forward order of the gold sequence. Because the cosine similarity terms in Equation (12) are bounded in $[-1, 1]$, the model receives additional reward for generating sequences that are different from the reverse-ordered gold sequence.

Relative Order Similarly, the relative order reward is generated by the relative order teacher (§2.3), which evaluates subsequences of sentences, rather than the whole sequence. For a sentence s_j, the reward is computed as:

$$r_{rel}(s_j) = \frac{1}{L}\sum_{\ell=\ell_{min}}^{\ell_{max}}\left(\frac{\langle f(S'_{j-\ell:j}), f(\overrightarrow{S}_{j-\ell:j})\rangle}{\|f(S'_{j-\ell:j})\|\|f(\overrightarrow{S}_{j-\ell:j})\|} - \frac{\langle f(S'_{j-\ell:j}), f(\overleftarrow{S}_{j-\ell:j})\rangle}{\|f(S'_{j-\ell:j})\|\|f(\overleftarrow{S}_{j-\ell:j})\|}\right) \tag{13}$$

where ℓ_{min} and ℓ_{max} define the window of sentences to include in the computation of the reward. Similar to the absolute order teacher, the relative order teacher produces scores bounded in $[-1, 1]$, giving the model additional reward for generating sequences that are different from the reverse-ordered gold subsequences.

Credit Assignment When rewarding tokens with the absolute ordering teacher, each generated token receives the same sequence-level reward from the absolute order teacher:

$$r(y_t) = r_{abs}(y) \tag{14}$$

The relative order teacher, meanwhile, computes rewards for sentences based on their imitation of nearby sentences in the gold recipe. Rather than combining all rewards from the teacher to compute a full sequence reward, sentences should only be rewarded for their own quality. Each token in a sentence corresponds to a position in the full sequence. When relative order rewards are computed by the teacher, the correct sentence reward is indexed for each token. Consequently, when training with a relative order teacher, words only receive rewards for the sentences they belong to:

$$r(y_t) = \sum_{j=1}^{|S|} \mathbb{1}(y_t \in \hat{s}_j) r_{rel}(\hat{s}_j) \tag{15}$$

where $|S|$ is the number of sentences in the generated recipe, and $\mathbb{1}$ is an indicator variable identifying word y_t belonging to sentence s_j.

4.3 Mixed Training

As the model learns parameters to optimize the amount of reward it receives from the teacher, it is not explicity encouraged to produce fluent generations. The model quickly learns to generate simple sequences that *exploit* the teacher for high rewards despite being incoherent recipes (e.g., Figure 4). Consequently, it is possible that generated sequences are no longer readable (Pasunuru and Bansal, 2017; Paulus et al., 2018).

Title: Chili Grits
Ingredients: boiling water, butter, shredded cheddar cheese, jalapenos, eggs, chicken cream of soup, salt
Generated Recipe: Here .

Figure 4: Recipe generated from a self-critical model with no mixed training

To remedy this effect, the model optimizes a mixed objective that balances learning the discourse-focused policy while maintaining the generator's language model:

$$\mathcal{L}_{mix} = \gamma \mathcal{L}_{rl} + (1 - \gamma)\mathcal{L}_{mle} \tag{16}$$

where L_{mle} is the objective from Equation (10), L_{rl} is the objective from either Equation (11), and γ is a hyperparameter in [0, 1].

5 Experimental Setup

5.1 Datasets

We use the Now You're Cooking dataset with the same training/test/development splits from Kiddon et al. (2016). For training, we use 109567 recipes with 1000 recipes set aside for both development and test.

5.2 Training

Teacher Models The teachers are trained before the recipe generator and their parameters are fixed during generation. We tune hyperparameters on the development set. To train the relative order teacher, we sample 20 subsequences from each recipe of $\ell_{min} = 3$ to $\ell_{max} = 6$ sentences. Additional details are provided in Appendix A.2.

Recipe Generator We pretrain a recipe generator using a variant of the encoder-decoder baseline from Kiddon et al. (2016). Comprehensive hyperparameter details can be found in Appendix A.3.

Policy Learning We train a different model for three different teacher-provided rewards: absolute ordering (AO), relative ordering (RO) and a joint reward of relative ordering and BLEU-4 (RO + B4), where the full-sequence BLEU-4 reward and the sentence-level relative ordering reward are summed at each time step. The best model for the absolute and relative ordering rewards are the ones that receive the highest average reward on the development set. The best model for the mixed reward was chosen as the one that achieved the highest average geometric mean of BLEU-4 reward and average relative ordering reward for each generated sequence y in the development set:

$$\bar{r} = \frac{r_{b4}(y)}{T} \sum_{t=1}^{T} r_{RO}(y_t) \tag{17}$$

where r_{b4} is the BLEU-4 score of the whole generated sequence, and r_{RO} is computed using Equa-

Model	BLEU-1	BLEU-4	R-L	AB1	AB4	AR-L	SCB1	SCB4	SCR-L
Cross-entropy (MLE)	26.86	4.74	28.86	31.23	4.83	28.51	51.92	26.35	50.21
BLEU-4 (Rennie et al., 2017)	7.75	1.38	13.93	5.69	0.84	10.37	10.76	5.05	20.87
CIDEr (Rennie et al., 2017)	12.67	1.90	21.20	14.61	1.79	21.70	26.07	12.30	41.65
ROUGE-L (Paulus et al., 2018)	29.00	4.86	29.10	33.49	4.73	28.11	56.86	27.83	51.26
BLEU-1 ($\gamma = 0.97$)	**31.16**	**5.60**	29.53	32.28	5.09	29.34	52.63	25.43	51.58
BLEU-4 ($\gamma = 0.99$)	30.56	5.42	29.16	32.53	4.99	28.99	53.48	26.35	51.02
CIDEr ($\gamma = 0.97$)	29.60	5.10	28.79	33.93	4.81	28.41	57.00	27.55	50.57
ROUGE-L ($\gamma = 0.97$)	26.88	4.66	29.49	31.85	5.01	29.25	53.84	26.77	51.88
Absolute Ordering (AO)	23.70	4.25	28.43	28.22	4.44	27.88	47.93	24.47	50.15
Relative Ordering (RO)	27.75	4.88	29.60	34.37	**5.60**	**29.36**	58.31	29.14	**53.08**
Relative Ordering + BLEU-4	29.58	5.26	**29.78**	**35.13**	5.55	29.33	**59.13**	**29.19**	52.46

Table 1: Evaluation results for generated sequences by models and baselines. We **bold** the top performing result. The second to fourth columns list word-level scores. Columns AB1, AB4, and AR-L list action-level scores (§6.1). Columns SCB1, SCB4, and SCR-L list state change level scores (§6.1).

tion (15). Our best models use $\gamma = 0.97$ when training with the mixed objective from Equation (16).

5.3 Baselines

As baselines, we report results for a model trained only with cross-entropy loss (MLE) and for re-implemented versions of models from Rennie et al. (2017) and Paulus et al. (2018). These baselines achieved state of the art results in image captioning and document summarization tasks. We found, however, that their high γ (1 and 0.9984, respectively) led to low fluency, resulting in reduced performance on word-level scores. To control for this effect, we trained additional versions of each baseline with different values for γ and report the best performing configurations (see Table 1).

6 Results

6.1 Overlap Metrics

Scores We compute the example-level BLEU-1, BLEU-4, and ROUGE-L (R-L) scores for all recipes in the test set. A generated recipe, however, must be coherent at both the *word-level*, linking words and phrases sensibly, and the *world-level*, describing events that are grounded in real-world actions. Because n-gram scores do not evaluate if a generated recipe models this latent process, we also report these scores on the *action* and *state change* sequence described in the recipe. These words depict a *simulated* world where actions are taken and state changes are induced. A generated recipe should follow the sequence of actions taken in the gold recipe, and induce the same state changes as those in the gold recipe.

We use the state change lexicon from Bosselut et al. (2018) to map recipe words to ordered sequences of actions and state changes. Each entry in the lexicon contains an action in the cooking domain as well as the state changes that result from that action in the set of {LOCATION, COMPOSITION, COOKEDNESS, TEMPERATURE, SHAPE, CLEANLINESS}.

Action sequences are formed by mapping lemmas of words in generated sequences to entries in the lexicon. We compare these event sequences to the gold event sequences using the same scores as for words – BLEU-1, BLEU-4, and ROUGE-L. Intuitively, these scores can be seen as evaluating the following: whether the generated recipe depicts the same actions (AB1), subsequences of consecutive actions (AB4), and full action sequence (AR-L) as the gold recipe.

State change sequences are more coarse-grained than action sequences, and are formed by mapping actions to their state changes in the lexicon from Bosselut et al. (2018). These scores evaluate whether the generated recipe implies the same induced state changes (SCB1), subsequences of consecutive state changes (SCB4), and global state change order (SCR-L) as the gold recipe.

Results Our results in Table 1 show that models optimized on word overlap metrics achieve the greatest improvements for those scores. Optimizing scores such as BLEU-1 encourages the model to output words and phrases that overlap often with reference sequences, but that may not describe main events in the recipe process.

When examining models trained using a neural teacher, we see that the model optimized with

	MLE	RO + B4	Tie
Fluency	0.330	**0.447**	0.223
Ingredient Use	0.350	**0.440**	0.210
Title Completion	0.347	**0.430**	0.223
Action Order	0.377	**0.453**	0.170
	BLEU-1	RO + B4	Tie
Fluency	**0.387**	0.373	0.240
Ingredient Use	0.327	**0.363**	0.310
Title Completion	0.353	**0.377**	0.270
Action Order	**0.410**	0.403	0.187

Table 2: Human evaluation measuring proportion of winners. Upper table compares MLE baseline with RO + B4 model. Lower table compares BLEU-1 baseline with RO + B4 model.

	MLE	RO + B4	Tie
Fluency	0.317	**0.425**	0.258
Ingredient Use	0.342	**0.458**	0.200
Title Completion	0.358	**0.450**	0.192
Action Order	0.367	**0.483**	0.150
	BLEU-1	RO + B4	Tie
Fluency	**0.391**	0.383	0.225
Ingredient Use	0.267	**0.392**	0.342
Title Completion	0.325	**0.418**	0.258
Action Order	0.433	**0.442**	0.125

Table 3: Proportion of winners for **long** generated recipes. Upper table compares MLE baseline with RO + B4 model. Lower table compares BLEU-1 baseline with mixed RO + B4 model.

the absolute ordering reward performs worse than most baselines for every word-level score. The relative ordering model, however, raises every word-level score above the cross-entropy baseline, indicating the importance of fine-grained credit assignment at the sentence-level. The model trained with mixed rewards from the teacher and BLEU-4 achieves even higher scores, showing the benefits of training with diverse rewards.

When evaluating these metrics for the action and state change sequence, the models trained with feedback from the relative ordering teacher show large improvement over the baselines, indicating that the models exhibit more understanding of the latent process underlying the task. While optimizing word-level scores teaches the generator to output common sequences of words, the relative ordering reward teaches the model to focus on learning co-occurrences between recipe events.

6.2 Human Evaluation

We perform a human evaluation on 100 recipes sampled from the test set to evaluate our model on four aspects of recipe quality: fluency, ingredient use, title completion, and action ordering. For each example, three judges from Amazon Mechanical Turk are shown a pair of recipes, each generated by a different model and asked to select the recipe that is better according to the criteria above. For ingredient use, judges select the recipe that uses more of the ingredients correctly. For title completion, we ask judges to select the recipe that best completes the dish described in the recipe title. Finally, for action ordering, judges choose the recipe that better links subtasks in the recipes.

Models We use the Relative Ordering + BLEU-4 model (RO + B4) and compared to two baselines, the cross-entropy model (MLE), and the BLEU-1 model, which achieved the best scores on several word-level metrics (§6.1).

Results We report results in Table 2. Our model outperforms the cross-entropy baseline, consistently being preferred on aggregate for every question. Workers preferred the BLEU-1 baseline for the fluency and action order questions, while preferring recipes generated by the teacher-trained model for the ingredient use and title ordering questions. Upon further analysis, we see that the strength of the BLEU-1 model depends on the length of the original reference sequence. In Table 3, we show evaluation scores for recipes where the gold recipe was longer than 100 words. Our model's performance rises compared to the BLEU-1 model for every question, showing that modeling discourse structure as learned reward improves global coherence in long text.

6.3 Insights

Qualitative Analysis In Table 4, we see the effect that the neural teacher has on the recipe generator. The teacher rewards behavior that more closely imitates the actions in the gold recipe. In the first example, the generator learns to complete the actions of placing the mixture into the a *greased casserole* and then *baking* it, which the MLE model misses. The teacher also discourages repetitive phrases, as they provide no increase in reward during training. One weakness of our teacher models, however, is that they encourage common temporal patterns, such as in the third

Title **Ings**	Royale Casserole Broccoli, chopped; butter; flour; salt; black pepper; nutmeg; milk; cheese shredded swiss; lemon juice; cooked cubed turkey; mushrooms; grated parmesan cheese; biscuits, refrigerated
MLE	Cook broccoli according to package directions. Drain. Melt butter in skillet. Add flour, stirring constantly. Cook, stirring constantly, until thickened. Add milk, salt, pepper, and nutmeg. Cook over low heat, stirring constantly, until thickened. Add cheese, salt, pepper and pimento. Cook and stir till thickened. Add cheese, stirring until melted. Add **broccoli and mushrooms**. Cook over low heat for 5 minutes. Add cheese. Cook and stir until cheese melts. Add cheese. Cook and stir until cheese melts. Add cheese. Cook and stir until cheese melts. Add cheese and stir until melted. Serve over hot noodles.
RO	Melt butter in saucepan. Add flour and cook, stirring constantly, until thickened. Add milk, salt, pepper, and nutmeg. Cook, stirring constantly, until thickened. Add cheese, salt and pepper. Cook over low heat, stirring constantly, until mixture is thickened and smooth. Remove from heat. Stir in cheese. Spoon into **greased casserole**. Top with cheese. **Bake in 350 f oven for 30 minutes.** Serves 6.
Gold	Preheat oven to 375. Melt butter in saucepan. Blend in flour, salt, pepper, and nutmeg; cook 1-2 minutes. Gradually add milk; cook, stirring, until slightly thickened. Stir in frozen vegetables. Remove from heat; stir in cheese until melted. Add lemon juice, turkey, mushrooms and broccoli. Pour mixture into a lightly greased baking dish; sprinkle with parmesan and top with biscuits. Bake 20 minutes, or until biscuits are golden brown.
Title **Ings**	Wine Sauerkraut butter; dry white wine; 16 oz drained sauerkraut
MLE	In a saucepan, combine soup, wine, and butter. Heat, uncovered, in microwave oven 2 minutes or until mixture is heated through.
RO	Melt butter in skillet. **Add sauerkraut** and wine; heat to boiling. Reduce heat; cover and simmer 15 minutes. Add wine and heat to boiling; reduce heat. Cover and cook 10 minutes or until mixture is tender. Serve on rice.
Gold	Melt butter in 10-inch skillet over low heat; add sauerkraut. Cover and cook, stirring occasionally, 30 minutes; add wine. Heat to boiling; reduce heat. Cover and simmer until liquid is absorbed, about 45 minutes. Follow directions except simmer until liquid is absorbed, about 1 hour.
Title **Ings**	Strawberry Glazed Pie fresh strawberries; granulated sugar; salt; cornstarch; lemon juice; baked, cooled pie shell
MLE	Combine sugar, cornstarch and salt in top of a double boiler. Cook and stir over medium heat until mixture thickens and boils. Boil and stir 1 minute. Remove from heat and stir in lemon juice. Cool. Add strawberries and lemon juice. Pour into baked pie shell. Top with whipped cream. Serve immediately.
RO	Combine sugar, cornstarch, and salt in a saucepan; cook over medium heat, stirring constantly, until mixture boils and thickens. Remove from heat; stir in lemon juice. Cool. Add strawberries; mix well. Pour into baked pie shell. Bake in preheated 325-degree oven for 10 minutes. Cool. Garnish with whipped cream.
Gold	Wash, drain thoroughly, and hull strawberries. Arrange about 3 cups of whole berries over bottom of baked pastry shell. Crush remaining berries in a saucepan. In a bowl, mix sugar, salt and cornstarch; stir into crushed berries. Heat slowly, stirring constantly, until mixture comes to a boil and thickens. Remove from heat and stir in lemon juice. Cool, then spoon over berries in pie shell chill until glaze is set. Garnish with whipped cream.

Table 4: Example recipe generations from our model and comparative baselines. **Boxed** spans indicate recipe events missed by another model's generation. Red spans indicate superfluous events. The **Ings** row lists the ingredients (separated by semicolons) provided to make the dish in the title.

example in Table 4, where the generator mentions *baking the pie*. The model recognizes pies are generally supposed to be baked, even if it is not appropriate for that particular recipe.

Teacher Feedback Frequency We design the reward functions in Eq. 12 and Eq. 13 to require two passes through the teacher, one comparing the generated sequence to the forward gold sequence, and one comparing it to the reverse gold sequence. With no teacher comparison to the reverse-ordered sequence, the generator learns to *exploit* the teacher for reward with very simple sequences such as "Serve." and "Here's direction." When comparing with both orders, however, this effect is dampened, hinting at the importance of ensembling feedback from multiple sources for robust reward production. Another solution to this effect was mixing policy learning and maximum likelihood learning (Eq. 16) as the underlying language model of the generator did not deteriorate.

Impact of ℓ_{max} and γ Two hyperparameters to tune when training with teacher models are the mixed loss coefficient γ, which balances MLE learning with policy learning, and $[\ell_{min}, \ell_{max}]$, the number of sentences to consider when computing the relative order reward. We fix $\ell_{min} = 3$, and vary $\ell_{max} \in [3, 6]$ and $\gamma \in \{0.95, 0.97, 0.98\}$. Figure 5 shows the importance of tuning γ. A low γ will not allow the teacher to guide the model's learning, while a high γ causes the lan-

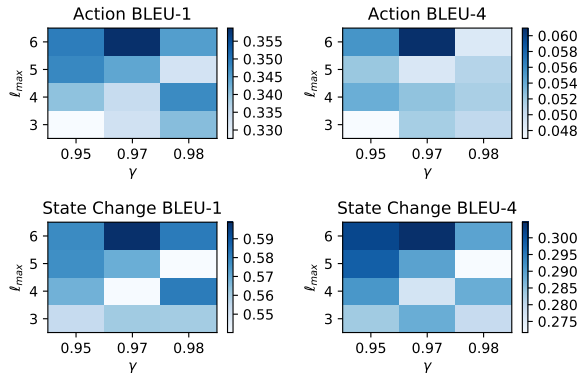


Figure 5: Action and State Change BLEU Metrics for different initializations of ℓ_{max} and γ

guage model to deteriorate. Interestingly, a higher ℓ_{max} leads to better performance on global coherence scores, implying that relative order rewards conditioned on more sentences allow the model to learn longer-range context co-occurrences.

7 Related Work

The field of neural text generation has received considerable attention in tasks such as image captioning (Vinyals et al., 2015; Xu et al., 2015), summarization (Rush et al., 2015; See et al., 2017), machine translation (Bahdanau et al., 2015), and recipe generation (Kiddon et al., 2016). While these works have focused on developing new neural architectures that introduce structural biases for easier *learning*, our work uses a simple architecture and focuses on improving the optimization of the learner (i.e., better *teaching*).

The importance of better teaching for RNN generators was outlined in Bengio et al. (2015), which showed that *exposure* bias from a misaligned train and test setup limited the capabilities of sequence-to-sequence models. This limitation had been addressed in previous work by augmenting training data with examples generated by pretrained models to make models robust to their own errors (Daumé III et al., 2009; Ross et al., 2011).

More recent work on training RNNs for generation has used sequence scores such as ROUGE (Paulus et al., 2018), CIDEr (Rennie et al., 2017; Pasunuru and Bansal, 2017), BLEU (Ranzato et al., 2015) and mixtures of them (Liu et al., 2017) as a global reward to train a policy with the REINFORCE algorithm (Williams, 1992). In contrast, our work uses a neural teacher to reward a model for capturing discourse semantics.

Most similar to our work is work on using neural and embedding rewards to improve dialogue (Li et al., 2016), image captioning (Ren et al., 2017), simplification (Zhang and Lapata, 2017), and paraphrase generation (Li et al., 2017). While these works use single-sentence similarity rewards for short generation tasks, our work designs teachers to reward long-range ordering patterns.

Finally, our teachers can be seen as rewarding generators that approximate script patterns in recipes. Previous work in learning script knowledge (Schank and Abelson, 1975) has focused on extracting scripts from long texts (Chambers and Jurafsky, 2009; Pichotta and Mooney, 2016), with some of that work focusing on recipes (Kiddon et al., 2015; Mori et al., 2014, 2012). Our teachers implicitly learn this script knowledge and reward recipe generators for exhibiting it.

8 Conclusion

We introduce the *absolute ordering* and *relative ordering* teachers, two neural networks that score a sequence's adherence to discourse structure in long text. The teachers are used to compute rewards for a self-critical reinforcement learning framework, allowing a recipe generator to be rewarded for capturing temporal semantics of the cooking domain. Empirical results demonstrate that our teacher-trained generator better models the latent event sequences of cooking recipes, and a human evaluation shows that this improvement is mainly due to maintaining semantic coherence in longer recipes.

Acknowledgments

This research was supported in part by NSF (IIS-1524371), DARPA under the CwC program through the ARO (W911NF-15-1-0543) and Samsung Research.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference for Learning Representations*.

Regina Barzilay and Mirella Lapata. 2005. Modeling local coherence: An entity-based approach. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics* 34(1).

Regina Barzilay and Lillian Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *HLT-NAACL*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*.

Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. 2018. Simulating action dynamics with neural process networks. *Proceedings of the 6th International Conference for Learning Representations* .

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction .

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9(8).

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*. ACM.

Chloé Kiddon, Ganesa Thandavam Ponnuraj, Luke Zettlemoyer, and Yejin Choi. 2015. Mise en place: Unsupervised interpretation of instructional recipes. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Jiwei Li and Eduard H Hovy. 2014. A model of coherence based on distributed sentence representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. 2017. Paraphrase generation with deep reinforcement learning. *arXiv preprint arXiv:1711.00279* .

Chin-Yew Lin. 2004. ROUGE: a package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*. Barcelona, Spain, volume 8.

Siqi Liu, Zhenhai Zhu, Ning Ye, Sergio Guadarrama, and Kevin Murphy. 2017. Improved image captioning via policy gradient optimization of spider. *Proceedings of the 2017 IEEE International Conference on Computer Vision* .

Ryan Lowe, Michael Noseworthy, Iulian Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic turing test: Learning to evaluate dialogue responses. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Shinsuke Mori, Hirokuni Maeta, Yoko Yamakata, and Tetsuro Sasada. 2014. Flow graph corpus from recipe texts. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*.

Shinsuke Mori, Tetsuro Sasada, Yoko Yamakata, and Koichiro Yoshino. 2012. A machine learning approach to recipe text processing. In *Proceedings of the 1st Cooking with Computer Workshop*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Ramakanth Pasunuru and Mohit Bansal. 2017. Reinforced video captioning with entailment rewards. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *Proceedings of the 6th International Conference for Learning Representations*.

Karl Pichotta and Raymond J. Mooney. 2016. Using sentence-level lstm language models for script inference. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. In *Proceedings of the 4th International Conference for Learning Representations*.

Zhou Ren, Xiaoyu Wang, Ning Zhang, Xutao Lv, and Li-Jia Li. 2017. Deep reinforcement learning-based image captioning with embedding reward. *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition* .

Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition* .

Stéphane Ross, Geoffrey J Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Roger C Schank and Robert P Abelson. 1975. *Scripts, plans, and knowledge*. Yale University.

Abigale See, Peter J. Liu, and Christopher Manning. 2017. Gettothepoint: Summarization with pointer-generatornetworks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. CIDEr: Consensus-based image description evaluation. In *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the 2015 IEEE Conference on Computer Cision and Pattern Recognition*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4).

Sam Wiseman, Stuart M. Shieber, and Alexander M. Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* .

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of The 32nd International Conference on Machine Learning*.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2017. Reference-aware language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

A Hyperparameters

A.1 Data

Each recipe is batched based on the number of tokens and number of ingredients it has. We use a minibatch size of 32.

A.2 Teachers

The hidden size of the reward generator is 100, the word embeddings have dimensionality 100. We use dropout with a rate of 0.3 between the bag of words layers and the recurrent layers.

A.3 Pretrained Recipe Generator

We use a hidden size of 256 for the encoder and 256 for the decoder. We initialize three different sets of embeddings for the recipe titles, ingredient lists, and text, each of size 256. All models are trained with a dropout rate of 0.3 and are single-layer. We use a temperature coefficient of $\beta = 2$ to make the output word distribution more peaky (Kiddon et al., 2016), allowing for more controlled exploration during self-critical learning. We use scheduled sampling with a linear decay schedule of 5% every 5 epochs up to a max of 50%. We use a learning rate of $\eta = 0.0003$ and train with the Adam optimizer.

A.4 Policy Learning

We use the same model hyperparameters as during pretraining, but re-initialize the Adam optimizer, use $\eta = 3 \times 10^{-5}$ as the learning rate, and do not train with scheduled sampling.

B Baseline Selection

For each baseline we trained, we report the score of the γ setting that achieved the highest score for the metric on which it was trained. For example, for baselines trained with ROUGE-L reward, we report the results for the model trained with the value of γ that scored the highest ROUGE-L score on the development set. For the models trained with the CIDEr reward, we select the model with value of γ that achieved the highest CIDEr score on the development set. We do the same for models trained with BLEU-1 and BLEU-4 rewards. The values of γ yielding the best performance on the development set were 0.97 for the BLEU-1, ROUGE-L, and CIDEr-trained models, and 0.99 for the BLEU-4 trained baseline. For each baseline, the best model is chosen by selecting the checkpoint that achieves the highest reward (or lowest loss for the MLE model) for the metric it was trained on.

Natural Answer Generation with Heterogeneous Memory

Yao Fu and **Yansong Feng**
Institute of Computer Science and Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
{francis_yao,fengyansong}@pku.edu.cn

Abstract

Memory augmented encoder-decoder framework has achieved promising progress for natural language generation tasks. Such frameworks enable a decoder to retrieve from a memory during generation. However, less research has been done to take care of the memory contents from different sources, which are often of heterogeneous formats. In this work, we propose a novel attention mechanism to encourage the decoder to actively interact with the memory by taking its heterogeneity into account. Our solution attends across the generated history and memory to explicitly avoid repetition, and introduce related knowledge to enrich our generated sentences. Experiments on the answer sentence generation task show that our method can effectively explore heterogeneous memory to produce readable and meaningful answer sentences while maintaining high coverage for given answer information.

1 Introduction

Most previous question answering systems focus on finding candidate words, phrases or sentence snippets from many resources, and ranking them for their users (Chu-Carroll et al., 2004; Xu et al., 2016). Typically, candidate answers are collected from different resources, such as knowledge base (KB) or textual documents, which are often with heterogeneous formats, e.g., KB triples or semi-structured results from Information Extraction (IE). For factoid questions, a single answer word or phrase is chosen as the response for users, as shown in Table 1 (A1).

However, in many real-world scenarios, users may prefer more natural responses rather than a single word. For example, as A2 in Table 1, *James Cameron directed the Titanic.* is more favorable than the single name *James Cameron.* A straightforward solution to compose an answer sentence is to build a template based model, where the answer word *James Cameron* and topic word in the question *the Titanic* are filled into a pre-defined template (Chu-Carroll et al., 2004). But such systems intrinsically lack variety, hence hard to generalize to new domains.

Q	Who is the director of the Titanic?
A1	**James Cameron**
A2	**James Cameron** directed **the Titanic**.
A3	**James Cameron** directed **it**.
A4	**James Cameron** directed it in **1999**.

Table 1: Answer sentences generated by different QA systems

To produce more natural answer sentences, Yin et al. (2015) proposed GenQA, an encoder-decoder based model to select candidate answers from a KB styled memory during decoding to generate an answer sentence. CoreQA (He et al., 2017b) further extended GenQA with a copy mechanism to learn to copy words from the question. The application of attention mechanism enables those attempts to successfully learn sentence varieties from the memory and training data, such as usage of pronouns (A3 in Table 1). However, since they are within the encoder-decoder framework, they also encounter the well noticed repetition issue: due to loss of temporary decoder state, an RNN based decoder may repeat what has already been said during generation (Tu et al., 2016a,b).

Both GenQA and CoreQA are designed to work with a structured KB as the memory, while in most real-world scenarios, we require knowledge from different resources, hence of different formats. This knowledge may come from structured KBs, documents, or even tables. It is admittedly challenging to leverage a heterogeneous memory in a neural generation framework, and it is not well studied in previous works (Miller et al., 2016). Here in our case, the memory should contain two main formats: KB triples and semi-structured en-

Proceedings of NAACL-HLT 2018, pages 185–195
New Orleans, Louisiana, June 1 - 6, 2018.

tities from IE, forming a heterogeneous memory (HM). The former is usually organized in is a subject-predicate-object form, while, the latter is usually extracted from textual documents, in the form of keywords, sometimes associated with certain categories or tags oriented to specific tasks (Bordes and Weston, 2016).

Miller et al. (2016) discuss different knowledge representations for a simple factoid QA task and show that classic structured KBs organized in a Key-Value Memory style work the best. However, dealing with heterogeneous memory is not trivial. Figure 1 shows an example of generating answer sentences from HM in a Key-Value style, which is indeed more challenging than only using a classic KB memory. Keys and values play different roles during decoding. A *director* key indicates this slot contains the answer. Same *James Cameron* values with different keys indicate duplication. The decoder needs this information to proactively perform memory addressing. Because keys from documents are not canonicalized, e.g., *doc directed* and *doc director*, they may lead to redundancy with the structured KB, e.g., *kb directed_by* and *doc director*. A decoder could repetitively output a director twice simply because there are two different memory slots hit by the query, both indicating the same director. This will make the the repetition issue even worse.

Although many neural generation systems can produce coherent answer sentences, they often focus on how to guarantee the chosen answer words to appear in the output, while ignoring many related or meaningful background information in the memory that can further improve user experiences. In real-world applications like chatbots or personal assistants, users may want to know not only the exact answer word, but also information related to the answers or the questions. This information is potentially helpful to attract users' attention, and make the output sentences more natural. For example in Table 1 (A4), the extra *1999* not only enriches the answer with the movie's release year, but also can act as a clue to help distinguish ambiguous candidate answers, e.g., *Titanic (1999)* and *Titanic (HD, 2016)*.

In this paper, we propose a sequence to sequence model tailing for heterogeneous memory. In order to bridge the gap between decoder states and memory heterogeneity, we split decoder states into separate vectors, which can be used to address

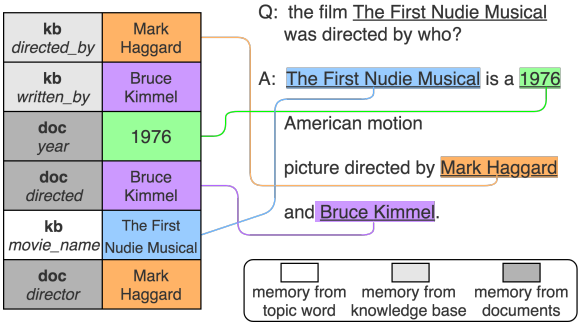


Figure 1: An example qa-pair with heterogeneous memory

different memory components explicitly. To avoid redundancy, we propose the **Cumulative Attention** mechanism, which uses the context of the decoder history to address the memory, thus reduces repetition at memory addressing time. We conduct experiments on two `WikiMovies` datasets, and experimental results show that our model is able to generate natural answer sentences composed of extra related facts about the question.

2 Related Work

Natural Answer Generation with Sequence to Sequence Learning: Sequence to sequence models (with attention) have achieved successful results in many NLP tasks (Cho et al., 2014; Bahdanau et al., 2014; Vinyals et al., 2015; See et al., 2017). Memory is an effective way to equip seq2seq systems with external information (Weston et al., 2014; Sukhbaatar et al., 2015; Miller et al., 2016; Kumar et al., 2015). GenQA (Yin et al., 2015) applies a seq2seq model to generate natural answer sentences from a knowledge base, and CoreQA (He et al., 2017b) extends it with copying mechanism (Gu et al., 2016). But they do not consider the heterogeneity of the memory, only tackle questions with one single answer word, and do not study information enrichment.

Memory and Attention: There are also increasing works focusing on different memory representations and the interaction between the decoder and memory, i.e., attention. Miller et al. (2016) propose the Key-Value style memory to explore textual knowledge (both structured and unstructured) from different sources, but they still utilize them separately, without a uniform addressing and attention mechanism. Daniluk et al. (2017) split the decoder states into key and value representation, and increase language modeling

performance. Multiple variants of attention mechanism have also been studied. Sukhbaatar et al. (2015) introduce multi-hop attention, and extend it to convolutional sequence to sequence learning (Gehring et al., 2017). Kumar et al. (2015) further extend it by using a Gated Recurrent Unit (Chung et al., 2014) between hops. These models show that multiple hops may increase the model's ability to reason. These multi-hop attention is performed within a single homogeneous memory. Our Cumulative Attention is inspired by them, but we utilize it cross different memory, hence can explicitly reason over different memory components.

Conditional Sentence Generation: Controllable sentence generation with external information is wildly studied from different views. From the task perspective, Fan et al. (2017) utilize label information for generation, and tackle information coverage in a summarization task. He et al. (2017a) use recursive Network to represent knowledge base, and Bordes and Weston (2016) track generation states and provide information enrichment, both are in a dialog setting. In terms of network architecture, Wen et al. (2015) equip LSTM with a semantic control cell to improve informativeness of generated sentence. Kiddon et al. (2016) propose the neural checklist model to explicitly track what has been mentioned and what left to say by splitting these two into different lists. Our model is related to these models with respect to information representation and challenges from coverage and redundancy. The most closely related one is the checklist model. But it does not explicitly study information redundancy. Also, the information we track is heterogeneous, and we track it in a different way, i.e. using Cumulative attention.

Due to loss of states across time steps, the decoder may generate duplicate outputs. Attempts have been made to address this problem. Some architectures try to utilize History attention records. See et al. (2017) introduce a coverage mechanism, and Paulus et al. (2017) use history attention weights to normalize new attention. Others are featured in network modules. Suzuki and Nagata (2017) estimate the frequency of target words and record the occurrence. Our model shows that simply attending to history decoder states can reduce redundancy. Then we use the context vector of attention to history decoder states to perform attention to the memory. Doing this enables the decoder to correctly decide what to say at memory addressing time, rather than decoding time, thus increasing answer coverage and information enrichment.

3 Task Definition

Given a question q and a memory M storing related information, our task is to retrieve all the answer words from the memory, generate an answer sentence x, and use the rest information as enrichment.

Answer Coverage is the primary objective of our task. Since many answers contain multiple words, the system needs to cover all the target words.

Information Redundancy is one challenge for this task. It is well noticed that the decoder language model may lose track of its state, thus repeating itself. Also, the decoder needs to reason over the semantic gap between heterogeneous memory slots, figuring out different keys may refer to the same value. These two kinds of redundancy should both be addressed.

Information Enrichment is another challenge. It requires the decoder to interact with the memory effectively and use the right word to enrich the answer.

The tradeoff between redundancy and coverage/enrichment is one of our main considerations. This is because when the decoder generates a word, it either generates a new word or a mentioned word. The more answer words and information enrichment are considered, the more likely the model repeats what it has already generated.

4 Our Model

Our model consists of the question encoder, the heterogeneous memory, and the decoder. The encoder embeds the question into a vector representation. The decoder reads questions, retrieves the memory, and generates answer sentences.

We use a Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) for question encoding and encode the question into an embedding. It takes every word embedding $(q_1, q_2...q_n)$ of question words as inputs, and generates hidden states $s_t = LSTM_{enc}(q_t, s_{t-1})$. These s are later used for decoder's attention. The last hidden state s_n is used as the vector representation of the question, and is later put into the initial hidden state of the decoder.

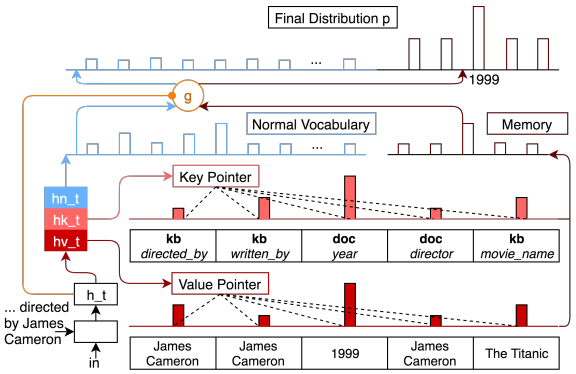


Figure 2: The Decoder with Heterogeneous States

We use a key-value memory M to represent the information heterogeneity. In our experiments, we study information from KB, topic words, and words extracted from documents. The memory is formatted as $((m_0^{\langle K\rangle}, m_0^{\langle V\rangle}), (m_1^{\langle K\rangle}, m_1^{\langle V\rangle})$ ... $(m_n^{\langle K\rangle}, m_n^{\langle V\rangle}))$, where $m_i^{\langle K\rangle}$ and $m_i^{\langle V\rangle}$ are respectively the key embedding and word embedding for the i-th memory slot. The vocabulary for keys V^{key} consists of all predicates in the KB, and all tags we use to classify the value words (e.g: *director, actor, or release_year*). The vocabulary for values V^{val} consists all related words from web documents, subjects and objects from the KB. This memory is later used in two ways: 1. the decoder uses its previous hidden state to perform attention and generate context vectors. 2. the decoder uses the updated hidden states as pointers (Vinyals et al., 2015) to retrieve the memory and copy the memory contents into the decoder's output.

4.1 Decoder with Heterogeneous States

As in the standard encoder-decoder architecture with attention, the word embedding of the decoder's previous time step x_t and context vector c_t is fed as the input of the next time step, and the hidden state h_t is updated then. The initial hidden state is the question embedding concatenated with average memory key and value:

$$h_t = LSTM_{dec}(x_t, c_t, h_{t-1})$$
$$h_0 = [s_n, avg(m^{\langle K\rangle}), avg(m^{\langle V\rangle})]$$

where $[\cdot,\cdot]$ denotes concatenation.

As shown in figure 2, to match the key-value memory representation, we use three linear transformations to convert the decoder's current h_t into $h_t^{\langle N\rangle}$, $h_t^{\langle K\rangle}$, and $h_t^{\langle V\rangle}$:

$$\begin{aligned} h_t^{\langle N\rangle} &= W_n h_t \\ h_t^{\langle K\rangle} &= W_k h_t \\ h_t^{\langle V\rangle} &= W_v h_t \end{aligned}$$

where the Ws are initialized as identity matrix $I = diag(1, 1...1)$. $h_t^{\langle N\rangle}$ will be projected to normal word vocabulary V^{norm} to form a distribution $p_t^{\langle N\rangle}$. $h_t^{\langle K\rangle}$ and $h_t^{\langle V\rangle}$ will be used as pointers to perform attention to memory keys $m^{\langle K\rangle}$ and values $m^{\langle V\rangle}$, respectively, and forms two distributions: $p_t^{\langle MK\rangle}$ and $p_t^{\langle MV\rangle}$. We use the average of the two as distribution over the memory: $p_t^{\langle M\rangle} = (p_t^{\langle MK\rangle} + p_t^{\langle MV\rangle})/2$. By doing this, we bridge the decoder's semantic space with the memory's semantic space, and explicitly maintains heterogeneity.

The decoder then uses a gating mechanism $g = sigmoid(W_g h_t + b_g)$ to decide whether the output x_t comes from the normal vocabulary or the memory. By mixing $p_t^{\langle N\rangle}$ and $p_t^{\langle M\rangle}$ with g, we get the distribution for the next decoder output:

$$\begin{aligned} &P(x_t|q, M, x_0, x_1, ...x_{t-1}) = \\ &\quad g \times P(X_t = w_k|q, M, x_0, x_1...x_{t-1}) + \\ &\quad (1-g) \times P(X_t = m_k|q, M, x_0, x_1...x_{t-1}) \end{aligned} \tag{1}$$

where

$$P(X_t = w_k|q, M, x_0, x_1...x_{t-1}) = p_t^{\langle N\rangle}$$
$$P(X_t = m_k|q, M, x_0, x_1...x_{t-1}) = p_t^{\langle M\rangle}$$

The three hs are then recorded as history states for later decoding time steps to perform the self-attention. We will explain this in the next section.

4.2 Cumulative Attention

As shown in Figure 3, our Cumulative Attention mechanism is exploited similarly to a multi-hop attention (Sukhbaatar et al., 2015). The difference is that the multi-hop attention uses context vector over one single memory at different hops, while our Cumulative Attention utilizes the context vector to query different memories. As shown in the left part of Figure 3, the decoder first performs self-attention to its history $h_t^{\langle N\rangle}$, $h_t^{\langle K\rangle}$, and $h_t^{\langle V\rangle}$, and generates corresponding context vectors c as:

$$\begin{aligned} c_t^{\langle HN\rangle} &= attn(h_{t-1}, hist(h_t^{\langle N\rangle})) \\ c_t^{\langle HK\rangle} &= attn(h_{t-1}, hist(h_t^{\langle K\rangle}) \\ c_t^{\langle HV\rangle} &= attn(h_{t-1}, hist(h_t^{\langle V\rangle})) \end{aligned}$$

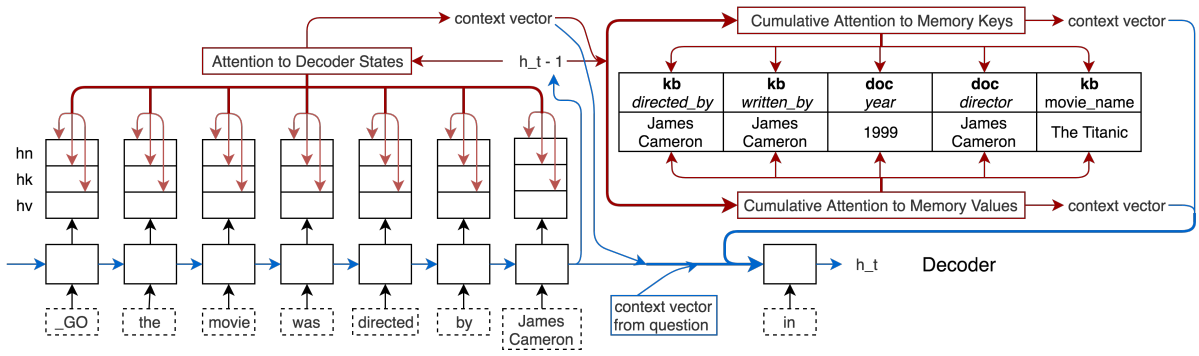


Figure 3: The Cumulative Attention Mechanism

where $c = attn(query, memory)$ denotes the attention function (Bahdanau et al., 2014), and the decoder's history states are defined as:

$$hist(h_t^{\langle N \rangle}) = (h_0^{\langle N \rangle}, h_1^{\langle N \rangle}, ...h_{t-1}^{\langle N \rangle})$$
$$hist(h_t^{\langle K \rangle}) = (h_0^{\langle K \rangle}, h_1^{\langle K \rangle}, ...h_{t-1}^{\langle K \rangle})$$
$$hist(h_t^{\langle V \rangle}) = (h_0^{\langle V \rangle}, h_1^{\langle V \rangle}, ...h_{t-1}^{\langle V \rangle})$$

The overall context vector is obtained through concatenation : $c_t^{\langle H \rangle} = [c_t^{\langle HN \rangle}, c_t^{\langle HK \rangle}, c_t^{\langle HV \rangle}]$, which is then used together with $h^{\langle K \rangle}$ and $h^{\langle V \rangle}$ to perform attention to $m^{\langle K \rangle}$ and $m^{\langle V \rangle}$, respectively:

$$c_t^{\langle MK \rangle} = attn([h_{t-1}^{\langle K \rangle}, c_t^{\langle H \rangle}], m^{\langle K \rangle})$$
$$c_t^{\langle MV \rangle} = attn([h_{t-1}^{\langle V \rangle}, c_t^{\langle H \rangle}], m^{\langle V \rangle})$$

where $m^{\langle K \rangle} = (m_0^{\langle K \rangle}, m_1^{\langle K \rangle}...m_n^{\langle K \rangle})$ and $m^{\langle V \rangle} = (m_0^{\langle V \rangle}, m_1^{\langle V \rangle}...m_n^{\langle V \rangle})$, as shown in the right part of Figure 3.

The decoder also performs attention to the question to get context vector $c_t^{\langle Q \rangle}$, as in the standard seq2seq attention model.

At time step t, all context vectors are concatenated: $c_t = [c_t^{\langle Q \rangle}, c_t^{\langle H \rangle}, c_t^{\langle MK \rangle}, c_t^{\langle MV \rangle}]$ to form the current input to the decoder. The decoder takes the context vector, the previous output, and the previous state to update its state, then generates a distribution for the next token, as shown in Section 4.1. We use the greedy decoding approach and choose the word with the highest probability as the current output.

For optimization, we jointly optimize the negative log-probability of the output sentence and the cross entropy $\mathcal{H}$ for gate g. Since g is the probability about whether the current output comes from the memory or the vocabulary, we can extract the label for g by matching sentence words with the memory. The overall loss function $\mathcal{L}$ can be written as:

$$\mathcal{L} = -\sum_{t=1}^{N} log(P(x_t|q, M, x_0...x_{t-1})) + \mathcal{H}(g, \hat{g})$$

We optimize $\mathcal{L}$ with gradient descent based optimizers.

5 Experiments

Our experiments are designed to answer the following questions: (1) whether our model can properly utilize heterogeneous memories to generate readable answer sentences, (2) whether our model can cover all target answers during generation, (3) whether our model can introduce related knowledge in the output while avoiding repetition.

5.1 Datasets

Our task requires a question, and a memory storing all the answer words and related knowledge as input, and produces a **natural, readable** sentence as the output. Unfortunately, there is no existing dataset that naturally fits to our task. We thus tailor the `WikiMovies`[1] dataset according to our requirements. This `WikiMovies` dataset was originally constructed for answering simple factoid questions, using memory networks with different knowledge representations, i.e., structured KB (**KB entries** in Table 2), raw textual documents (**Doc**), or processed documents obtained through information extraction (**IE**), respectively. The first is in the classic subject-predicate-object format. The second contains sentences from Wikipedia and also sentences automatically generated from predefined templates. The third is in the subject-verb-object format, collected by applying off-the-shell information extractor to all sentences.

[1] http://fb.ai/babi

The original data format		
Question	Who directed the film Blade Runner?	
KB entries	Blade Runner *directed_by* Ridley Scott Blade Runner *release_year* 1982 Blade Runner *written_by* Philip K. Dick	
IE	*year* 1982 *starred* Harrison Ford	
Doc	Blade Runner is a **1982** American film directed by **Ridley Scott** and starring **Harrison Ford**. It is directed by **Ridley Scott** and written by **Philip K. Dick**. It comes out in **1982**.	
Answer	Ridley Scott	
Our modified data format		
Question	Who directed the film Blade Runner?	
Memory	`Key`	`Value`
	directed_by	Ridley Scott
	release_year	1982
	written_by	Philip K. Dick
	movie	Blade Runner
	year	1982
	starred	Harrison Ford
Answer	Blade Runner is a **1982** American film directed by **Ridley Scott** and starring **Harrison Ford**.	

Table 2: The data format of `WikiMovies` used in our experiment.

As shown in Table 2, we treat each question in `WikiMovies` with its original answer (usually one or more words) as a QA pair, and one of the question's supportive sentences (either from Wikipedia or templates) as its gold-standard answer sentence. For each question, the memory will contain all knowledge triples about the question's topic movie from the **KB entries**, and also include entities and keywords extracted from its **IE** portion. For each entry in **KB entries**, we use the predicate as the key and the object as value to construct a new entry in our memory. For those from **IE**, we keep the extracted tags as the key and entities or other expressions as the value. Given a question, if an entity/expression in the memory is not the answer, it will be treated as information enrichment. According to whether the supportive sentences are generated by predefined templates or not, we split the dataset into `WikiMovies-Synthetic` and `WikiMovies-Wikipedia`.

The resulting `WikiMovies-Synthetic` includes 115 question patterns and 194 answer patterns, covering 10 topics, e.g., director, genre, actor, release year, etc. We follow its original data split, i.e., 47,226 QA-pairs for training, 8,895 for validation and 8,910 for testing.

In `WikiMovies-Wikipedia`, answer sentences are extracted from Wikipedia, admittedly noisy in nature. Note that there are more than 10K Wikipedia sentences that cannot be paired with any questions. We thus left their questions as blank and treat it as a pure generation task from a given memory, which can be viewed as a form of data augmentation to improve sentence variety. We split `WikiMovies-Wikipedia` the dataset randomly into 47,309 cases for training, 4,093 for testing and 3,954 for validation. We treat normal words occurring less than 10 times as `UNK`, and, eventually, have 24,850 normal words and 37,898 entity words. We cut the maximum length of answer sentences to 20, and the maximum memory size to 10, which covers most cases in both synthetic and Wikipedia datasets.

5.2 Metrics

We evaluate our answer sentences in terms of answer **coverage**, information **enrichment**, and **redundancy**. For cases with only one answer word, we design $\mathbf{C}_{single}$ to indicate the percentage of cases being correctly answered. Cases with more than one answer word are evaluated by $\mathbf{C}_{part}$, i.e., the percentage of answer words covered correctly, and $\mathbf{C}_{perfect}$ is the percentage of cases whose answers are perfectly covered. Here, the definition of **coverage** is similar in spirit with the conventional **recall** as both measure how many gold words are included in the output. Specifically, $\mathbf{C}_{part}$ is essentially the same as **recall** with respect to its own cases. Note that perfect coverage is the most difficult, while single coverage is the easiest one. For **Enrich**, we measure the number of none-answer memory items included in the output. Regarding **Redundancy**, we calculate the times of repetition for memory values in the answer sentence. We also compute BLEU scores (Papineni et al., 2002) on the `WikiMovies-Wikipedia`, as an indicator of naturalness, to some extent.

5.3 Comparison Models

We compare our full model (HS-CumuAttn) with state-of-the-art answer generation models and constrained sentence generation models. Our first baseline is GenQA (Yin et al., 2015), a standard encoder-decoder model with attention mechanism. We equip it with our Key-Value style heterogeneous memory. We also compare with its two variants. HS-GenQA: we split its decoder state into heterogeneous representations. The other one,

Model	Redundancy	C_{single}	C_{part}	$C_{perfect}$	Enrich
GenQA	0.1109	91.25%	69.19%	38.92%	0.1535
HS-GenQA	0.1218	94.10%	76.47%	50.10%	0.1951
GenQA-AttnHist	0.1280	95.99%	73.44%	44.94%	0.1903
CheckList	0.1176	93.80%	76.32%	50.04%	0.1963
HS-AttnHist	0.1295	97.17%	**77.90%**	**51.55%**	**0.1996**
HS-CumuAttn	**0.0983**	**98.15%**	**77.28%**	50.79%	0.1665

Table 3: Results on the `WikiMovies-Synthetic` dataset

Model	BLEU	Redundancy	C_{part}	$C_{perfect}$	Enrich
GenQA	42.50	0.2603	62.80%	18.24%	0.5903
CheckList	43.69	0.2744	63.42%	18.23%	0.6094
HS-CumuAttn	**44.97**	**0.2385**	**64.06%**	**19.09%**	**0.6218**

Table 4: Results on the `WikiMovies-Wikipedia` dataset

GenQA-AttnHist, is enhanced with a history attention during decoding.

CheckList (Kiddon et al., 2016) is the state-of-the-art model for generating long sentences with large agenda to mention. It keeps words that have been mentioned and words to mention using two separate records, and updates the records dynamically during decoding. To adapt to our task, we modify CheckList with a question encoder and a KV memory.

We also compare with one variant of our own model, HS-AttnHist, which does not benefit from the Cumulative Attention.

5.4 Implementation

Our model is implemented with the Tensorflow framework[2], version 1.2. We use the Adam optimizer (Kingma and Ba, 2014) with its default setting. The embedding dimension is set to be 256, as is the LSTM state size. We set the batch size to 128 and train the model up to 80 epochs.

As mentioned, there is a tradeoff between Coverage/Enrichment and Redundancy. To set up a more fair comparison for different models, we ask the control group to reach a comparable level of **Redundancy**, i.e., approximately 0.11-0.12 on `WikiMovies-Synthetic` and 0.26-0.27 on `WikiMovies-Wikipedia`. Keeping the **Redundancy** in around the same bucket, we compare their Coverage and Enrichment.

5.5 Results and Discussion

Let us first look at the performance on the `Synthetic` set in Table 3. GenQA is originally proposed to read only one single fact during decoding, so it is not surprising that it has the lowest answer coverage (38.92% $C_{perfect}$) and information enrichment (0.1535). After splitting the decoder state, HS-GenQA obtains significant improvement in both coverage (50.10% $C_{perfect}$) and enrichment (0.1952). When considering history for attention, GenQA-AttnHist achieves even better coverage (+3.% in C_{part} and +5% in $C_{perfect}$). By combining these two mechanisms, HS-AttnHist achieves the best perfect coverage, 51.55%. Although CheckList is not originally designed for our task, it still gives a strong performance (50.04% $C_{perfect}$ and 0.1963 enrichment), at a slightly lower redundancy (0.1176). Finally, our full model, HS-CumuAttn, achieves the best single coverage 98.15%, and comparable partial/perfect coverage, with the lowest redundancy (0.0983). Due to the lower level of redundancy, HS-CumuAttn does not include as much enrichment as other strong models, but still outperforms GenQA.

Question	the movie Torn Curtain starred who?	
Memory	0 *actor*	Julie Andrews
	1 *starred_actors*	Julie Andrews
	2 *starred_actors*	Paul Newman
	3 *movie*	Torn Curtain
	4 *year*	1966
	5 *director*	Alfred Hitchcock
	6 *actor*	Paul Newman
GenQA	It stared **Julie Andrews**$_0$ and **Julie Andrews**$_0$ and and.	
CheckList	**Torn Curtain**$_3$ is a **1966**$_4$ American film starring **Paul Newman**$_2$ and **Julie Andrews**$_0$ and **Julie Andrews**$_1$.	
HS-CumuAttn	**Torn Curtain**$_3$ is a **1966**$_4$ American political thriller film directed by **Alfred Hitchcock**$_5$, starring **Paul Newman**$_2$ and **Julie Andrews**$_0$.	

Table 5: Example sentences generated by different models, where an underlined bold phrase is the value of a memory slot selected from the memory by its corresponding generation model, and its subscript number is the index of this slot in the memory.

[2]www.tensorflow.org

Question 1	who starred in Cemetery Man ?			
Memory	0 *ans_actor* 2 *starred_actors* 4 *movie*	Rupert Everett Rupert Everett Cemetery Man	1 *ans_actor* 3 *starred_actors*	Anna Falchi Anna Falchi
Answer	The film stars Rupert Everett$_0$, _UNK , and Anna Falchi$_1$.			
Question 2	who was Dying Breed written by ?			
Memory	0 *ans_release_year* 2 *ans_actor* 4 *written_by*	2008 Nathan Phillips Jody Dwyer	1 *ans_writer* 3 *ans_writer* 5 *movie*	Jody Dwyer Leigh Whannell Dying Breed
Answer	Dying Breed$_5$ is a 2008$_0$ Australian horror film that was directed by Jody Dwyer$_1$ and stars Leigh Whannell$_3$ and Nathan Phillips$_2$.			
Question 3	who is the director that directed Livid ?			
Memory	0 *ans_director* 2 *ans_release_year* 4 *movie* 6 *ans_language*	Julien Maury 2011 Livid French	1 *directed_by* 3 *ans_director* 5 *directed_by*	Alexandre Bustillo Alexandre Bustillo Julien Maury
Answer	Livid$_4$ ( ) is a 2011$_2$ French$_6$ supernatural horror film directed and written by Julien Maury$_0$ and Alexandre Bustillo$_3$.			
Question 4	Drag Me to Hell , when was it released?			
Memory	0 *ans_director* 2 *release_year* 4 *ans_release_year*	Sam Raimi 2009 2009	1 *ans_wiki* 3 *ans_genre* 5 *movie*	Scream Horror Drag Me to Hell
Answer	Scream$_1$ is a 2009$_4$ film			
Question 5	the movie Lights in the Dusk starred who ?			
Memory	0 *starred_actors* 2 *starred_actors* 4 *starred_actors* 6 *ans_actor* 8 *ans_actor*	Janne Hyytiäinen Maria Järvenhelmi Ilkka Koivula Ilkka Koivula Maria Järvenhelmi	1 *ans_language* 3 *ans_actor* 5 *movie* 7 *ans_release_year*	Finnish Janne Hyytiäinen Lights in the Dusk 2006
Answer	Lights in the Dusk$_5$ (,) is a 2006$_7$ Finnish$_1$ drama film starring Janne Hyytiäinen$_3$, Ilkka Koivula$_6$ and Maria Järvenhelmi$_8$.			

Table 6: Example answers generated by our model. In an answer sentence, an underlined phrase is the value of a memory slot selected from the memory by our model, and the subscript number is the index of this slot in the memory.

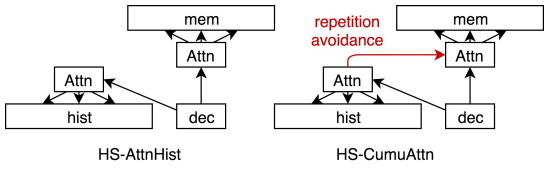


Figure 4: Two methods of using context of history to address the memory

We further break down the contributions from different mechanisms. Compared to vanilla GenQA, HS-GenQA splits the decoder states, thus improves the decoder's memory addressing process by performing attention separately, leading to improvements in both coverage and enrichment. Improvements of GenQA-AttnHist are of a different rationale. Looking at the history enables the decoder to avoid what are already said. Compared with HS-GenQA, GenQA-AttnHist improves Enrichment by avoiding repetition when introducing related information, while, HS-GenQA improves Enrichment by better memory addressing to select proper slots. Combining the two mechanisms together gives HS-AttnHist the best performance in Enrichment. However, HS-AttnHist still suffers from the repetition issue, to certain extent. Because when choosing memory content, there is no explicit mechanism to help the decoder to avoid repetitions according to the history (left of Figure 4). Therefore, a generated word may still be chosen again at the memory addressing step, leaving all the burden of avoiding repetition to the generation step. Our Cumulative Attention mechanism is designed to utilize the context vector of the history to address the memory, thus helps avoid choosing those already mentioned slots at memory addressing time (right of Figure 4), leading to almost the best coverage with the lowest redundancy.

Now we compare the three main models, GenQA, CheckList and our HS-CumuAttn on `WikiMovies-Wikipedia` (Table 4), which is admittedly more challenging than `WikiMovies-Synthetic`. We skip the C_{single} metrics here since most questions in `WikiMovies-Wikipedia` contain more than one answer word. It is not surprising that

CheckList, with a lower redundancy, still outperforms GenQA in almost all metrics, except $C_{perfect}$, since CheckList is originally designed to perform well with larger agenda/memory and longer sentences. On the other hand, our model, HS-CumuAttn, achieves the best performance in all metrics. Although the BLEU score is not designed to fully reflect the naturalness, it still indicates that our model can output sentences that share more n-gram snippets with reference sentences and are more similar to those composed by humans.

Case Study and Error Analysis Table 5 provides the system outputs from different models for an example question. We can see that GenQA may lose track of the decoder history, and repeat itself (*and and*), because there is no explicit mechanism to help avoid repetition. Also, it lacks informativeness and may not utilize other information stored in the memory. CheckList keeps records of what have been said and what are left to mention, thus reaches a good answer coverage. But its decoder is unable to explicitly address separate components within one memory slot, so it may not realize that the two *Julie Andrews*s are essentially the same person. HS-CumuAttn is able to find all the answer words correctly and also include the *director* into the sentence. After generating *Paul Newman*, the Cumulative Attention mechanism enables the model to realize that *Paul Newman* in slot 2 has been said, and *Paul Newman* in slot 6 is the same as slot 2, so it should not choose the 6th slot again. Rather it should move to *Julie Andrews*. Although the decoder may figure out the two *Paul Newman* are the same during decoding, the Cumulative Attention can explicitly help make the clarification during memory addressing. Intuitively, the attention across memory and history induces a stronger signal for the decoder to gather the right information.

Table 6 lists more typical imperfect output from our model. In question 1, there is considerable redundancy in the memory, but our decoder is still able to avoid repeatedly choosing the same entities from difference sources, though it produces a "_UNK" showing a slight incoherence. We think it comes from the gate g as it fails to decide that the current word should come from the memory. In question 2, the model correctly chooses the memory slot, but outputs the word "*directed*" while the correct word should be "*written*". This also shows an word choice inconsistency between the language model and the memory retrieval. Question 3 makes the same mistake, where it indeed chooses the right answer, but adds an incorrect word "*written*". We also observe a pair of additional parentheses, which are often used to accomodate *movie tags*, but we do not see any tags in this memory, so it has to be left blank. Question 4 shows an incorrect memory retrieval, where the decoder should have chosen slot 5 as the movie name. Question 5 is generally good enough, except the same parenthesis error as in question 4.

It is also interesting to see additional descriptions like "*Australian*", "*supernatural*" and "*drama*" in question 2, 3, and 5, introduced by the language model, rather than the memory. Although our model prevents repetition and obtains general naturalness, it cannot guarantee that the decoder can precisely use the right language to describe the memory information. We see the general readability of these sentences, yet they are still not as good as human composed ones. It is fairly subtle for the decoder to collaborate with the memory in different levels of semantics. The semantic coherency and word choice consistency is still a challenge in natural language generation.

6 Conclusion and Future Work

In this paper, we propose a novel mechanism within an encoder-decoder framework to enable the decoder to actively interact with a memory by taking its heterogeneity into account. Our solution can read multiple memory slots from different sources, attend across the generated history and the memory to explicitly avoid repetition, and enrich the answer sentences with related information from the memory. In the future, we plan to extend our work through 1) investigating more sophisticated structures in the memory such as knowledge graph, 2) solving more complex questions, such as those involving deep reasoning over multiple facts.

Acknowledgments

This work is supported by National High Technology R&D Program of China (Grant No.2015AA015403), Natural Science Foundation of China (Grant No. 61672057, 61672058). For any correspondence, please contact Yansong Feng.

References

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

Antoine Bordes and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *CoRR* abs/1605.07683. http://arxiv.org/abs/1605.07683.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR* abs/1406.1078. http://arxiv.org/abs/1406.1078.

Jennifer Chu-Carroll, Krzysztof Czuba, John M. Prager, Abraham Ittycheriah, and Sasha Blair-Goldensohn. 2004. Ibm's piquant ii in trec 2004. In *TREC*.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR* abs/1412.3555. http://arxiv.org/abs/1412.3555.

Michal Daniluk, Tim Rocktäschel, Johannes Welbl, and Sebastian Riedel. 2017. Frustratingly short attention spans in neural language modeling. *CoRR* abs/1702.04521. http://arxiv.org/abs/1702.04521.

Angela Fan, David Grangier, and Michael Auli. 2017. Controllable abstractive summarization. *CoRR* abs/1711.05217. http://arxiv.org/abs/1711.05217.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. *CoRR* abs/1705.03122. http://arxiv.org/abs/1705.03122.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *CoRR* abs/1603.06393. http://arxiv.org/abs/1603.06393.

He He, Anusha Balakrishnan, Mihail Eric, and Percy Liang. 2017a. Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. *CoRR* abs/1704.07130. http://arxiv.org/abs/1704.07130.

Shizhu He, Cao Liu, Kang Liu, and Jun Zhao. 2017b. Generating natural answers by incorporating copying and retrieving mechanisms in sequence-to-sequence learning. pages 199–208.

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

Chloé Kiddon, Luke S. Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *EMNLP*.

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

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *CoRR* abs/1506.07285. http://arxiv.org/abs/1506.07285.

Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *CoRR* abs/1606.03126. http://arxiv.org/abs/1606.03126.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '02, pages 311–318. https://doi.org/10.3115/1073083.1073135.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *CoRR* abs/1705.04304. http://arxiv.org/abs/1705.04304.

Abigail See, Peter Liu, and Christopher Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Association for Computational Linguistics*. https://arxiv.org/abs/1704.04368.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. MIT Press, Cambridge, MA, USA, NIPS'15, pages 2440–2448. http://dl.acm.org/citation.cfm?id=2969442.2969512.

Jun Suzuki and Masaaki Nagata. 2017. Cutting-off redundant repeating generations for neural abstractive summarization. In *EACL*.

Zhaopeng Tu, Yang Liu, Zhengdong Lu, Xiaohua Liu, and Hang Li. 2016a. Context gates for neural machine translation .

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016b. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 76–85. https://doi.org/10.18653/v1/P16-1008.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. MIT Press, Cambridge, MA, USA, NIPS'15, pages 2692–2700. http://dl.acm.org/citation.cfm?id=2969442.2969540.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1711–1721. https://doi.org/10.18653/v1/D15-1199.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *CoRR* abs/1410.3916. http://arxiv.org/abs/1410.3916.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Hybrid question answering over knowledge base and free text. In *COLING*.

Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. 2015. Neural generative question answering. *CoRR* abs/1512.01337. http://arxiv.org/abs/1512.01337.

Query and Output: Generating Words by Querying Distributed Word Representations for Paraphrase Generation

Shuming Ma[1], Xu Sun[1,2], Wei Li[1], Sujian Li[1], Wenjie Li[3], Xuancheng Ren[1]
[1]MOE Key Lab of Computational Linguistics, School of EECS, Peking University
[2]Deep Learning Lab, Beijing Institute of Big Data Research, Peking University
[3]Department of Computing, The Hong Kong Polytechnic University
{shumingma, xusun, liweitj47, lisujian, renxc}@pku.edu.cn
cswjli@comp.polyu.edu.hk

Abstract

Most recent approaches use the sequence-to-sequence model for paraphrase generation. The existing sequence-to-sequence model tends to memorize the words and the patterns in the training dataset instead of learning the meaning of the words. Therefore, the generated sentences are often grammatically correct but semantically improper. In this work, we introduce a novel model based on the encoder-decoder framework, called Word Embedding Attention Network (WEAN). Our proposed model generates the words by querying distributed word representations (i.e. neural word embeddings), hoping to capturing the meaning of the according words. Following previous work, we evaluate our model on two paraphrase-oriented tasks, namely text simplification and short text abstractive summarization. Experimental results show that our model outperforms the sequence-to-sequence baseline by the BLEU score of 6.3 and 5.5 on two English text simplification datasets, and the ROUGE-2 F1 score of 5.7 on a Chinese summarization dataset. Moreover, our model achieves state-of-the-art performances on these three benchmark datasets.[1]

1 Introduction

Paraphrase is a restatement of the meaning of a text using other words. Many natural language generation tasks are paraphrase-orientated, such as text simplification and short text summarization. Text simplification is to make the text easier to read and understand, especially for poor readers, while short text summarization is to generate a brief sentence to describe the short texts (e.g. posts on the social media). Most recent approaches use sequence-to-sequence model for paraphrase generation (Prakash et al., 2016; Cao et al., 2017). It compresses the source text information into dense vectors with the neural encoder, and the neural decoder generates the target text using the compressed vectors.

Although neural network models achieve success in paraphrase generation, there are still two major problems. One of the problem is that the existing sequence-to-sequence model tends to memorize the words and the patterns in the training dataset instead of the meaning of the words. The main reason is that the word generator (i.e. the output layer of the decoder) does not model the semantic information. The word generator, which consists of a linear transformation and a softmax operation, converts the Recurrent Neural Network (RNN) output from a small dimension (e.g. 500) to a much larger dimension (e.g. 50,000 words in the vocabulary), where each dimension represents the score of each word. The latent assumption of the word generator is that each word is independent and the score is irrelevant to each other. Therefore, the scores of a word and its synonyms may be of great difference, which means the word generator learns the word itself rather than the relationship between words.

The other problem is that the word generator has a huge number of parameters. Suppose we have a sequence-to-sequence model with a hidden size of 500 and a vocabulary size of 50,000. The word generator has up to 25 million parameters, which is even larger than other parts of the encoder-decoder model in total. The huge size of parameters will result in slow convergence, because there are a lot of parameters to be learned. Moreover, under the distributed framework, the more parameters a model has, the more bandwidth and memory it consumes.

To tackle both of the problems, we propose a novel model called Word Embedding Attention Network (WEAN). The word generator of WEAN

[1]The code is available at https://github.com/lancopku/WEAN

Proceedings of NAACL-HLT 2018, pages 196–206
New Orleans, Louisiana, June 1 - 6, 2018.

is attention based, instead of the simple linear softmax operation. In our attention based word generator, the RNN output is a query, the candidate words are the values, and the corresponding word representations are the keys. In order to predict the word, the attention mechanism is used to select the value matching the query most, by means of querying the keys. In this way, our model generates the words according to the distributed word representations (i.e. neural word embeddings) in a retrieval style rather than the traditional generative style. Our model is able to capture the semantic meaning of a word by referring to its embedding. Besides, the attention mechanism has a much smaller number of parameters compared with the linear transformation directly from the RNN output space to the vocabulary space. The reduction of the parameters can increase the convergence rate and speed up the training process. Moreover, the word embedding is updated from three sources: the input of the encoder, the input of the decoder, and the query of the output layer.

Following previous work (Cao et al., 2017), we evaluate our model on two paraphrase-oriented tasks, namely text simplification and short text abstractive summarization. Experimental results show that our model outperforms the sequence-to-sequence baseline by the BLEU score of 6.3 and 5.5 on two English text simplification datasets, and the ROUGE-2 F1 score of 5.7 on a Chinese summarization dataset. Moreover, our model achieves state-of-the-art performances on all of the benchmark datasets.

2 Proposed Model

We propose a novel model based on the encoder-decoder framework, which generates the words by querying distributed word representations with the attention mechanism. In this section, we first present the overview of the model architecture. Then, we explain the details of the word generation, especially the way to query word embeddings.

2.1 Overview

Word Embedding Attention Network is based on the encoder-decoder framework, which consists of two components: a source text encoder, and a target text decoder. Figure 1 is an illustration of our model. Given the source texts, the encoder compresses the source texts into dense representation vectors, and the decoder generates the paraphrased texts. To predict a word, the decoder uses the hidden output to query the word embeddings. The word embeddings assess all the candidate words, and return the word whose embedding matches the query most. The selected word is emitted as the predicted token, and its embedding is then used as the input of the LSTM at the next time step. After the back propagation, the word embedding is updated from three sources: the input of the encoder, the input of the decoder, and the query of the output layer. We show the details of our WEAN in the following subsection.

2.2 Encoder and Decoder

The goal of the source text encoder is to provide a series of dense representation of complex source texts for the decoder. In our model, the source text encoder is a Long Short-term Memory Network (LSTM), which produces the dense representation $\{h_1, h_2, ..., h_N\}$ from the source text $\{x_1, x_2, ..., x_N\}$:

The goal of the target text decoder is to generate a series of paraphrased words from the dense representation of source texts. Fisrt, the LSTM of the decoder compute the dense representation of generated words s_t. Then, the dense representations are fed into an attention layer (Bahdanau et al., 2014) to generate the context vector c_t, which captures context information of source texts. Attention vector c_t is calculated by the weighted sum of encoder hidden states:

$$c_t = \sum_{i=1}^{N} \alpha_{ti} h_i \tag{1}$$

$$\alpha_{ti} = \frac{e^{g(s_t, h_i)}}{\sum_{j=1}^{N} e^{g(s_t, h_j)}} \tag{2}$$

where $g(s_t, h_i)$ is an attentive score between the decoder hidden state s_t and the encoder hidden state h_i.

In this way, c_t and s_t respectively represent the context information of source texts and the target texts at the t^{th} time step.

2.3 Word Generation by Querying Word Embedding

For the current sequence-to-sequence model, the word generator computes the distribution of output words y_t in a generative style:

$$p(y_t) = softmax(W s_t) \tag{3}$$

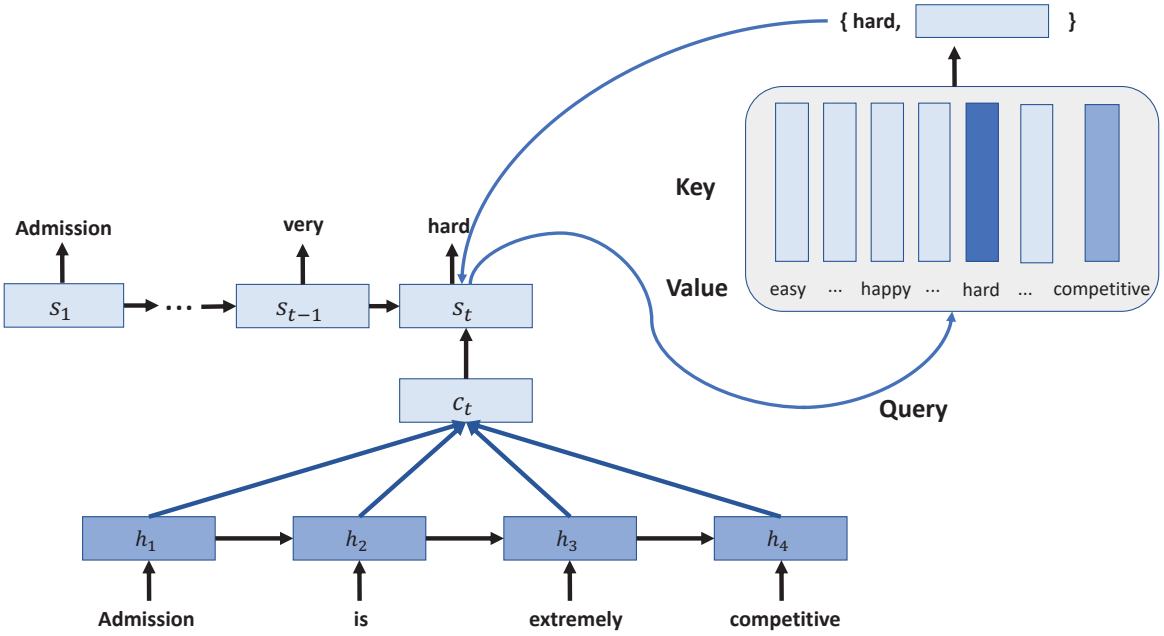


Figure 1: An overview of Word Embedding Attention Network.

where $W \in R^{k \times V}$ is a trainable parameter matrix, k is hidden size, and V is the number of words in the vocabulary. When the vocabulary is large, the number of parameters will be huge.

Our model generates the words in a retrieval style rather than the traditional generative style, by querying the word embeddings. We denote the combination of the source context vector c_t and the target context vector s_t as the query q_t:

$$q_t = \tanh(W_c[s_t; c_t]) \tag{4}$$

The candidate words w_i and their corresponding embeddings e_i are paired as the key-value pairs $\{w_i, e_i\}(i = 1, 2, ..., n)$, where n is the number of candidate words. We give the details of how to determine the set of candidate words in Section 2.4. Our model uses q_t to query the key-value pairs $\{w_i, e_i\}(i = 1, 2, ..., n)$ by evaluating the relevance between the query q_t and each word vector e_i with a score function $f(q_t, e_i)$. The query process can be regarded as the attentive selection of the word embeddings. We borrow the attention energy functions (Luong et al., 2015) as the relevance score function $f(q_t, e_i)$:

$$f(q_t, e_i) = \begin{cases} q_t^T e_i & \text{dot} \\ q_t^T W_a e_i & \text{general} \\ v^T \tanh(W_q q_t + W_e e_i) & \text{concat} \end{cases} \tag{5}$$

where W_q and W_e are two trainable parameter matrices, and v^T is a trainable parameter vector. In implementation, we select the general attention function as the relevance score function, based on the performance on the validation sets. The key-value pair with the highest score $\{w_t, e_t\}$ is selected. At the test stage, the decoder generates the key w_t as the t^{th} predicted word, and inputs the value e_t to the LSTM unit at the $t + 1^{th}$ time step. At the training stage, the scores are normalized as the word probability distribution:

$$p(y_t) = softmax(f(q_t, e_i)) \tag{6}$$

2.4 Selection of Candidate Key-value Pairs

As described in Section 2.3, the model generates the words in a retrieval style, which selects a word according to its embedding from a set of candidate key-value pairs. We now give the details of how to obtain the set of candidate key-value pairs. We extract the vocabulary from the source text in the training set, and select the n most frequent words as the candidate words. We reuse the embeddings of the decoder inputs as the values of the candidate words, which means that the decoder input and the predicted output share the same vocabulary and word embeddings. Besides, we do not use any pretrained word embeddings in our model, so that all of the parameters are learned from scratch.

2.5 Training

Although our generator is a retrieval style, WEAN is as differentiable as the sequence-to-sequence model. The objective of training is to minimize the

cross entropy between the predicted word probability distribution and the golden one-hot distribution:

$$L = -\sum_i \hat{y}_i \log p(y_i) \quad (7)$$

We use Adam optimization method to train the model, with the default hyper-parameters: the learning rate $\alpha = 0.001$, and $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-8$.

3 Experiments

Following the previous work (Cao et al., 2017), we test our model on the following two paraphrase orientated tasks: text simplification and short text abstractive summarization.

3.1 Text Simplification

3.1.1 Datasets

The datasets are both from the alignments between English Wikipedia website[2] and Simple English Wikipedia website.[3] The Simple English Wikipedia is built for "the children and adults who are learning the English language", and the articles are composed with "easy words and short sentences". Therefore, Simple English Wikipedia is a natural public simplified text corpus.

- **Parallel Wikipedia Simplification Corpus (PWKP).** PWKP (Zhu et al., 2010) is a widely used benchmark for evaluating text simplification systems. It consists of aligned complex text from English WikiPedia (as of Aug. 22nd, 2009) and simple text from Simple Wikipedia (as of Aug. 17th, 2009). The dataset contains 108,016 sentence pairs, with 25.01 words on average per complex sentence and 20.87 words per simple sentence. Following the previous work (Zhang and Lapata, 2017), we remove the duplicate sentence pairs, and split the corpus with 89,042 pairs for training, 205 pairs for validation and 100 pairs for test.

- **English Wikipedia and Simple English Wikipedia (EW-SEW).** EW-SEW is a publicly available dataset provided by Hwang et al. (2015). To build the corpus, they first align the complex-simple sentence pairs, score the semantic similarity between the complex sentence and the simple sentence, and classify each sentence pair as a good, good partial, partial, or bad match. Following the previous work (Nisioi et al., 2017), we discard the unclassified matches, and use the good matches and partial matches with a scaled threshold greater than 0.45. The corpus contains about 150K good matches and 130K good partial matches. We use this corpus as the training set, and the dataset provided by Xu et al. (Xu et al., 2016) as the validation set and the test set. The validation set consists of 2,000 sentence pairs, and the test set contains 359 sentence pairs. Besides, each complex sentence is paired with 8 reference simplified sentences provided by Amazon Mechanical Turk workers.

3.1.2 Evaluation Metrics

Following the previous work (Nisioi et al., 2017; Hu et al., 2015), we evaluate our model with different metrics on two tasks.

- **Automatic evaluation.** We use the BLEU score (Papineni et al., 2002) as the automatic evaluation metric. BLEU is a widely used metric for machine translation and text simplification, which measures the agreement between the model outputs and the gold references. The references can be either single or multiple. In our experiments, the references are single on PWKP, and multiple on EW-SEW.

- **Human evaluation.** Human evaluation is essential to evaluate the quality of the model outputs. Following Nisioi et al. (2017) and Zhang et al. (2017), we ask the human raters to rate the simplified text in three dimensions: Fluency, Adequacy and Simplicity. Fluency assesses whether the outputs are grammatically right and well formed. Adequacy represents the meaning preservation of the simplified text. Both the scores of fluency and adequacy range from 1 to 5 (1 is very bad and 5 is very good). Simplicity shows how simpler the model outputs are than the source text, which ranges from 1 to 5.

3.1.3 Settings

Our proposed model is based on the encoder-decoder framework. The encoder is implemented on LSTM, and the decoder is based on LSTM with Luong style attention (Luong et al., 2015). We

[2] http://en.wikipedia.org
[3] http://simple.wikipedia.org

PWKP	BLEU
PBMT (Wubben et al., 2012)	46.31
Hybrid (Narayan and Gardent, 2014)	53.94
EncDecA (Zhang and Lapata, 2017)	47.93
DRESS (Zhang and Lapata, 2017)	34.53
DRESS-LS (Zhang and Lapata, 2017)	36.32
Seq2seq (our implementation)	48.26
WEAN (our proposal)	**54.54**

Table 1: Automatic evaluation of our model and other related systems on PWKP datasets. The results are reported on the test sets.

EW-SEW	BLEU
PBMT-R (Wubben et al., 2012)	67.79
Hybrid (Narayan and Gardent, 2014)	48.97
SBMT-SARI (Xu et al., 2016)	73.62
NTS (Nisioi et al., 2017)	84.70
NTS-w2v (Nisioi et al., 2017)	87.50
EncDecA (Zhang and Lapata, 2017)	88.85
DRESS (Zhang and Lapata, 2017)	77.18
DRESS-LS (Zhang and Lapata, 2017)	80.12
Seq2seq (our implementation)	88.97
WEAN (our proposal)	**94.45**

Table 2: Automatic evaluation of our model and other related systems on EW-SEW datasets. The results are reported on the test sets.

tune our hyper-parameter on the development set. The model has two LSTM layers. The hidden size of LSTM is 256, and the embedding size is 256. We use Adam optimizer (Kingma and Ba, 2014) to learn the parameters, and the batch size is set to be 64. We set the dropout rate (Srivastava et al., 2014) to be 0.4. All of the gradients are clipped when the norm exceeds 5.

3.1.4 Baselines

We compare our model with several neural text simplification systems.

- **Seq2seq** is our implementation of the sequence-to-sequence model with attention mechanism, which is the most popular neural model for text generation.
- **NTS** and **NTS-w2v** (Nisioi et al., 2017) are two sequence-to-sequence model with extra mechanism like prediction ranking, and NTS-w2v uses a pretrain word2vec.
- **DRESS** and **DRESS-LS** (Zhang and Lapata, 2017) are two deep reinforcement learning sentence simplification models.
- **EncDecA** is a model based on the encoder-decoder with attention, implemented by Zhang and Lapata (2017).
- **PBMT-R** (Wubben et al., 2012) is a phrase based machine translation model which reranks the outputs.
- **Hybrid** (Narayan and Gardent, 2014) is a hybrid approach which combines deep semantics and mono-lingual machine translation.
- **SBMT-SARI** (Xu et al., 2016) is a syntax-based machine translation model which is trained on PPDB dataset (Ganitkevitch et al., 2013) and tuned with SARI.

PWKP	Fluency	Adequacy	Simplicity	All
NTS-w2v	3.54	3.47	3.38	3.46
DRESS-LS	3.68	3.55	3.50	3.58
WEAN	**3.77**	**3.66**	**3.58**	**3.67**
Reference	3.76	3.60	3.44	3.60

EW-SEW	Fluency	Adequacy	Simplicity	All
PBMT-R	3.36	2.92	3.37	3.22
SBMT-SARI	3.41	**3.63**	3.25	3.43
NTS-w2v	3.56	3.52	3.42	3.50
DRESS-LS	3.59	3.43	**3.65**	3.56
WEAN	**3.61**	3.56	**3.65**	**3.61**
Reference	3.71	3.64	3.45	3.60

Table 3: Human evaluation of our model and other related systems on PWKP and EW-SEW datasets. The results are reported on the test sets.

3.1.5 Results

We compare WEAN with state-of-the-art models for text simplification. Table 1 and Table 2 summarize the results of the automatic evaluation. On PWKP dataset, we compare WEAN with PBMT, Hybrid, EncDecA, DRESS and DRESS-LS. WEAN achieves a BLEU score of 54.54, outperforming all of the previous systems. On EW-SEW dataset, we compare WEAN with PBMT-R, Hybrid, SBMT-SARI, and the neural models described above. We do not find any public release code of PBMT-R and SBMT-SARI. Fortunately, Xu et al. (2016) provides the predictions of PBMT-R and SBMT-SARI on EW-SEW test set, so that we can compare our model with these systems.

LCSTS	R-1	R-2	R-L
RNN-W(Hu et al., 2015)	17.7	8.5	15.8
RNN(Hu et al., 2015)	21.5	8.9	18.6
RNN-cont-W(Hu et al., 2015)	26.8	16.1	24.1
RNN-cont(Hu et al., 2015)	29.9	17.4	27.2
SRB(Ma et al., 2017)	33.3	20.0	30.1
CopyNet-W(Gu et al., 2016)	35.0	22.3	32.0
CopyNet(Gu et al., 2016)	34.4	21.6	31.3
RNN-dist(Chen et al., 2016)	35.2	22.6	32.5
DRGD(Li et al., 2017)	37.0	24.2	34.2
Seq2seq	32.1	19.9	29.2
WEAN	**37.8**	**25.6**	**35.2**

Table 4: ROUGE F1 score on the LCSTS test set. R-1, R-2, and R-L denote ROUGE-1, ROUGE-2, and ROUGE-L, respectively. The models with a suffix of 'W' in the table are word-based, while the rest of models are character-based.

It shows that the neural models have better performance in BLEU, and WEAN achieves the best BLEU score with 94.45.

We perform the human evaluation of WEAN and other related systems, and the results are shown in Table 3. DRESS-LS is based on the reinforcement learning, and it encourages the fluency, simplicity and relevance of the outputs. Therefore, it achieves a high score in our human evaluation. WEAN gains a even better score than DRESS-LS. Besides, WEAN generates more adequate and simpler outputs than the reference on PWKP. The predictions of SBMT-SARI are the most adequate among the compared systems on EW-SEW. In general, WEAN outperforms all of the other systems, considering the balance of fluency, adequate and simplicity. We conduct significance tests based on t-test. The significance tests suggest that WEAN has a very significant improvement over baseline, with $p \leq 0.001$ over DRESS-LS in all of the dimension on PWKP, $p \leq 0.05$ over DRESS-LS in the dimension of fluency, $p \leq 0.005$ over NTS-w2v in the dimension of simplicity and $p \leq 0.005$ over DRESS-LS in the dimension of all.

3.2 Large Scale Text Summarization

3.2.1 Dataset

Large Scale Chinese Social Media Short Text Summarization Dataset (LCSTS): LCSTS is constructed by Hu et al. (2015). The dataset consists of more than 2,400,000 text-summary pairs, constructed from a famous Chinese social media website called Sina Weibo.[4] It is split into three parts, with 2,400,591 pairs in PART I, 10,666 pairs in PART II and 1,106 pairs in PART III. All the text-summary pairs in PART II and PART III are manually annotated with relevant scores ranged from 1 to 5. We only reserve pairs with scores no less than 3, leaving 8,685 pairs in PART II and 725 pairs in PART III. Following the previous work (Hu et al., 2015), we use PART I as training set, PART II as validation set, and PART III as test set.

3.2.2 Evaluation Metrics

Our evaluation metric is ROUGE score (Lin and Hovy, 2003), which is popular for summarization evaluation. The metrics compare an automatically produced summary against the reference summaries, by computing overlapping lexical units, including unigram, bigram, trigram, and longest common subsequence (LCS). Following previous work (Rush et al., 2015; Hu et al., 2015), we use ROUGE-1 (unigram), ROUGE-2 (bi-gram) and ROUGE-L (LCS) as the evaluation metrics in the reported experimental results.

3.2.3 Settings

The vocabularies are extracted from the training sets, and the source contents and the summaries share the same vocabularies. We tune the hyper-parameters based on the ROUGE scores on the validation sets. In order to alleviate the risk of word segmentation mistakes, we split the Chinese sentences into characters. We prune the vocabulary size to 4,000, which covers most of the common characters. We set the word embedding size and the hidden size to 512, the number of LSTM layers of the encoder is 2, and the number of LSTM layers of the decoder is 1. The batch size is 64, and we do not use dropout (Srivastava et al., 2014) on this dataset. Following the previous work (Li et al., 2017), we implement a beam search optimization, and set the beam size to 5.

3.2.4 Baselines

We compare our model with the state-of-the-art baselines.

- **RNN** and **RNN-cont** are two sequence-to-sequence baseline with GRU encoder and decoder, provided by Hu et al. (2015).

[4] http://weibo.com

#Param	PWKP	EWSEW	LCSTS
Seq2seq	12.80M	12.80M	2.05M
WEAN	0.13M	0.13M	0.52M

Table 5: The number of the parameters in the output layer. The numbers of rest parameters between Seq2seq and WEAN are the same.

- **RNN-dist** (Chen et al., 2016) is a distraction-based neural model, which the attention mechanism focuses on the different parts of the source content.

- **CopyNet** (Gu et al., 2016) incorporates a copy mechanism to allow part of the generated summary is copied from the source content.

- **SRB** (Ma et al., 2017) is a sequence-to-sequence based neural model with improving the semantic relevance between the input text and the output summary.

- **DRGD** (Li et al., 2017) is a deep recurrent generative decoder model, combining the decoder with a variational autoencoder.

- **Seq2seq** is our implementation of the sequence-to-sequence model with the attention mechanism.

3.2.5 Results

We report the ROUGE F1 score of our model and the baseline models on the test sets. Table 4 summarizes the comparison between our model and the baselines. Our model achieves the score of 37.8 ROUGE-1, 25.6 ROUGE-2, and 35.2 ROUGE-L, outperforming all of the previous models. First, we compare our model with the sequence-to-sequence model. It shows that our model significant outperforms the sequence-to-sequence baseline with a large margin of 5.7 ROUGE-1, 5.7 ROUGE-2, and 6.0 ROUGE-L. Then, we compare our model with other related models. The state-of-the-art model is DRGD (Li et al., 2017), which obtains the score of 37.0 ROUGE-1, 24.2 ROUGE-2, and 34.2 ROUGE-L. Our model has a relative gain of 0.8 ROUGE-1, 1.4 ROUGE-2 and 1.0 ROUGE-L over the state-of-the-art models.

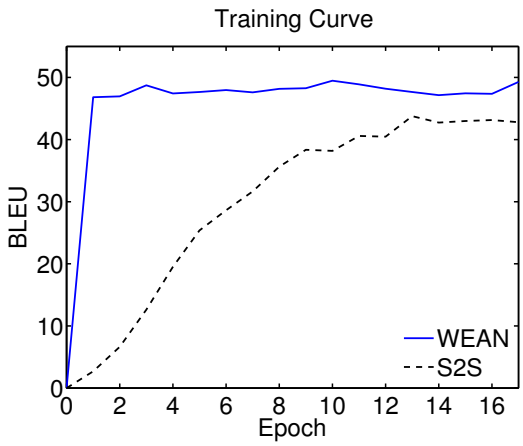


Figure 2: The training curve of WEAN and Seq2seq on the PWKP validation set.

4 Analysis and Discussion

4.1 Reducing Parameters

Our WEAN reduces a large number of the parameters in the output layer. To analyze the parameter reduction, we compare our WEAN model with the sequence-to-sequence model. Table 5 lists the number of the parameters in the output layers of two models. Both PWKP and EWSEWhave the vocabulary size of 50000 words and the hidden size of 256, resulting $50000 \times 256 = 12,800,000$ parameters. LCSTS has a vocabulary size of 4000 and the hidden size of 512, so the seq2seq has $4000 \times 512 = 2,048,000$ parameters in the output layers. WEAN only has two parameter matrices and one parameter vector at most in Equation 5, without regard to the vocabulary size. It has $256 \times 256 \times 2 + 256 = 131,328$ parameters on PWKP and EWSEW, and $512 \times 512 \times 2 + 512 = 524,800$ parameters on LCSTS. Besides, WEAN does not have any extra parameters in the other part of the model.

4.2 Speeding up Convergence

Figure 2 shows the training curve of WEAN and Seq2seq on the PWKP validation set. WEAN achieve near the optimal score in only 2-3 epochs, while Seq2seq takes more than 15 epochs to achieve the optimal score. Therefore, WEAN has much faster convergence rate, compared with Seq2seq. With the much faster training speed, WEAN does not suffer loss in BLEU, and even improve the BLEU score.

Source	*Yoghurt or yogurt is a dairy product produced by bacterial fermentation of milk .*
Reference	*Yoghurt or yogurt is a dairy product* ***made*** *by bacterial fermentation of milk .*
NTS	***. or yoghurt*** *is a dairy product* ***produced*** *by bacterial fermentation of milk .*
NTS-w2v	***It is made*** *by bacterial fermentation of milk .*
PBMT-R	*Yoghurt or yogurt is a dairy product* ***produced*** *by bacterial fermentation* ***of .***
SBMT-SARI	*Yogurt or yogurt is a dairy product* ***drawn up*** *by bacterial fermentation of milk .*
WEAN	*Yoghurt or yogurt is a dairy product* ***made*** *by bacterial fermentation of milk .*
Source	*Depending on the context, another closely-related meaning of constituent is that of a citizen residing in the area governed, represented, or otherwise served by a politician; sometimes this is restricted to citizens who elected the politician.*
Reference	***The word constituent can also be used to refer to*** *a citizen* ***who lives*** *in the area that is governed, represented, or otherwise served by a politician; sometimes* ***the word*** *is restricted to citizens who elected the politician.*
NTS	*Depending on the context, another closely-related meaning of constituent is that of a citizen* ***living*** *in the area governed, represented, or otherwise served by a politician; sometimes this is restricted to citizens who elected the politician.*
NTS-w2v	*This is restricted to citizens who elected the politician.*
PBMT-R	*Depending on the context and meaning of closely-related* ***siemens-martin -rrb- is a*** *citizen living in the area, or otherwise,* ***was governed by a 1924-1930 shurba****; this is restricted to people who elected* ***it****.*
SBMT-SARI	***In terms of*** *the context, another closely-related* ***sense of the component*** *is that of a citizen* ***living*** *in the area* ***covered, make up, or if not, served by a policy****; sometimes this is* ***limited*** *to the people who elected the* ***policy****.*
WEAN	*Depending on the context, another closely-related meaning of constituent is that of a citizen* ***who lives*** *in the area governed, represented, or otherwise served by a politician; sometimes* ***the word*** *is restricted to citizens who elected the politician.*

Table 6: Two examples of different text simplification system outputs in EW-SEW dataset. Differences from the source texts are shown in bold.

4.3 Case Study

Table 6 shows two examples of different text simplification system outputs on EW-SEW. For the first example, NTS, NTS-w2v and PBMT-R miss some essential constituents, so that the sentences are incomplete and not fluent. SBMT-SARI generates a fluent sentence, but the output does not preserve the original meaning. The predicted sentence of WEAN is fluent, simple, and the same as the reference. For the second example, NTS-w2v omits so many words that it lacks a lot of information. PBMT-R generates some irrelevant words, like 'siemens-martin', '-rrb-', and '-shurba', which hurts the fluency and adequacy of the generated sentence. SBMT-SARI is able to generate a fluent sentence, but the meaning is different from the source text, and even more difficult to understand. Compared with the statistic model, WEAN generates a more fluent sentence. Besides, WEAN can capture the semantic meaning of the word by querying the word embeddings, so the generated sentence is semantically correct, and very close to the original meaning.

5 Related Work

Our work is related to the encoder-decoder framework (Cho et al., 2014) and the attention mechanism (Bahdanau et al., 2014). Encoder-decoder framework, like sequence-to-sequence model, has achieved success in machine translation (Sutskever et al., 2014; Jean et al., 2015; Luong et al., 2015; Lin et al., 2018), text summarization (Rush et al., 2015; Chopra et al., 2016; Nallapati et al., 2016; Wang et al., 2017; Ma and Sun, 2017), and other natural language processing tasks (Liu et al., 2017). There are many other methods to improve neural attention model (Jean et al., 2015; Luong et al., 2015).

Zhu et al. (2010) constructs a wikipedia dataset, and proposes a tree-based simplification model. Woodsend and Lapata (2011) introduces a data-driven model based on quasi-synchronous grammar, which captures structural mismatches and complex rewrite operations. Wubben et al. (2012)

presents a method for text simplification using phrase based machine translation with re-ranking the outputs. Kauchak (2013) proposes a text simplification corpus, and evaluates language modeling for text simplification on the proposed corpus. Narayan and Gardent (2014) propose a hybrid approach to sentence simplification which combines deep semantics and monolingual machine translation. Hwang et al. (2015) introduces a parallel simplification corpus by evaluating the similarity between the source text and the simplified text based on WordNet. Glavaš and Štajner (2015) propose an unsupervised approach to lexical simplification that makes use of word vectors and require only regular corpora. Xu et al. (2016) design automatic metrics for text simplification. Recently, most works focus on the neural sequence-to-sequence model. Nisioi et al. (2017) present a sequence-to-sequence model, and re-ranks the predictions with BLEU and SARI. Zhang and Lapata (2017) propose a deep reinforcement learning model to improve the simplicity, fluency and adequacy of the simplified texts. Cao et al. (2017) introduce a novel sequence-to-sequence model to join copying and restricted generation for text simplification.

Rush et al. (2015) first used an attention-based encoder to compress texts and a neural network language decoder to generate summaries. Following this work, recurrent encoder was introduced to text summarization, and gained better performance (Lopyrev, 2015; Chopra et al., 2016). Towards Chinese texts, Hu et al. (2015) built a large corpus of Chinese short text summarization. To deal with unknown word problem, Nallapati et al. (2016) proposed a generator-pointer model so that the decoder is able to generate words in source texts. Gu et al. (2016) also solved this issue by incorporating copying mechanism.

6 Conclusion

We propose a novel model based on the encoder-decoder framework, which generates the words by querying distributed word representations. Experimental results show that our model outperforms the sequence-to-sequence baseline by the BLEU score of 6.3 and 5.5 on two English text simplification datasets, and the ROUGE-2 F1 score of 5.7 on a Chinese summarization dataset. Moreover, our model achieves state-of-the-art performances on these three benchmark datasets.

Acknowledgements

This work was supported in part by National Natural Science Foundation of China (No. 61673028), National High Technology Research and Development Program of China (863 Program, No. 2015AA015404), and the National Thousand Young Talents Program. Xu Sun is the corresponding author of this paper.

References

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

Ziqiang Cao, Chuwei Luo, Wenjie Li, and Sujian Li. 2017. Joint copying and restricted generation for paraphrase. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. pages 3152–3158.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for modeling documents. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2015)*. AAAI, New York, NY.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*. pages 1724–1734.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 93–98.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: the paraphrase database. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings*. pages 758–764.

Goran Glavaš and Sanja Štajner. 2015. Simplifying lexical simplification: Do we need simplified corpora? In *Proceedings of the 53rd Annual Meeting of*

the Association for Computational Linguistics, ACL. pages 63–68.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016*.

Baotian Hu, Qingcai Chen, and Fangze Zhu. 2015. LCSTS: A large scale chinese short text summarization dataset. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. pages 1967–1972.

William Hwang, Hannaneh Hajishirzi, Mari Ostendorf, and Wei Wu. 2015. Aligning sentences from standard wikipedia to simple wikipedia. In *NAACL HLT 2015*. pages 211–217.

Sébastien Jean, KyungHyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics, ACL 2015*. pages 1–10.

David Kauchak. 2013. Improving text simplification language modeling using unsimplified text data. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL*. pages 1537–1546.

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

Piji Li, Wai Lam, Lidong Bing, and Zihao Wang. 2017. Deep recurrent generative decoder for abstractive text summarization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. pages 2091–2100.

Chin-Yew Lin and Eduard H. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, HLT-NAACL 2003*.

Junyang Lin, Shuming Ma, Qi Su, and Xu Sun. 2018. Decoding-history-based adaptive control of attention for neural machine translation. *CoRR* abs/1802.01812.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2017. Table-to-text generation by structure-aware seq2seq learning. *CoRR* abs/1711.09724.

Konstantin Lopyrev. 2015. Generating news headlines with recurrent neural networks. *CoRR* abs/1512.01712.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*. pages 1412–1421.

Shuming Ma and Xu Sun. 2017. A semantic relevance based neural network for text summarization and text simplification. *CoRR* abs/1710.02318.

Shuming Ma, Xu Sun, Jingjing Xu, Houfeng Wang, Wenjie Li, and Qi Su. 2017. Improving semantic relevance for sequence-to-sequence learning of chinese social media text summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 2: Short Papers*. pages 635–640.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*. pages 280–290.

Shashi Narayan and Claire Gardent. 2014. Hybrid simplification using deep semantics and machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL*. pages 435–445.

Sergiu Nisioi, Sanja Stajner, Simone Paolo Ponzetto, and Liviu P. Dinu. 2017. Exploring neural text simplification models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL*. pages 85–91.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. pages 311–318.

Aaditya Prakash, Sadid A. Hasan, Kathy Lee, Vivek V. Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2016. Neural paraphrase generation with stacked residual LSTM networks. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*. pages 2923–2934.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. pages 379–389.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural

networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.

Xu Sun, Xuancheng Ren, Shuming Ma, and Houfeng Wang. 2017a. meprop: Sparsified back propagation for accelerated deep learning with reduced overfitting. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. pages 3299–3308.

Xu Sun, Xuancheng Ren, Shuming Ma, Bingzhen Wei, Wei Li, and Houfeng Wang. 2017b. Training simplification and model simplification for deep learning: A minimal effort back propagation method. *CoRR* abs/1711.06528.

Xu Sun, Bingzhen Wei, Xuancheng Ren, and Shuming Ma. 2017c. Label embedding network: Learning label representation for soft training of deep networks. *CoRR* abs/1710.10393.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*. pages 3104–3112.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on abstract meaning representation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. pages 1054–1059.

Kexiang Wang, Tianyu Liu, Zhifang Sui, and Baobao Chang. 2017. Affinity-preserving random walk for multi-document summarization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. pages 210–220.

Kristian Woodsend and Mirella Lapata. 2011. Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP*. pages 409–420.

Sander Wubben, Antal van den Bosch, and Emiel Krahmer. 2012. Sentence simplification by monolingual machine translation. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*. pages 1015–1024.

Jingjing Xu, Xu Sun, Xuancheng Ren, Junyang Lin, Binzhen Wei, and Wei Li. 2018. Dp-gan: Diversity-promoting generative adversarial network for generating informative and diversified text. *CoRR* abs/1802.01345.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *TACL* 4:401–415.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. pages 584–594.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *COLING 2010*. pages 1353–1361.

Simplification Using Paraphrases and Context-based Lexical Substitution

Reno Kriz[*], **Eleni Miltsakaki**[†],
Marianna Apidianaki[*△] and **Chris Callison-Burch**[*]
[*] Computer and Information Science Department, University of Pennsylvania
[△] LIMSI, CNRS, Université Paris-Saclay, 91403 Orsay
[†] Choosito, Inc.
{rekriz,marapi,ccb}@seas.upenn.edu, eleni@choosito.com

Abstract

Lexical simplification involves identifying complex words or phrases that need to be simplified, and recommending simpler meaning-preserving substitutes that can be more easily understood. We propose a complex word identification (CWI) model that exploits both lexical and contextual features, and a simplification mechanism which relies on a word-embedding lexical substitution model to replace the detected complex words with simpler paraphrases. We compare our CWI and lexical simplification models to several baselines, and evaluate the performance of our simplification system against human judgments. The results show that our models are able to detect complex words with higher accuracy than other commonly used methods, and propose good simplification substitutes in context. They also highlight the limited contribution of context features for CWI, which nonetheless improve simplification compared to context-unaware models.

1 Introduction

Automated text simplification is the process that involves transforming a complex text into one with the same meaning, but can be more easily read and understood by a broader audience (Saggion, 2017). This process includes several subtasks such as complex word and sentence identification, lexical simplification, syntactic simplification, and sentence splitting. In this paper, we focus on lexical simplification, the task of replacing difficult words in a text with words that are easier to understand.

Lexical simplification involves two main processes: identifying complex words within a text, and suggesting simpler paraphrases for these words that preserve their meaning in this context. To identify complex words, we train a model on data manually annotated for complexity. Unlike previous work, our classifier takes into account both lexical and context features. We extract candidate substitutes for the identified complex words from SimplePPDB (Pavlick and Callison-Burch, 2016), a database of 4.5 million English simplification rules linking English complex words to simpler paraphrases. We select the substitutes that best fit each context using a word embedding-based lexical substitution model (Melamud et al., 2015). An example sentence, along with the complex words identified by our model and the proposed replacements, is shown in Figure 1. We show that our complex word identification classifier and substitution model improve over several baselines which exploit other types of information and do not account for context. Our approach proposes highly accurate substitutes that are simpler than the target words and preserve the meaning of the corresponding sentences.

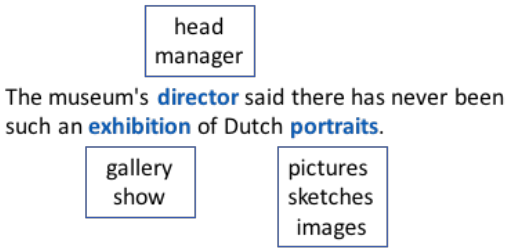


Figure 1: An example sentence with complex words identified by our classifier, and their substitutes proposed by the embedding-based substitution model.

2 Related Work

Prior approaches to text simplification have addressed the task as a monolingual translation problem (Zhu et al., 2010; Coster and Kauchak, 2011; Wubben et al., 2012). The proposed models are trained on aligned sentences extracted from Wikipedia and Simple Wikipedia, a corpus that

Proceedings of NAACL-HLT 2018, pages 207–217
New Orleans, Louisiana, June 1 - 6, 2018.

contains instances of transformation operations needed for simplification such as rewording, reordering, insertion and deletion. Zhu et al. (2010) propose to use a tree-based translation model which covers splitting, dropping, reordering and substitution. Coster and Kauchak (2011) employ a phrase-based Machine Translation system extended to support phrase deletion, and Wubben et al. (2012) augment a phrase-based system with a re-ranking heuristic.

Woodsend and Lapata (2011) view simplification as a monolingual text generation task. They propose a model based on a quasi-synchronous grammar, a formalism able to capture structural mismatches and complex rewrite operations. The grammar is also induced from a parallel Wikipedia corpus, and an integer linear programming model selects the most appropriate simplification from the space of possible rewrites generated by the grammar. The hybrid model of Angrosh et al. (2014) combines a synchronous grammar extracted from the same parallel corpus with a set of hand crafted syntactic simplification rules. In recent work, Zhang and Lapata (2017) propose a reinforcement learning-based text simplification model which jointly models simplicity, grammaticality, and semantic fidelity to the input. In contrast to these methods, Narayan and Gardent (2016)'s sentence simplification approach does not need a parallel corpus for training, but rather uses a deep semantic representation as input for simplification.

The above-mentioned systems support the full range of transformations involved in text simplification. Other works address specific subtasks, such as syntactic or lexical simplification, which involve identifying grammatical or lexical complexities in a text and rewriting these using simpler words and structures. Syntactic simplification might involve operations such as sentence splitting, rewriting of sentences including passive voice and anaphora resolution (Chandrasekar and Srinivas, 1997; Klerke and Søgaard, 2013).[1] Lexical simplification involves complex word identification, substitute generation, context-based substitute selection and simplicity ranking. To identify the words to be simplified, Shardlow (2013a) proposes to use a Support Vector Machine (SVM) that exploits several lexical features, such as frequency, character and syllable length. Our approach also uses a SVM classifier for identifying complex words, but complements this set of features with context-related features that have not been exploited in previous work.[2]

In the lexical simplification subtask, existing methods differ in their decision to include a word sense disambiguation (WSD) step for substitute selection and in the ranking method used. Ranking is often addressed in terms of word frequency in a large corpus since it has been shown that frequent words increase a text's readability (Devlin and Tait, 1998; Kauchak, 2013). Models that include a semantic processing step for substitute selection aim to ensure that the selected substitutes express the correct meaning of words in specific contexts. WSD is often carried out by selecting the correct synset (i.e. set of synonyms describing a sense) for a target word in WordNet (Miller, 1995) and retrieving the synonyms describing that sense. Thomas and Anderson (2012) use WordNet's tree structure (hypernymy relations) to reduce the size of the vocabulary in a document. Biran et al. (2011) perform disambiguation in an unsupervised manner. They learn simplification rules from comparable corpora and apply them to new sentences using vector-based context similarity measures to select words that are the most likely candidates for substitution in a given context. This process does not involve an explicit WSD step, and simplification is addressed as a context-aware lexical substitution task. The SemEval 2012 English Lexical Simplification task (Specia et al., 2012) also addresses simplification as lexical substitution (McCarthy and Navigli, 2007), allowing systems to use external sense inventories or to directly perform in-context substitution.

In our work, we opt for an approach which addresses lexical substitution in a direct way and does not include an explicit disambiguation step. Lexical substitution systems perform substitute ranking in context using vector-space models (Thater et al., 2011; Kremer et al., 2014; Melamud et al., 2015). Recently, Apidianaki (2016) showed that a syntax-based substitution model can successfully filter the paraphrases available in the

[1] For a detailed overview of syntactic simplification works, see (Shardlow, 2014).

[2] Datasets for system training and evaluation have been made available in the SemEval 2016 Complex Word Identification task (Paetzold and Specia, 2016) but present several issues that make system comparison problematic. We explain the drawbacks of the proposed datasets that led to their exclusion from this work in Section 5.

Paraphrase Database (PPDB) (Ganitkevitch et al., 2013) to select the ones that are adequate in specific contexts. In the same line, Cocos et al. (2017) used a word embedding-based substitution model (Melamud et al., 2015) for ranking PPDB paraphrases in context. We extend this work and adapt the Melamud et al. (2015) model to the simplification setting by using candidate paraphrases extracted from the Simple PPDB resource (Pavlick and Callison-Burch, 2016), a subset of the PPDB that contains complex words and phrases, and their simpler counterparts that can be used for in-context simplification.

3 Identifying Complex Words

3.1 Data

The first step for lexical simplification is to identify the complex words that should be simplified. The bulk of prior work on text simplification has addressed the complex word identification problem by training machine learning algorithms on the parallel Wikipedia Simplification (PWKP) corpus (Zhu et al., 2010). The PWKP corpus, however, has several shortcomings, as described in Xu et al. (2015). Namely, it was determined that 50% of the parallel sentences in PWKP were either not aligned correctly, or the simple sentence was not in fact simpler than the complex sentence. Xu et al. (2015) created a more reliably annotated dataset, which uses a corpus consisting of 1,130 articles, manually rewritten by experts at Newsela[3] at four different reading levels. Xu et al. (2015) also aligned sentences from these texts, extracting 141,582 complex/simple sentence pairs.

We use the Newsela corpus to create a gold-standard dataset of complex and simple words for training and testing our models. We do this by hiring crowdsourced annotators through Amazon Mechanical Turk, and asking them to identify complex words in the context of given texts. We randomly select 200 texts from the Newsela corpus, and take the first 200 tokens from each to be labeled by nine annotators. We preprocess the texts using the Stanford CoreNLP suite (Manning et al., 2014) for tokenization, lemmatization, part-of-speech (POS) tagging, and named entity recognition. The annotators are instructed to label at least 10 complex words they deem worth simplifying for young children, people with disabilities, and second language learners. After filtering out stop words (articles, conjunctions, prepositions, pronouns) and named entities, we are left with 17,318 labeled tokens. Tokens identified by at least three annotators are considered as complex, and tokens labeled by less than three or no annotators as simple. This increases the likelihood of complex segments being actually complex; as we can see from Table 1, words identified by only one or two annotators tend to be somewhat noisy.

Annotators	Prevalence	Example Words
0	0.617	heard, sat, feet, shops, town
1	0.118	protests, pump, trial
2	0.062	sentenced, fraction, primary
3	0.047	measures, involved, elite
4	0.035	fore, pact, collapsed
5	0.031	slew, enrolled, widespread
6	0.029	edible, seize, dwindled
7	0.023	perilous, activist, remorse
8	0.023	vintners, adherents, amassed
9	0.015	abdicate, detained, liaison

Table 1: Examples of words identified as difficult to understand within a text by n annotators, where $0 \leq n \leq 9$. Column 2 (Prevalence) shows the proportion of the total number of words identified as complex by n annotators.

3.2 Methods

Following Shardlow (2013a), we use a Support Vector Machine classifier. We also conduct experiments with a Random Forest Classifier. Shardlow (2013a) identified several features that help to determine whether or not a word is complex, including word length, number of syllables, word frequency, number of unique WordNet synsets, and number of WordNet synonyms. Shardlow used word frequencies extracted from SUBTLEX, a corpus of 51 million words extracted from English subtitles.[4] We instead use n-gram frequencies from the Google Web1T corpus Brants and Franz (2006) (henceforth Google n-gram).

Our motivation for using Google n-gram frequencies is based on the hypothesis that word frequency is a strong indicator of word difficulty. More frequent words are more likely to be easy, and less frequent words are more likely to be unknown and therefore hard to understand. The size of the Google n-gram corpus, consisting of a variety of texts across many genres and years, makes

[3]Newsela is a company that provides reading materials for students in elementary through high school. The Newsela corpus can be requested at https://newsela.com/data/

[4]SUBTLEX can be found at: https://www.ugent.be/pp/experimentele-psychologie/en/research/documents/subtlexus

it a good candidate for computing more accurate word frequencies.

In addition to word frequencies and word specific features, we include several context-specific features: average length of words in the sentence, average number of syllables, average word frequency, average number of WordNet synsets, average number of WordNet synonyms, and sentence length. The intuition for including context-specific features is that if a target word is surrounded by simple words, a reader is likely better able to understand the meaning of the target word, which would thus not need it simplified.

4 Lexical Simplification

4.1 Data

For our model and baselines, we consider candidate substitutions from three datasets. The first is WordNet (Miller, 1995), a lexical network encoding manually identified semantic relationships between words, such as synonymy, hypernymy and hyponymy. This resource has been widely used in substitution tasks (McCarthy and Navigli, 2007). We also use paraphrases extracted from the Paraphrase Database (PPDB) and the Simple Paraphrase Database (SimplePPDB). PPDB is a collection of more than 100 million English paraphrase pairs (Ganitkevitch et al., 2013). These pairs were extracted using a bilingual pivoting technique (Bannard and Callison-Burch, 2005), which assumes that two English phrases that translate to the same foreign phrase have the same meaning. PPDB was updated by Pavlick et al. (2015) to assign labels stating the precise entailment relationship between paraphrase pairs (e.g. forward/backward entailment), and new confidence scores (PPDB 2.0 scores) reflecting the strength of paraphrase relations.

SimplePPDB is a subset of PPDB which contains 4.5 million simplification rules, linking a complex word or phrase with a simpler paraphrase with the same meaning. Simplification rules come with both a PPDB 2.0 score and a simplification confidence score (Pavlick and Callison-Burch, 2016), which represent both the strength of the paraphrase relation and how well the replacement word simplifies the target word. These rules were created by sampling 1,000 PPDB phrases, using crowdsourcing to find correct simplifications for each phrase, and building a model to identify rules that simplify the input phrase.

To evaluate the performance of our lexical simplification model, we create a test set from the Newsela corpus. We extract lexical simplification rules from these parallel sentences using two methods. First, we find sentence pairs with only one lexical replacement and use these word pairs as simplification instances. Next, we use a monolingual word alignment software (Sultan et al., 2014) to extract all non-identical aligned word pairs. We only consider word pairs corresponding to different lemmas (i.e. words with different base forms). From this process, we collect a test set of 14,436 word pairs.

4.2 In-context Ranking and Substitution

To accurately replace words in texts with simpler paraphrases and ensure the generated sentences preserve the meaning of the original, we need to take into account the surrounding context. To do this, we adapt the word embedding-based lexical substitution model of Melamud et al. (2015) to the simplification task. Vector-space models have been shown to effectively filter PPDB paraphrases in context while preserving the meaning of the original sentences (Apidianaki, 2016; Cocos et al., 2017).

The Melamud et al. (2015) model (hereafter AddCos) quantifies the fit of substitute word s for target word t in context C by measuring the semantic similarity of the substitute to the target, and the similarity of the substitute to the context:

$$AddCos(s,t,C) = \frac{\cos(s,t)+\sum_{w\in C}\cos(s,w)}{|C|+1} \quad (1)$$

The vectors s and t are word embeddings of the substitute and target generated by the *skip-gram with negative sampling* model (Mikolov et al., 2013b,a). The context C is the set of context embeddings generated by *skip-gram* for words appearing within a fixed-width window of the target t in a sentence. We use a context window of 1; while this seems counter-intuitive, this is the best-performing window found by (Cocos et al., 2017), and we also confirm this result remains true in Section 5.2. We use the AddCos implementation of Cocos et al. (2017)[5], and 300-dimensional word and context embeddings trained over the 4 billion words in the AGiga corpus (Napoles

[5]Available at https://github.com/acocos/lexsub_addcos

et al., 2012) using the gensim word2vec package (Mikolov et al., 2013b,a; Řehůřek and Sojka, 2010). [6]

In our experiments, candidate substitutes for a target word are its paraphrases in the PPDB and SimplePPDB resources. The model needs to select among these candidates the ones that best carry the meaning of target words in specific contexts. We only consider content words (nouns, verbs, adjectives and adverbs) as simplification targets.

For a "target word-substitute" pair, we include in the model the following features which encode the strength of the semantic relationship between them:

- PPDB 1.0 and 2.0 scores, which represent the overall quality of paraphrases.
- Distributional similarity scores calculated by Ganitkevitch et al. (2013) on the Google n-grams and the AGiga corpus.
- Independence probability, that is the probability that there is no semantic entailment relationship between the paraphrase pair, as calculated by Pavlick et al. (2015).
- SimplePPDB score (Pavlick and Callison-Burch, 2016) – when considering SimplePPDB paraphrases – which reflects the confidence in the simplification rule.

5 Evaluation

5.1 Complex Word Identification

Datasets for training and evaluating Complex Word Identification (CWI) systems were created and released in the SemEval 2016 competition (Paetzold and Specia, 2016) but we decided not to use them for several reasons. Although this was a CWI task, surprisingly only 4.7% of the words in the test data were identified as complex, and all the other words were viewed as simple. As a consequence, none of the systems that participated in the SemEval task managed to beat the accuracy of the "All Simple" baseline which labeled all words in the test set as simple (0.953). As noted by Paetzold and Specia (2016), the inverse problem is present in the corpus developed by Shardlow (2013b), where the "All Complex" baseline achieved higher accuracy, recall and F-scores than all other tested systems, suggesting that marking all words in a sentence as complex is the most effective approach for CWI.

Model	Precision	Recall	F-Score
All-Complex	0.500	**1.000**	0.667
Token Length	0.757	0.900	0.822
n-gram Frequency	0.632	0.862	0.729
SVM-word	**0.880**	0.834	**0.857**
SVM-Context	0.871	0.831	0.850
RF-word	0.805	0.840	0.822
RF-Context	0.824	0.851	0.837

Table 2: Cross-validation performance for four different complex words identification classifiers. Comparison to three baselines. Scores are calculated using **unique** words in our training data.

Another problem in the SemEval-2016 dataset is that although the number of complex words is much higher in the training data (32%), 18% of all words were annotated as complex by only one out of 20 annotators and considered as complex. In addition to the highly different number of complex words in the training and test data, the two datasets are also imbalanced in terms of size, with only 2,237 training instances and 88,211 testing instances. These factors make this dataset a dubious choice for system evaluation. Comparison to the participating systems is also extremely difficult, since the best systems are ones that label most of the data as simple. For these reasons, we decided to create and use our crowdsourced data for training and evaluation.[7]

We compare the performance of an SVM classifier with only word features (SVM-word) to one that exploits both word and context features (SVM-context). We use 5-fold cross validation on unique words from the training data collected through Mechanical Turk (see Section 4.1). We also compare a Random Forest classifier with only word features (RF-word) to one with word and context features (RF-context). We consider three baselines:

- labeling all words as complex (All-Complex).
- thresholding for word length (Token Length), considering longer words as complex; the length threshold with the best performance was 7.

[6] The `word2vec` training parameters we use are a context window of size 3, learning rate *alpha* from 0.025 to 0.0001, minimum word count 100, sampling parameter $1e^{-4}$, 10 negative samples per target word, and 5 training epochs.

[7] We have released the new datasets at https://rekriz11.github.io

		Words with ≥1 paraphrase			All words		
Model	**Coverage**	**Top 1**	**Top 5**	**Oracle**	**Top 1**	**Top 5**	**Oracle**
WordNet frequency	0.911	0.141	0.267	0.291	0.129	0.244	0.265
SimplePPDB Score	0.935	0.180	0.403	0.669	0.168	0.377	0.626
AddCos-PPDB	**0.975**	0.196	0.444	**0.962**	0.191	0.433	**0.938**
AddCos-SimplePPDB	0.819	**0.353**	**0.601**	0.643	**0.289**	**0.492**	0.527

Table 3: Performance of the lexical simplification models on the Newsela aligned test set. Columns 3-5 show the performance of each model on only words with at least one paraphrase in the dataset. Columns 6-8 show the performance of each model on all words; this penalizes for the coverage of the databases.

- thresholding for word frequency using Google n-gram counts (n-gram Frequency), considering more frequent words as simple; the frequency threshold with the best performance was 19,950,000.

The results of this experiment are shown in Table 2. While the Token Length and n-gram Frequency baselines have higher recall, both of our models show substantial improvements in terms of precision and increases overall accuracy and F-score, with SVM outperforming Random Forest. The context-based features seem to have an ambiguous impact, in that they do not improve the performance of the SVM classifier, but they do improve that of the Random Forest classifier. While there are indeed some cases where a relatively simple word is more difficult to understand, due to the size of our corpus, these cases are not found that often in our dataset.

5.2 Lexical Simplification Evaluation

We evaluate the performance of the lexical substitution model using Simple PPDB paraphrases on a test set created from the Newsela corpus, described in Section 4.1. Using the complex word and the corresponding sentence, we find the top suggestions made by our word-embedding based substitution model using SimplePPDB. We compare to three baselines:

- WordNet Frequency: We extract all WordNet synonyms for a complex word, and collect the Google n-gram frequencies for each synonym. We then rank the synonyms in decreasing order of frequency (i.e. the most frequent synonym will be ranked first, and the least frequent one will be ranked last.
- SimplePPDB Score: We extract all SimplePPDB synonyms for a complex word. We then rank the synonyms in decreasing order of their SimplePPDB score.

Context Window	**Top 1**	**Top 5**
0	0.180	0.403
1	0.353	0.601
2	0.352	0.596
3	0.334	0.590
4	0.312	0.585
5	0.291	0.581
6	0.269	0.578
7	0.264	0.577
8	0.252	0.576
9	0.247	0.574
10	0.242	0.572

Table 4: Quality of substitutions proposed by AddCos-SimplePPDB with different context window size as measured by Top 1 and Top 5 accuracy on the Newsela aligned test set.

- AddCos-PPDB: We extract all PPDB synonyms for a complex word and rank them using the AddCos model described above.

The performance of AddCos with SimplePPDB paraphrases (AddCos-SimplePPDB) in the lexical simplification task is compared to performance of the baselines in Table 3. For each model, we calculate Top 1 and Top 5 accuracy scores, which show how often the gold-standard simple word was proposed as the best fitting or among the 5 highest-ranked paraphrases. In addition, we calculate the upper bound performance for each dataset (PPDB, SimplePPDB and WordNet), i.e. how often the gold-standard simple word was found as a paraphrase of the target word in the dataset. This is useful in telling us how well we could potentially do, if we could perfectly rank the paraphrases.

When performing this experiment, we also evaluated the impact of the context window size on the quality of the proposed substitutions. We varied the context window used by the AddCos-SimplePPDB model from 0 to 10. The results of this comparison are found in Table 4. As we can see, the largest effect, as expected, is when the model changes from using no context to choosing a window size of 1 word on either side of the word that is being replaced. As the context window in-

Synonym Rank	Substitution	Simplification	Both
1	**0.396**	**0.280**	**0.227**
2	0.311	0.214	0.153
3	0.278	0.184	0.127
4	0.228	0.142	0.093
5	0.193	0.123	0.075
All	**0.622**	**0.553**	**0.435**

Table 5: Performance of our overall lexical simplification system. We give the proportion of substitutes the system ranked at positions 1 to 5 (i.e. from the top ranked to the fifth-ranked paraphrase in context) which was identified by a majority of workers as (a) a good substitute in context (Substitution); (b) simpler than the target word (Simplification); (c) both a good and simpler substitute (Both). We also show the proportion of complex words where at least one of the top 5 paraphrases satisfies these criteria in the last row.

Baseline	Simple	Complex
n-gram Frequency	dug, sled, chart, lakes, push, tight, harm	estimates, frequent, attributed, isolated, preferred, liability
Token Length	nursing, unknown, squares, feeling, teaching, strength	adorns, asylum, myriad, rigors, nutria, edible
RF-Context	malls, hungry, therefore, hears, heavily, rainy	engaging, secular, gridlock, torrent, sanctions, lobbying
SVM-Context	peacefully, favorite, amazing, websites, harmful, somewhat	swelled, entice, tether, chaotic, vessel, midst

Table 6: Examples of words that were incorrectly classified by the two best performing baselines and the RF-Context model, but were correctly classified by the SVM-Context model. The last row shows examples of words that were incorrectly classified by the SVM-Context model.

creases above 2, however, we see a significant decrease in Top 1 accuracy, and a slower decrease in Top 5 accuracy. Thus, in our model, we chose to use a context window of 1.

We experimented with filtering the substitution candidates using SimplePPDB confidence scores, PPDB paraphrase quality scores, and AddCos context similarity scores, but these all resulted in a slight, non-significant increase in performance, and a significant decrease in coverage. We will also explore other ways for promoting high-quality substitutions without hurting the overall coverage of the system in the future.

One thing to note is that just because a model does not find the gold-standard simple word, does not necessarily mean that it does not find any good substitutes in context. Concrete examples of this are shown in Section 6.

5.3 Overall Simplification System

We integrate the best complex word identification classifier (SVM-context) and the substitution model that provided the best ranking in context (AddCos-SimplePPDB), into a simplification pipeline. The input text is a complex text that needs to be simplified and the output consists of simplification suggestions for experts to choose from in order to create simpler versions of texts. The input text is pre-processed using the Stanford CoreNLP suite (Manning et al., 2014) which performs tokenization, sentence splitting, lemmatization, part-of-speech and named entity tagging. The SVM-Context classifier is used to classify each content word that is not part of a named entity as either simple or complex.

The lexical substitution model then gathers the SimplePPDB substitutes available for the complex target word and ranks them according to how well they fit the corresponding context. We only keep the top five suggestions made by the model as final output.

To evaluate the performance of the overall simplification system, we used the 930 texts from the Newsela corpus that were not used in the training of the CWI classifier. Our model identified over 170,000 complex words that also had paraphrases in SimplePPDB. We again asked crowdsourced annotators to evaluate the suggestions made for a random sample of 2,500 complex words on Amazon Mechanical Turk, in order to determine the number of good substitutions in context, the number of suggested paraphrases that are simpler than the target words, and the suggestions that are both simpler paraphrases and good in-context substitutes.

Table 5 shows the quality of the paraphrases ranked by our system in positions from one to five. We can see that the paraphrases our system selects as the best have a higher likelihood of being both good substitutes in context and simpler than the target word. We also show the proportion of target words that had at least one good substitute in context, one simple substitute, and one good and simple substitute.

6 Error Analysis

In this section, we give examples of words for which our models give the correct output and the

Sentence	Gold-Standard	WordNet Frequency	SimplePPDB Score	AddCos-PPDB	AddCos-SimplePPDB
(7.1) Advocates **argue** that including women will help end harassment of female troops.	**say**	reason, fence, debate, contend, indicate	**say**, think, tell, talk, mean	contend, assert, acknowledge, insist, complain	**say**, claim, believe, suggest, debate
(7.2) But in April , detainees covered cameras used to **monitor** them.	**watch**	supervise, proctor, admonisher	find, meet, give, try, allow	track, manipulate, control, analyze, supervise	track, control, check, **watch**, follow
(7.3) Similarly , police can investigate cases and have the **authority** to seize animals.	**power**	agency, potency, bureau, assurance	force, control, permission, office, limit	jurisdiction, discretion, right, prerogative, ability	**power**, responsibility, body, agency

Table 7: Examples of the top-5 substitutes for our three baselines and our best model (AddCos-SimplePPDB). We also provide the gold-standard simplification (Gold-Standard).

Sentence	Bad Substitutions
(8.1)b	basic
(8.2) Russian poultry is more expensive, and U.S. producers enjoy numerous cost **advantages**.	prospect, benefits, revenue, merit, feature
(8.3) Although the calculus may be different with Syrian **refugees**, the parallel for me is politics.	life, right, return, shelter, million
(8.4) He saw them bring in animals to a university, where they'll be cared for and put up for **adoption**.	acceptance, passage, approval, endorsement

Table 8: Examples of words and their context where our model fails to provide any good replacements.

baselines fail to do so. In addition, we give examples of words on which our models perform poorly.

First, we consider examples of words that were incorrectly classified by each of the four best performing CWI models: the RF-Context and SVM-Context models, and the n-gram Frequency and Token Length baselines. (Table 6). In the first three rows, we give words that were correctly identified by SVM-Context, but incorrectly categorized by the two baselines and RF-Context; in the last row, we give examples of words incorrectly classified by SVM-Context. We observe that the n-gram Frequency model tends to incorrectly classify relatively short words that are rare in the Google n-gram corpus as complex. On the other end, the Token Length model shows that using this feature alone leads to incorrectly identifying shorter words such as "adorn" and "myriad" as simple, when these words are relatively complex.

Table 7 presents examples of substitution where the baseline systems did not find the correct paraphrase, but AddCos-SimplePPDB did. As we have mentioned, even when a model did not find the gold-standard paraphrase, they sometimes did find a different paraphrase that works well in the context. In Example 7.2, the top paraphrase identified by both AddCos-PPDB and AddCos-Simple PPDB for the word "monitor" is "track", which is a reasonable substitute. On the other hand, in Example 7.3, AddCos-Simple PPDB model was able to identify a good simple substitute, when none of the other models were able to identify a suitable word with comparable complexity.

Finally, Table 8 shows examples of output of the overall simplification system. Here, the **blue** word is a word that our CWI classifier identified as complex (for simplicity, we only look at one complex word per sentence). From there, we consider the five top-ranked substitutes proposed by AddCos-Simple PPDB, and show which were identified by the majority of annotators as good substitutes for the target word, simpler than the target, good simpler substitutes, and bad substitutes. In row 5 of Table 8, we can see that for the word "adoption", all five words identified by our model are considered to be bad substitutes, since they are all synonyms describing a different sense of adoption. Even though SimplePPDB is quite large, it does not cover all senses of the words represented. Another issue is that SimplePPDB contains some noisy paraphrases, as is the case with all automatically collected synonym banks. We see this with "recognize" being a synonym of "recognition", even though we specified that "recognition" is a noun. Our model does filter out the worst paraphrases (with PPDB2.0 score < 2), but there are still some words that are simply poor substitutes.

We reviewed the examples where our system failed to generate acceptable substitutions for the identified complex words. Below we present the major categories of errors.

- The identified complex term is part of a phrase and no substitution is acceptable. For example, in Example 8.1, Elementary, Mid-

dle or High School is a description of the type of school. *Elementary School* has an alternative name in some cases but *High School* should never become *Tall School*.

- The complex word has no simpler synonym that would be a good substitute. The difficulty of the word might reside in its meaning which can be unknown to the reader. In Example 8.3, it would be more useful to point to the definition of *refugees*.
- The complex word is part of a predicate with arguments that are not accessible to our model. In Example 8.4, the intended meaning of *adoption*, human adoption, is hard to capture in the vicinity of the complex word.
- Finally, in some cases, our annotators were quite strict in admitting a substitute. In Example 8.2, for example, *cost merit* would not be syntactically correct but *cost merits* would be acceptable.

7 Conclusions and Future Work

We present a novel model for simplification that first identifies complex words in text, and then ranks lexical simplification candidates according to their adequacy in these specific contexts. We perform experiments showing that our model makes correct simplification suggestions 35% of the time as measured by top-1 accuracy (versus 20% of the time for the best baseline), and produces a good substitution in its top-5 predictions 60% of the time (versus 44% for the best baseline). We perform a detailed error analysis that suggests future improvements, e.g. not replacing words within collocations like *elementary school*, and extending the context model to include the arguments of words that are going to be simplified.

Achieving high performance on single words is crucial for any system that hopes to adequately holistically simplify a text. Our methods can also be extended to the phrase level. SimplePPDB contains phrasal simplification rules, as well as lexical simplification rules. We can assign a vector representation to phrases to be used by the AddCos model, by applying a vector composition method to the vectors of individual words in the phrase. We plan to extend our method in this direction in future work.

Although our system outperforms simpler baselines on both tasks, the performance of the overall system is relatively low. The filtering mechanisms we have experimented with up to now in order to make high-confidence predictions, increased the quality of the proposed substitutions but significantly decreased the coverage. We will explore other ways for promoting high-quality substitutions without hurting the overall coverage of the system in the future.

The AddCos implementation we used in this work does not rely on syntactic annotations and can be easily applied to new languages. In future work, we plan to experiment with syntactic substitution models and with syntax-based word embeddings like the ones used in the initial AddCos implementation (Melamud et al., 2015). We expect syntactic information to further enhance the quality of the proposed substitutions, ensuring the functional similarity of the lexical substitutions to the target word. Furthermore, we intend to integrate lexical and syntactic simplification, both crucial steps towards text simplification.

8 Data and Software

We release the data that we collected, which is of higher quality than the data used in previous shared tasks on Complex Word Identification. We also release our software for performing context-aware paraphrase substitutions. The dataset and the code can be found at https://rekriz11.github.io

9 Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments and feedback on this work, and Anne Cocos for sharing with us her implementation of the AddCos model with PPDB paraphrase substitutes.

This material is based in part on research sponsored by DARPA under grant number FA8750-13-2-0017 (the DEFT program) and HR0011-15-C-0115 (the LORELEI program). The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes. The views and conclusions contained in this publication are those of the authors and should not be interpreted as representing official policies or endorsements of DARPA and the U.S. Government. The work has also been supported by the French National Research Agency under project ANR-16-CE33-0013. Finally, we gratefully acknowledge the support of NSF-SBIR grant 1456186.

References

Mandya Angrosh, Tadashi Nomoto, and Advaith Siddharthan. 2014. Lexico-syntactic text simplification and compression with typed dependencies. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin, Ireland, pages 1996–2006.

Marianna Apidianaki. 2016. Vector-space models for PPDB paraphrase ranking in context. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas, pages 2028–2034.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with Bilingual Parallel Corpora. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*. Ann Arbor, Michigan, pages 597–604.

Or Biran, Samuel Brody, and Noemie Elhadad. 2011. Putting it simply: a context-aware approach to lexical simplification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA, pages 496–501.

Thorsten Brants and Alex Franz. 2006. Web 1t 5-gram version 1 ldc2006t13 .

R Chandrasekar and B Srinivas. 1997. Automatic induction of rules for text simplification 10:183–190.

Anne Cocos, Marianna Apidianaki, and Chris Callison-Burch. 2017. Word Sense Filtering Improves Embedding-Based Lexical Substitution. In *Proceedings of the 1st Workshop on Sense, Concept and Entity Representations and their Applications*. Valencia, Spain, pages 110–119.

Will Coster and David Kauchak. 2011. Learning to simplify sentences using wikipedia. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*. Portland, Oregon, pages 1–9.

Siobhan Devlin and John Tait. 1998. The use of a psycholinguistic database in the simplification of text for aphasic readers. *Linguistic Databases* pages 161–173.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia, pages 758–764.

David Kauchak. 2013. Improving text simplification language modeling using unsimplified text data. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Sofia, Bulgaria, pages 1537–1546.

Sigrid Klerke and Anders Søgaard. 2013. Simple, readable sub-sentences. In *51st Annual Meeting of the Association for Computational Linguistics Proceedings of the Student Research Workshop*. Sofia, Bulgaria, pages 142–149.

Gerhard Kremer, Katrin Erk, Sebastian Padó, and Stefan Thater. 2014. What substitutes tell us - analysis of an "all-words" lexical substitution corpus. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. Gothenburg, Sweden, pages 540–549.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Baltimore, Maryland, pages 55–60.

Diana McCarthy and Roberto Navigli. 2007. Semeval-2007 task 10: English lexical substitution task. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*. Prague, Czech Republic, pages 48–53.

Oren Melamud, Omer Levy, and Ido Dagan. 2015. A Simple Word Embedding Model for Lexical Substitution. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*. Denver, Colorado, pages 1–7.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

George A. Miller. 1995. WordNet: A lexical database for English. *Communications of the ACM* 38(11):39–41.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated Gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX)*. Montreal, Canada, pages 95–100.

Shashi Narayan and Claire Gardent. 2016. Unsupervised sentence simplification using deep semantics. In *Proceedings of the 2016 International Conference (INLG)*. Edinburgh, Scotland, pages 111—120.

Gustavo Paetzold and Lucia Specia. 2016. SemEval 2016 Task 11: Complex Word Identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. Association for Computational Linguistics, San Diego, California, pages 560–569.

Ellie Pavlick and Chris Callison-Burch. 2016. Simple PPDB: A Paraphrase Database for Simplification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*. Berlin, Germany.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Beijing, China, pages 425–430.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta, pages 45–50.

Horacio Saggion. 2017. *Automatic Text Simplification*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Matthew Shardlow. 2013a. A Comparison of Techniques to Automatically Identify Complex Words. In *51st Annual Meeting of the Association for Computational Linguistics Proceedings of the Student Research Workshop*. Association for Computational Linguistics, Sofia, Bulgaria, pages 103–109.

Matthew Shardlow. 2013b. The cw corpus: A new resource for evaluating the identification of complex words. In *Proceedings of the Second Workshop on Predicting and Improving Text Readability for Target Reader Populations*. Sofia, Bulgaria, pages 69–77.

Matthew Shardlow. 2014. A Survey of Automated Text Simplification. *International Journal of Advanced Computer Science and Applications(IJACSA), Special Issue on Natural Language Processing* 2:58–70.

Lucia Specia, Sujay Kumar Jauhar, and Rada Mihalcea. 2012. Semeval-2012 task 1: English lexical simplification. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*. Montréal, Canada, pages 347–355.

Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2014. Back to Basics for Monolingual Alignment: Exploiting Word Similarity and Contextual Evidence. *Transactions of the Association for Computational Linguistics* 2:219–230.

Stefan Thater, Hagen Fürstenau, and Manfred Pinkal. 2011. Word meaning in context: A simple and effective vector model. In *Proceedings of 5th International Joint Conference on Natural Language Processing*. Asian Federation of Natural Language Processing, Chiang Mai, Thailand, pages 1134–1143.

S. Rebecca Thomas and Sven Anderson. 2012. Wordnet-based lexical simplification of a document. In Jeremy Jancsary, editor, *Proceedings of KONVENS 2012*. ÖGAI, pages 80–88.

Kristian Woodsend and Mirella Lapata. 2011. Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh, Scotland, UK., pages 409–420.

Sander Wubben, Antal van den Bosch, and Emiel Krahmer. 2012. Sentence simplification by monolingual machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Jeju Island, Korea, pages 1015–1024.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in current text simplification research: New data can help. *Transactions of the Association for Computational Linguistics* 3:283–297.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark, pages 584–594.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A Monolingual Tree-based Translation Model for Sentence Simplification. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. Coling 2010 Organizing Committee, Beijing, China, pages 1353–1361.

Zero-Shot Question Generation from Knowledge Graphs for Unseen Predicates and Entity Types

Hady Elsahar, Christophe Gravier, Frederique Laforest
Université de Lyon
Laboratoire Hubert Curien
Saint-Étienne, France
{firstname.lastname}@univ-st-etienne.fr

Abstract

We present a neural model for question generation from knowledge base triples in a "Zero-Shot" setup, that is generating questions for triples containing predicates, subject types or object types that were not seen at training time. Our model leverages triples occurrences in the natural language corpus in an encoder-decoder architecture, paired with an original part-of-speech copy action mechanism to generate questions. Benchmark and human evaluation show that our model sets a new state-of-the-art for zero-shot QG.

1 Introduction

Questions Generation (QG) from Knowledge Graphs is the task consisting in generating natural language questions given an input knowledge base (KB) triple (Serban et al., 2016). QG from knowledge graphs has shown to improve the performance of existing factoid question answering (QA) systems either by dual training or by augmenting existing training datasets (Dong et al., 2017; Khapra et al., 2017). Those methods rely on large-scale annotated datasets such as SimpleQuestions (Bordes et al., 2015). Building such datasets is a tedious task in practice, especially to obtain an unbiased dataset – i.e. a dataset that covers equally a large amount of triples in the KB. In practice many of the predicates and entity types in KB are not covered by those annotated datasets. For example 75.6% of Freebase predicates are not covered by the SimpleQuestions dataset [1]. Among those we can find important missing predicates such as: `fb:food/beer/country`, `fb:location/country/national_anthem`, `fb:astronomy/star_system/stars`.

One challenge for QG from knowledge graphs is to adapt to predicates and entity types that were *not* seen at training time (Zero-Shot Question Generation). Since state-of-the-art systems in factoid QA rely on the tremendous efforts made to create SimpleQuestions, these systems can only process questions on the subset of 24.4% of freebase predicates defined in SimpleQuestions. Previous works for factoid QG (Serban et al., 2016) claims to solve the issue of small size QA datasets. However encountering an unseen predicate / entity type will generate questions made out of random text generation for those out-of-vocabulary predicates a QG system had never seen. We go beyond this state-of-the-art by providing an original and non-trivial solution for creating a much broader set of questions for unseen predicates and entity types. Ultimately, generating questions to predicates and entity types unseen at training time will allow QA systems to cover predicates and entity types that would not have been used for QA otherwise.

Intuitively, a human who is given the task to write a question on a fact offered by a KB, would read natural language sentences where the entity or the predicate of the fact occur, and build up questions that are aligned with what he reads from both a lexical and grammatical standpoint. In this paper, we propose a model for Zero-Shot Question Generation that follows this intuitive process. In addition to the input KB triple, we feed our model with a set of textual contexts paired with the input KB triple through distant supervision. Our model derives an encoder-decoder architecture, in which the encoder encodes the input KB triple, along with a set of textual contexts into hidden representations. Those hidden representations are fed to a decoder equipped with an attention mechanism to generate an output question.

In the Zero-Shot setup, the emergence of new predicates and new class types during test time requires new lexicalizations to express these pred-

[1] replicate the observation `http://bit.ly/2GvVHae`

Proceedings of NAACL-HLT 2018, pages 218–228
New Orleans, Louisiana, June 1 - 6, 2018.

icates and classes in the output question. These lexicalizations might not be encountered by the model during training time and hence do not exist in the model vocabulary, or have been seen only few times not enough to learn a good representation for them by the model. Recent works on Text Generation tackle the rare words/unknown words problem using copy actions (Luong et al., 2015; Gülçehre et al., 2016): words with a specific position are copied from the source text to the output text – although this process is blind to the role and nature of the word in the source text. Inspired by research in open information extraction (Fader et al., 2011) and structure-content neural language models (Kiros et al., 2014), in which part-of-speech tags represent a distinctive feature when representing relations in text, we extend these positional copy actions. Instead of copying a word in a specific position in the source text, our model copies a word with a specific part-of-speech tag from the input text – we refer to those as part-of-speech copy actions. Experiments show that our model using contexts through distant supervision significantly outperforms the strongest baseline among six (+2.04 BLEU-4 score). Adding our copy action mechanism further increases this improvement (+2.39). Additionally, a human evaluation complements the comprehension of our model for edge cases; it supports the claim that the improvement brought by our copy action mechanism is even more significant than what the BLEU score suggests.

2 Related Work

QG became an essential component in many applications such as education (Heilman and Smith, 2010), tutoring (Graesser et al., 2004; Evens and Michael, 2006) and dialogue systems (Shang et al., 2015). In our paper we focus on the problem of QG from structured KB and how we can generalize it to unseen predicates and entity types. (Seyler et al., 2015) generate quiz questions from KB triples. Verbalization of entities and predicates relies on their existing labels in the KB and a dictionary. (Serban et al., 2016) use an encoder-decoder architecture with attention mechanism trained on the SimpleQuestions dataset (Bordes et al., 2015). (Dong et al., 2017) generate paraphrases of given questions to increases the performance of QA systems; paraphrases are generated relying on paraphrase datasets, neural machine translation and rule mining. (Khapra et al., 2017) generate a set of QA pairs given a KB entity. They model the problem of QG as a sequence to sequence problem by converting all the KB entities to a set of keywords. None of the previous work in QG from KB address the question of generalizing to unseen predicates and entity types.
Textual information has been used before in the Zero-Shot learning. (Socher et al., 2013) use information in pretrained word vectors for Zero-Shot visual object recognition. (Levy et al., 2017) incorporates a natural language question to the relation query to tackle Zero-Shot relation extraction problem.

Previous work in machine translation dealt with rare or unseen word problem problem for translating names and numbers in text. (Luong et al., 2015) propose a model that generates positional placeholders pointing to some words in source sentence and copy it to target sentence (*copy actions*). (Gülçehre et al., 2016; Gu et al., 2016) introduce separate trainable modules for copy actions to adapt to highly variable input sequences, for text summarization. For text generation from tables, (Lebret et al., 2016) extend positional copy actions to copy values from fields in the given table. For QG, (Serban et al., 2016) use a placeholder for the subject entity in the question to generalize to unseen entities. Their work is limited to unseen entities and does not study how they can generalize to unseen predicates and entity types.

3 Model

Let $F = \{s, p, o\}$ be the input fact provided to our model consisting of a subject s, a predicate p and an object o, and C be the set of textual contexts associated to this fact. Our goal is to learn a model that generates a sequence of T tokens $Y = y_1, y_2, \ldots, y_T$ representing a question about the subject s, where the object o is the correct answer. Our model approximates the conditional probability of the output question given an input fact $p(Y|F)$, to be the probability of the output question, given an input fact and the additional textual context C, modelled as follows:

$$p(Y|F) = \prod_{t=1}^{T} p(y_t|y_{<t}, F, C) \quad (1)$$

where $y_{<t}$ represents all previously generated tokens until time step t. Additional textual contexts are natural language representation of the triples

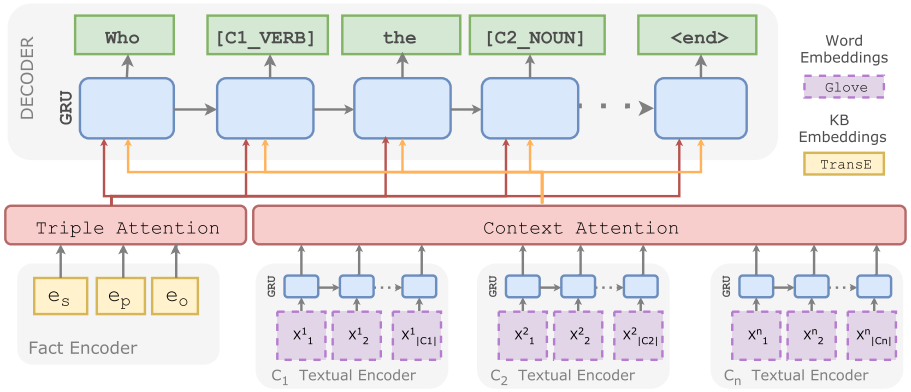


Figure 1: The proposed model for Question Generation. The model consists of a single fact encoder and n textual context encoders, each consists of a separate GRU. At each time step t, two attention vectors generated from the two attention modules are fed to the decoder to generate the next word in the output question.

that can be drawn from a corpus – our model is generic to any textual contexts that can be additionally provided, though we describe in Section 4.1 how to create such texts from Wikipedia.

Our model derives the encoder-decoder architecture of (Sutskever et al., 2014; Bahdanau et al., 2014) with two encoding modules: a feed forward architecture encodes the input triple (sec. 3.1) and a set of recurrent neural network (RNN) to encode each textual context (sec. 3.2). Our model has two attention modules (Bahdanau et al., 2014): one acts over the input triple and another acts over the input textual contexts (sec. 3.4). The decoder (sec. 3.3) is another RNN that generates the output question. At each time step, the decoder chooses to output either a word from the vocabulary or a special token indicating a copy action (sec. 3.5) from any of the textual contexts.

3.1 Fact Encoder

Given an input fact $F = \{s, p, o\}$, let each of e_s, e_p and e_o be a 1-hot vectors of size K. The fact encoder encodes each 1-hot vector into a fixed size vector $h_s = \mathbf{E_f}\, e_s$, $h_p = \mathbf{E_f}\, e_p$ and $h_o = \mathbf{E_f}\, e_o$, where $\mathbf{E_f} \in \mathbb{R}^{H_k \times K}$ is the KB embedding matrix, H_k is the size of the KB embedding and K is the size of the KB vocabulary. The *encoded fact* $h_f \in \mathbb{R}^{3H_k}$ represents the concatenation of those three vectors and we use it to initialize the decoder.

$$h_f = [h_s;\ h_p;\ h_o] \tag{2}$$

Following (Serban et al., 2016), we learn $\mathbf{E_f}$ using *TransE* (Bordes et al., 2015). We fix its weights and do not allow their update during training time.

3.2 Textual Context Encoder

Given a set of n textual contexts $C = \{c_1, c_2, \ldots, c_n\ : c_j = (x_1^j, x_2^j, \ldots, x_{|c_j|}^j)\}$, where x_i^j represents the 1-hot vector of the i^{th} token in the j^{th} textual context c_j, and $|c_j|$ is the length of the j^{th} context. We use a set of n Gated Recurrent Neural Networks (GRU) (Cho et al., 2014) to encode each of the textual concepts separately:

$$h_i^{c_j} = GRU_j\left(\mathbf{E_c}\, x_i^j,\ \ h_{i-1}^{c_j}\right) \tag{3}$$

where $h_i^{c_j} \in \mathbb{R}^{H_c}$ is the hidden state of the GRU that is equivalent to x_i^j and of size H_c . $\mathbf{E_c}$ is the input word embedding matrix. The *encoded context* represents the encoding of all the textual contexts; it is calculated as the concatenation of all the final states of all the encoded contexts:

$$h_c = [h_{|c_1|}^{c_1}; h_{|c_2|}^{c_2}; \ldots; h_{|c_n|}^{c_n}]. \tag{4}$$

3.3 Decoder

For the decoder we use another GRU with an attention mechanism (Bahdanau et al., 2014), in which the decoder hidden state $s_t \in \mathbb{R}^{H_d}$ at each time step t is calculated as:

$$s_t = z_t \circ s_{t-1} + (1 - z_t) \circ \tilde{s}_t\ , \tag{5}$$

Where:

$$\tilde{s}_t = tanh\left(W E_w y_{t-1} + U[r_t \circ s_{t-1}] + A\,[a_t^f; a_t^c]\right) \tag{6}$$

$$z_t = \sigma\left(W_z\, E_w\, y_{t-1} + U_z\, s_{t-1} + A_z\,[a_t^f; a_t^c]\right) \tag{7}$$

$$r_t = \sigma\left(W_r\, E_w\, y_{t-1} + U_r\, s_{t-1} + A_r\,[a_t^f; a_t^c]\right) \tag{8}$$

$W, W_z, W_r \in \mathbb{R}^{m \times H_d}$, $U, U_z, U_r, A, A_z, A_r \in \mathbb{R}^{H_d \times H_d}$ are learnable parameters of the GRU.

$E_w \in \mathbf{R}^{m \times V}$ is the word embedding matrix, m is the word embedding size and H_d is the size of the decoder hidden state. a_t^f, a_t^c are the outputs of the fact attention and the context attention modules respectively, detailed in the following subsection.
In order to enforce the model to pair output words with words from the textual inputs, we couple the word embedding matrices of both the decoder E_w and the textual context encoder E_c (eq.(3)). We initialize them with GloVe embeddings (Pennington et al., 2014) and allow the network to tune them.
The first hidden state of the decoder $s_0 = [h_f;\ h_c]$ is initialized using a concatenation of the encoded fact (eq.(2)) and the encoded context (eq.(4)) .
At each time step t, after calculating the hidden state of the decoder, the conditional probability distribution over each token y_t of the generated question is computed as the $softmax(W_o\, s_t)$ over all the entries in the output vocabulary, $W_o \in \mathbb{R}^{H_d \times V}$ is the weight matrix of the output layer of the decoder.

3.4 Attention

Our model has two attention modules:
Triple attention over the input triple to determine at each time step t an attention-based encoding of the input fact $a_t^f \in \mathbb{R}^{H_k}$:

$$a_t^f = \alpha_{s,t}\, h_s + \alpha_{p,t}\, h_p + \alpha_{s,t}\, h_o \ , \qquad (9)$$

$\alpha_{s,t}, \alpha_{p,t}, \alpha_{o,t}$ are scalar values calculated by the attention mechanism to determine at each time step which of the encoded subject, predicate, or object the decoder should attend to.
Textual contexts attention over all the hidden states of all the textual contexts $a_t^c \in \mathbb{R}^{H_c}$:

$$a_t^c = \sum_{i=1}^{|C|} \sum_{j=1}^{|c_i|} \alpha_{t,j}^{c_i}\, h_j^{c_i} \ , \qquad (10)$$

$\alpha_{t,j}^{c_i}$ is a scalar value determining the weight of the j^{th} word in the i^{th} context c^i at time step t.

Given a set of encoded input vectors $I = \{h_1, h_2, ...h_k\}$ and the decoder previous hidden state s_{t-1}, the attention mechanism calculates $\alpha_t = \alpha_{i,t}, \ldots, \alpha_{k,t}$ as a vector of scalar weights, each $\alpha_{i,t}$ determines the weight of its corresponding encoded input vector h_i.

$$e_{i,t} = \mathbf{v_a}^\top\ tanh(\mathbf{W_a}\, \mathbf{s_{t-1}} + \mathbf{U_a}\, \mathbf{h_i}) \qquad (11)$$

$$\alpha_{i,t} = \frac{exp\,(e_{i,t})}{\sum_{j=1}^{k} exp\,(e_{j,t})} \ , \qquad (12)$$

where $\mathbf{v_a}, \mathbf{W_a}, \mathbf{U_a}$ are trainable weight matrices of the attention modules. It is important to notice here that we encode each textual context separately using a different GRU, but we calculate an overall attention over all tokens in all textual contexts: at each time step the decoder should ideally attend to only one word from all the input contexts.

What caused the `[C1_NOUN]` of the `[C3_NOUN]` `[S]` ?	
C1	`[S]` **`death`** `by [O]`
	`[S]` **`[C1_NOUN]`** `[C1_ADP] [O]`
C2	Disease
	`[C2_NOUN]`
C3	Musical **artist**
	`[C3_ADJ]` **`[C3_NOUN]`**

Table 1: An annotated example of part-of-speech copy actions from several input textual contexts (C1, C2, C3), the words or placeholders in bold are copied in the generated question

3.5 Part-Of-Speech Copy Actions

We use the method of (Luong et al., 2015) by modeling all the copy actions on the data level through an annotation scheme. This method treats the model as a black box, which makes it adaptable to any text generation model. Instead of using positional copy actions, we use the part-of-speech information to decide the alignment process between the input and output texts to the model. Each word in every input textual context is replaced by a special token containing a combination of its context id (e.g. `C1`) and its POS tag (e.g. `NOUN`). Then, if a word in the output question matches a word in a textual context, it is replaced with its corresponding tag as shown in Table 1.
Unlike (Serban et al., 2016; Lebret et al., 2016) we model the copy actions in the input and the output levels. Our model does not have the drawback of losing the semantic information when replacing words with generic placeholders, since we provide the model with the input triple through the fact encoder. During inference the model chooses to either output words from the vocabulary or special tokens to copy from the textual contexts. In

a post-processing step those special tokens are replaced with their original words from the textual contexts.

4 Textual contexts dataset

As a source of question paired with KB triples we use the SimpleQuestions dataset (Bordes et al., 2015). It consists of 100K questions with their corresponding triples from Freebase, and was created manually through crowdsourcing. When asked to form a question from an input triple, human annotators usually tend to mainly focus on expressing the predicate of the input triple. For example, given a triple with the predicate `fb:spacecraft/manufacturer` the user may ask *"What is the manufacturer of `[S]` ?"*. Annotators may specify the entity type of the subject or the object of the triple: *"What is the manufacturer of the **spacecraft** `[S]`?"* or *"Which **company** manufactures `[S]`?"*. Motivated by this example we chose to associate each input triple with three textual contexts of three different types. The first is a phrase containing lexicalization of the predicate of the triple. The second and the third are two phrases containing the entity type of the subject and the object of the triple. In what follows we show the process of collection and preprocessing of those textual contexts.

4.1 Collection of Textual Contexts

We extend the set of triples given in the SimpleQuestions dataset by using the FB5M (Bordes et al., 2015) subset of Freebase. As a source of text documents, we rely on Wikipedia articles.

Predicate textual contexts: In order to collect textual contexts associated with the SimpleQuestions triples, we follow the distant supervision setup for relation extraction (Mintz et al., 2009). The distant supervision assumption has been effective in creating training data for relation extraction and shown to be 87% correct (Riedel et al., 2010) on Wikipedia text.
First, we align each triple in the FB5M KB to sentences in Wikipedia if the subject and the object of this triple co-occur in the same sentence. We use a simple string matching heuristic to find entity mentions in text[2]. Afterwards we reduce the sentence to the set of words that appear on the dependency path between the subject and the object mentions in the sentence. We replace the positions of the subject and the object mentions with `[S]` and `[O]` to the keep track of the information about the direction of the relation. The top occurring pattern for each predicate is associated to this predicate as its textual context. Table 2 shows examples of predicates and their corresponding textual context.

Freebase Relation	Predicate Textual Context
person/place_of_birth	[O] is birthplace of [S]
currency/former_countries	[S] was currency of [O]
dish/cuisine	[O] dish [S]
airliner_accident/flight_origin	[S] was flight from [O]
film_featured_song/performer	[S] is release by [O]
airline_accident/operator	[S] was accident for [O]
genre/artists	[S] became a genre of [O]
risk_factor/diseases	[S] increases likelihood of [O]
book/illustrations_by	[S] illustrated by [O]
religious_text/religion	[S] contains principles of [O]
spacecraft/manufacturer	[S] spacecraft developed by [O]

Table 2: Table showing an example of textual contexts extracted for freebase predicates

Sub-Type and Obj-Type textual contexts: We use the labels of the entity types as the sub-type and obj-type textual contexts. We collect the list of entity types of each entity in the FB5M through the predicate `fb:type/instance`. If an entity has multiple entity types we pick the entity type that is mentioned the most in the first sentence of each Wikipedia article. Thus the textual contexts will opt for entity types that is more natural to appear in free text and therefore questions.

4.2 Generation of Special tokens

To generate the special tokens for copy actions (sec. 3.5) we run POS tagging on each of the input textual contexts[3]. We replace every word in each textual context with a combination of its context id (e.g. `C1`) and its POS tag (e.g. `NOUN`). If the same POS tag appears multiple times in the textual context, it is given an additional id (e.g. `C1_NOUN_2`). If a word in the output question overlaps with a word in the input textual context, this word is replaced by its corresponding tag.
For sentence and word tokenization we use the Regex tokenizer from the NLTK toolkit (Bird, 2006), and for POS tagging and dependency pars-

[2] We map Freebase entities to Wikidata through the Wikidata property P646, then we extract their labels and aliases. We use the Wikidata truthy dump: `https://dumps.wikimedia.org/wikidatawiki/entities/`

[3] For the predicate textual contexts we run pos tagging on the original text not the lexicalized dependency path

		Train	Valid	Test
pred	# pred	169.4	24.2	48.4
	# samples	55566.7	7938.1	15876.2
	% samples	70.0 ± 2.77	10.0 ± 1.236	20.0 ± 2.12
sub-types	# types	112.7	16.1	32.2
	# samples	60002.6	8571.8	17143.6
	% samples	70.0 ± 7.9	10.0 ± 3.6	20.0 ± 6.2
obj-types	# types	521.6	189.9	282.2
	# samples	57878.1	8268.3	16536.6
	% samples	70.0 ± 4.7	10.0 ± 2.5	20.0 ± 3.8

Table 3: Dataset statistics across 10 folds for each experiment

ing we use the Spacy[4] implementation.

5 Experiments

5.1 Zero-Shot Setups

We develop three setups that follow the same procedure as (Levy et al., 2017) for Zero-Shot relation extraction to evaluate how our model generalizes to: 1) unseen predicates, 2) unseen sub-types and 3) unseen obj-types.

For the unseen predicates setup we group all the samples in SimpleQuestions by the predicate of the input triple, and keep groups that contain at least 50 samples. Afterwards we randomly split those groups to 70% train, 10% valid and 20% test mutual exclusive sets respectively. This guarantees that if the predicate `fb:person/place_of_birth` for example shows during test time, the training and validation set will not contain any input triples having this predicate. We repeat this process to create 10 cross validation folds, in our evaluation we report the mean and standard deviation results across those 10 folds. While doing this we make sure that the number of samples in each fold – not only unique predicates – follow the same 70%, 30%, 10% distribution. We repeat the same process for the subject entity types and object entity types (answer types) individually. Similarly, for example in the unseen object-type setup, the question *"Which **artist** was born in Berlin?"* appearing in the test set means that, there is no question in the training set having an entity of type ***artist***. Table 3 shows the mean number of samples, predicates, sub-types and obj-types across the 10 folds for each experiment setup.

[4]https://spacy.io/

5.2 Baselines

`SELECT` is a baseline built from (Serban et al., 2016) and adapted for the zero shot setup. During test time given a fact F, this baseline picks a fact F_c from the training set and outputs the question that corresponds to it. For evaluating unseen predicates, F_c has the same answer type (obj-type) as F. And while evaluating unseen sub-types or obj-types, F_c and F have the same predicate.

`R-TRANSE` is an extension that we propose for `SELECT`. The input triple is encoded using the concatenation of the TransE embeddings of the subject, predicate and object. At test time, `R-TRANSE` picks a fact from the training set that is the closest to the input fact using cosine similarity and outputs the question that corresponds to it. We provide two versions of this baseline: **`R-TRANSE`** which indexes and retrieves raw questions with only a single placeholder for the subject label, such as in (Serban et al., 2016). And **`R-TRANSE`**$_{\texttt{copy}}$ which indexes and retrieves questions using our copy actions mechanism (sec. 3.5).

`IR` is an information retrieval baseline. Information retrieval has been used before as baseline for QG from text input (Rush et al., 2015; Du et al., 2017). We rely on the textual context of each input triple as the search keyword for retrieval. First, the IR baseline encodes each question in the training set as a vector of TF-IDF weights (Joachims, 1997) and then does dimensionality reduction through LSA (Halko et al., 2011). At test time the textual context of the input triple is converted into a dense vector using the same process and then the question with the closest cosine distance to the input is retrieved. We provide two versions of this baseline: **`IR`** on raw text and **`IR`**$_{\texttt{copy}}$ on text with our placeholders for copy actions.

`Encoder-Decoder.` Finally, we compare our model to the Encoder-Decoder model with a single placeholder, the best performing model from (Serban et al., 2016). We initialize the encoder with TransE embeddings and the decoder with GloVe word embeddings. Although this model was not originally built to generalize to unseen predicates and entity types, it has some generalization abilities represented in the encoded infor-

mation in the pre-trained embeddings. Pretrained KB terms and word embeddings encode relations between entities or between words as translations in the vector space. Thus the model might be able to map new classes or predicates in the input fact to new words in the output question.

5.3 Training & Implementation Details

To train the neural network models we optimize the negative log-likelihood of the training data with respect to all the model parameters. For that we use the RMSProp optimization algorithm with a decreasing learning rate of 0.001, mini-batch size $= 200$, and clipping gradients with norms larger than 0.1. We use the same vocabulary for both the textual context encoders and the decoder outputs. We limit our vocabulary to the top $30,000$ words including the special tokens. For the word embeddings we chose GloVe (Pennington et al., 2014) pretrained embeddings of size 100. We train TransE embeddings of size $H_k = 200$, on the FB5M dataset (Bordes et al., 2015) using the TransE model implementation from (Lin et al., 2015). We set GRU hidden size of the decoder to $H_d = 500$, and textual encoder to $H_c = 200$. The networks hyperparameters are set with respect to the final BLEU-4 score over the validation set. All neural networks are implemented using Tensorflow (Abadi et al., 2015). All experiments and models source code are publicly available[5] for the sake of reproducibility.

5.4 Automatic Evaluation Metrics

To evaluate the quality of the generated question, we compare the original labeled questions by human annotators to the ones generated by each variation of our model and the baselines. We rely on a set of well established evaluation metrics for text generation: BLEU-1, BLEU-2, BLEU-3, BLEU-4 (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014) and ROUGE_L (Lin, 2004).

5.5 Human Evaluation

Automatic Metrics for evaluating text generation such as BLEU and METEOR give an measure of how close the generated questions are to the target correct labels. However, they still suffer from many limitations (Novikova et al., 2017). Automatic metrics might not be able to evaluate directly whether a specific predicate was explicitly mentioned in the generated text or not.
As an example, taking a target question and two corresponding generated questions A and B:

	What kind of film is kill bill vol. 2?	**BLEU**
A)	**What is *the name of* the film kill bill vol. 2?**	71
B)	**Which genre is kill bill vol. 2 in?**	55

We can find that the sentence A having a better BLEU score than B although it is not able to express the correct target predicate (*film genre*). For that reason we decide to run two further human evaluations to directly measure the following:
Predicate identification: annotators were asked to indicate whether the generated question contains the given predicate in the fact or not, either directly or implicitly.
Naturalness: following (Ngomo et al., 2013), we measure the comprehensibility and readability of the generated questions. Each annotator was asked to rate each generated question using a scale from 1 to 5, where: (5) perfectly clear and natural, (3) artificial but understandable, and (1) completely not understandable. We run our studies on 100 randomly sampled input facts alongside with their corresponding generated questions by each of the systems using the help of 4 annotators.

6 Results & Discussion

Automatic Evaluation Table 4 shows results of our model compared to all other baselines across all evaluation metrics. Our that encodes the KB fact and textual contexts achieves a significant enhancement over all the baselines in all evaluation metrics, with $+\mathbf{2.04}$ BLEU-4 score than the Encoder-Decoder baseline. Incorporating the part-of-speech copy actions further improves this enhancement to reach $+\mathbf{2.39}$ BLEU-4 points.
Among all baselines, the Encoder-Decoder baseline and the R-TRANSE baseline performed the best. This shows that TransE embeddings encode intra-predicates information and intra-class-types information to a great extent, and can generalize to some extent to unseen predicates and class types.

Similar patterns can be seen in the evaluation on unseen sub-types and obj-types (Table 5). Our model with copy actions was able to outperform

[5] https://github.com/hadyelsahar/Zeroshot-QuestionGeneration

	Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE_L	METEOR
Unseen Predicates	SELECT	46.81 ± 2.12	38.62 ± 1.78	31.26 ± 1.9	23.66 ± 2.22	52.04 ± 1.43	27.11 ± 0.74
	IR	48.43 ± 1.64	39.13 ± 1.34	31.4 ± 1.66	23.59 ± 2.36	52.88 ± 1.24	27.34 ± 0.55
	IR_COPY	48.22 ± 1.84	38.82 ± 1.5	31.01 ± 1.72	23.12 ± 2.24	52.72 ± 1.26	27.24 ± 0.57
	R-TRANSE	49.09 ± 1.69	40.75 ± 1.42	33.4 ± 1.7	25.97 ± 2.22	54.07 ± 1.31	28.13 ± 0.54
	$\text{R-TRANSE}_\text{COPY}$	49.0 ± 1.76	40.63 ± 1.48	33.28 ± 1.74	25.87 ± 2.23	54.09 ± 1.35	28.12 ± 0.57
	Encoder-Decoder	58.92 ± 2.05	47.7 ± 1.62	38.18 ± 1.86	28.71 ± 2.35	59.12 ± 1.16	34.28 ± 0.54
	Our-Model	60.8 ± 1.52	49.8 ± 1.37	40.32 ± 1.92	30.76 ± 2.7	60.07 ± 0.9	35.34 ± 0.43
	$\text{Our-Model}_\text{copy}$	**62.44** ± 1.85	**50.62** ± 1.46	**40.82** ± 1.77	**31.1** ± 2.46	**61.23** ± 1.2	**36.24** ± 0.65

Table 4: Evaluation results of our model and all other baselines for the unseen predicate evaluation setup

	Model	BLEU-4	ROUGE_L
Sub-Types	R-TRANSE	32.41 ± 1.74	59.27 ± 0.92
	Encoder-Decoder	42.14 ± 2.05	68.95 ± 0.86
	Our-Model	42.13 ± 1.88	69.35 ± 0.9
	$\text{Our-Model}_\text{copy}$	**42.2** ± 2.0	**69.37** ± 1.0
Obj-Types	R-TRANSE	30.59 ± 1.3	57.37 ± 1.17
	Encoder-Decoder	37.79 ± 2.65	65.69 ± 2.25
	Our-Model	37.78 ± 2.02	65.51 ± 1.56
	$\text{Our-Model}_\text{copy}$	**38.02** ± 1.9	**66.24** ± 1.38

Table 5: Automatic evaluation of our model against selected baselines for unseen sub-types and obj-types

Model	% Pred. Identified	Natural.
Encoder-Decoder	6	3.14
Our-Model (No Copy)	6	2.72
$\text{Our-Model}_\text{copy}$ (Types context)	**37**	**3.21**
$\text{Our-Model}_\text{copy}$ (All contexts)	**46**	2.61

Table 6: results of Human evaluation on % of predicates identified and naturalness 0-5

all the other systems. Majority of systems have reported a significantly higher BLEU-4 scores in these two tasks than when generalizing to unseen predicates (+12 and +8 BLEU-4 points respectively). This indicates that these tasks are relatively easier and hence our models achieve relatively smaller enhancements over the baselines.

Human Evaluation Table 6 shows how different variations of our system can express the unseen predicate in the target question with comparison to the Encoder-Decoder baseline.
Our proposed copy actions have scored a significant enhancement in the identification of unseen predicates with up to +40% more than best performing baseline and our model version without the copy actions.

By examining some of the generated questions (Table 7) we see that models without copy actions can generalize to unseen predicates that only have a very similar freebase predicate in the training set. For example `fb:tv_program/language` and `fb:film/language`, if one of those predicates exists in the training set the model can use the same questions for the other during test time.
Copy actions from the sub-type and the obj-type textual contexts can generalize to a great extent to unseen predicates because of the overlap between the predicate and the object type in many questions (Example 2 Table 7). Adding the predicate context to our model has enhanced model performance for expressing unseen predicates by +9% (Table 6). However we can see that it has affected the naturalness of the question. The post processing step does not take into consideration that some verbs and prepositions do not fit in the sentence structure, or that some words are already existing in the question words (Example 4 Table 7). This does not happen as much when having copy actions from the sub-type and the obj-type contexts because they are mainly formed of nouns which are more interchangeable than verbs or prepositions. A post-processing step to reform the question instead of direct copying from the input source is considered in our future work.

7 Conclusion

In this paper we presented a new neural model for question generation from knowledge bases, with a main focus on predicates, subject types or object types that were not seen at the training phase (Zero-Shot Question Generation). Our model is based on an encoder-decoder architecture that leverages textual contexts of triples, two attention layers for triples and textual contexts and

1	**Reference**	**what language is spoken in the tv show three sheets?**
	Enc-Dec.	in what **language** is three sheets in?
	Our-Model	what the the player is the three sheets?
	Our-Model$_{\text{Copy}}$	what is the **language** of three sheets?
2	**Reference**	**how is roosevelt in Africa classified?**
	Enc-Dec.	what is the name of a roosevelt in Africa?
	Our-Model	what is the name of the movie roosevelt in Africa?
	Our-Model$_{\text{Copy}}$	what is a **genre** of roosevelt in Africa?
3	**Reference**	**where can 5260 philvron be found**?
	Enc-Dec.	what is a release some that 5260 philvron wrote?
	Our-Model	what is the name of an artist 5260 philvron?
	Our-Model$_{\text{Copy}}$	which **star system** contains the star system body 5260 philvron?
4	**Reference**	**which university did ezra cornell create**?
	Enc-Dec.	which films are part of ezra cornell?
	Our-Model	what is a position of ezra cornell?
	Our-Model$_{\text{Copy}}$	what *founded* the name of a university that ezra cornell **founded**?
5	**Reference**	**who founded snocap , inc .**?
	Enc-Dec.	which asian snocap is most as?
	Our model	what is the name of a person of snocap?
	Our-Model$_{\text{Copy}}$	who is the **person behind** snocap?

Table 7: Examples of generated questions from different systems in comparison

finally a part-of-speech copy action mechanism. Our method exhibits significantly better results for Zero-Shot QG than a set of strong baselines including the state-of-the-art question generation from KB. Additionally, a complimentary human evaluation, helps in showing that the improvement brought by our part-of-speech copy action mechanism is even more significant than what the automatic evaluation suggests. The source code and the collected textual contexts are provided for the community [6]

[6] https://github.com/hadyelsahar/Zeroshot-QuestionGeneration

Acknowledgements

This research is partially supported by the Answering Questions using Web Data (WDAqua) project, a Marie Skłodowska-Curie Innovative Training Network under grant agreement No 642795, part of the Horizon 2020 programme.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. https://www.tensorflow.org/.

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

Steven Bird. 2006. NLTK: the natural language toolkit. In *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006*. http://aclweb.org/anthology/P06-4018.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *CoRR* abs/1506.02075. http://arxiv.org/abs/1506.02075.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR* abs/1406.1078.

Michael J. Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation, WMT@ACL 2014, June 26-27, 2014, Baltimore, Maryland, USA*. pages 376–380. http://aclweb.org/anthology/W/W14/W14-3348.pdf.

Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. 2017. Learning to paraphrase for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. pages 875–886. https://aclanthology.info/papers/D17-1091/d17-1091.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. pages 1342–1352. https://doi.org/10.18653/v1/P17-1123.

Martha Evens and Joel Michael. 2006. One-on-one tutoring by humans and machines. *Computer Science Department, Illinois Institute of Technology* .

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 1535–1545. http://www.aclweb.org/anthology/D11-1142.

Arthur C Graesser, Shulan Lu, George Tanner Jackson, Heather Hite Mitchell, Mathew Ventura, Andrew Olney, and Max M Louwerse. 2004. Autotutor: A tutor with dialogue in natural language. *Behavior Research Methods* 36(2):180–192.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. http://aclweb.org/anthology/P/P16/P16-1154.pdf.

Çaglar Gülçehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. http://aclweb.org/anthology/P/P16/P16-1014.pdf.

Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. 2011. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review* 53(2):217–288. https://doi.org/10.1137/090771806.

Michael Heilman and Noah A. Smith. 2010. Good question! statistical ranking for question generation. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 2-4, 2010, Los Angeles, California, USA*. pages 609–617. http://www.aclweb.org/anthology/N10-1086.

Thorsten Joachims. 1997. A probabilistic analysis of the rocchio algorithm with TFIDF for text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning (ICML 1997), Nashville, Tennessee, USA, July 8-12, 1997*. pages 143–151.

Mitesh M. Khapra, Dinesh Raghu, Sachindra Joshi, and Sathish Reddy. 2017. Generating natural language question-answer pairs from a knowledge graph using a RNN based question generation model. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*. pages 376–385. https://aclanthology.info/pdf/E/E17/E17-1036.pdf.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S. Zemel. 2014. Unifying visual-semantic embeddings with multimodal neural language models. *CoRR* abs/1411.2539. http://arxiv.org/abs/1411.2539.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. pages 1203–1213. http://aclweb.org/anthology/D/D16/D16-1128.pdf.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017), Vancouver, Canada, August 3-4, 2017*. pages 333–342. https://doi.org/10.18653/v1/K17-1034.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*. Barcelona, Spain, volume 8.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*. pages 2181–2187. http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9571.

Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine trans-

lation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*. pages 11–19. http://aclweb.org/anthology/P/P15/P15-1002.pdf.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2-7 August 2009, Singapore*. pages 1003–1011. http://www.aclweb.org/anthology/P09-1113.

Axel-Cyrille Ngonga Ngomo, Lorenz Bühmann, Christina Unger, Jens Lehmann, and Daniel Gerber. 2013. Sorry, i don't speak SPARQL: translating SPARQL queries into natural language. In *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*. pages 977–988. http://dl.acm.org/citation.cfm?id=2488473.

Jekaterina Novikova, Ondrej Dusek, Amanda Cercas Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for NLG. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. pages 2241–2252. https://aclanthology.info/papers/D17-1238/d17-1238.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA.*. pages 311–318. http://www.aclweb.org/anthology/P02-1040.pdf.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 1532–1543. http://aclweb.org/anthology/D/D14/D14-1162.pdf.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2010, Barcelona, Spain, September 20-24, 2010, Proceedings, Part III*. pages 148–163. https://doi.org/10.1007/978-3-642-15939-8_10.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. pages 379–389. http://aclweb.org/anthology/D/D15/D15-1044.pdf.

Iulian Vlad Serban, Alberto García-Durán, Çaglar Gülçehre, Sungjin Ahn, Sarath Chandar, Aaron C. Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. http://aclweb.org/anthology/P/P16/P16-1056.pdf.

Dominic Seyler, Mohamed Yahya, and Klaus Berberich. 2015. Generating quiz questions from knowledge graphs. In *Proceedings of the 24th International Conference on World Wide Web Companion, WWW 2015, Florence, Italy, May 18-22, 2015 - Companion Volume*. pages 113–114. https://doi.org/10.1145/2740908.2742722.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*. pages 1577–1586. http://aclweb.org/anthology/P/P15/P15-1152.pdf.

Richard Socher, Milind Ganjoo, Christopher D. Manning, and Andrew Y. Ng. 2013. Zero-shot learning through cross-modal transfer. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*. pages 935–943.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. pages 3104–3112.

Automated Essay Scoring in the Presence of Biased Ratings

Evelin Amorim
Computer Science Department
UFMG
Brazil
evelin@dcc.ufmg.br

Marcia Cançado
Linguistics Department
UFMG
Brazil
mcancado@ufmg.br

Adriano Veloso
Computer Science Department
UFMG
Brazil
adrianov@dcc.ufmg.br

Abstract

Studies in Social Sciences have revealed that when people evaluate someone else, their evaluations often reflect their biases. As a result, rater bias may introduce highly subjective factors that make their evaluations inaccurate. This may affect automated essay scoring models in many ways, as these models are typically designed to model (potentially biased) essay raters. While there is sizeable literature on rater effects in general settings, it remains unknown how rater bias affects automated essay scoring. To this end, we present a new annotated corpus containing essays and their respective scores. Different from existing corpora, our corpus also contains comments provided by the raters in order to ground their scores. We present features to quantify rater bias based on their comments, and we found that rater bias plays an important role in automated essay scoring. We investigated the extent to which rater bias affects models based on hand-crafted features. Finally, we propose to rectify the training set by removing essays associated with potentially biased scores while learning the scoring model.

1 Introduction

Automated Essay Scoring (AES) aims at developing models that can grade essays automatically or with reduced involvement of human raters (Page, 1967). AES systems may rely not only on grammars, but also on more complex features such as semantics, discourse and pragmatics (Davis and Veloso, 2016; Song et al., 2014; Farra et al., 2015; Somasundaran et al., 2014). Thus, a prominent approach to AES is to learn scoring models from previously graded samples, by modeling the scoring process of human raters. When given the same set of essays to evaluate and enough graded samples, AES systems tend to achieve high agreement levels with trained human raters (Taghipour and Ng, 2016).

While research in AES has focused on designing scoring models that maximize the agreement with human raters(Chen and He, 2013; Alikaniotis et al., 2016), there is a lack of discussion on how biased are human ratings. Despite making judgments on a common dimension, raters may be influenced by their attitudes, their cultural background, and their political and economic views (Guerra et al., 2011). Since AES models are designed to learn by analyzing human-graded essays, AES models could inherit rating biases present in the scores from human raters, and this may result in systematic errors. Thus, our objective in this paper is to examine the extent to which rater bias affects the effectiveness of state-of-the-art AES models. A deeper understanding of such factors may help mitigating the effects of rater bias, enabling AES models to achieve greater objectivity.

In order to study the effects of rater bias in essay scoring, we created an annotated corpus containing essays written by high school students as part of a standardized Brazilian national exam. Our corpus contains a number of essays, written in Portuguese, along with their respective scores. Further, raters must also provide a comment for each essay in order to ground their scores. As in (Recasens et al., 2013) we built subjectivity and sentiment lexicons that serve as features to represent the comments, that is, rater comments are represented according to the subjectivity distribution as given by specific subjectivity cues in our lexicons. We present empirical evidence suggesting that the subjectivity distribution within rater comment is a proxy for the score that is given to the essay. More specifically, very low (or very high) scores are associated with essays for which rater comments showed a very particular subjectivity distribution. We also investigated the relationship be-

Proceedings of NAACL-HLT 2018, pages 229–237
New Orleans, Louisiana, June 1 - 6, 2018.

tween subjectivity distribution and the misalignment between human raters and AES models. Interestingly, the subjectivity distribution becomes very characteristic as the misalignment increases.

Our main contributions are three-fold:

- We built subjectivity lexicons for the Portuguese language. These lexicons include words and phrases associated with different subjectivity dimensions – sentiments, factive verbs, entailments, intensifiers and hedges. We identify biased language within rater comments by calculating the word mover's distance (Kusner et al., 2015) between comments and the lexicons. This approach benefits from large unsupervised corpora, that can be used to learn effective word embeddings (Mikolov et al., 2013). By identifying biased language, we observed that biases can work to inflate essay scores or to deflate them.

- We employ a set of linguistic features in order to learn different AES models, and we evaluate the effects of biased ratings in the efficacy of these models. In summary, biased ratings affect AES models in different ways, but in general the misalignment between human rater and the AES model is more acute when the rater shows biased language in their comments.

- We propose simple ways of preventing and reducing the negative effects of biased ratings while learning AES models. Results in a controlled experimental setting revealed that detecting and removing biased ratings from the training set lead to significant improvements in automated essay scoring.

In the remainder of this paper, Section 2 discusses related work on automated essay scoring. Section 3 describes the features used for learning AES models, as well as the features used for identifying biased language in rater comments. Further, our debiasing approach is also discussed in Section 3. Section 4 describes the data, the setup and the results of our empirical evaluation. Finally, Section 5 provides our conclusions.

2 Related Work

Research in cognitive science, psychology and other social studies offer a great amount of work on (conscious and unconscious) biases and their effects on a variety of human activities (Kaheman and Tversky, 1972; Tversky and Kaheman, 1974). Biases can create situations that lead us to make decisions that project our experiences and values onto others (Baron, 2007; Ariely, 2008). While there is sizeable literature on rater effects in general settings (Myford and Wolfe, 2003), it remains unknown how biased ratings affect automated essay scoring models. Rather, works on automated essay scoring are mainly focused on designing AES models by maximizing the agreement with human raters, despite the assertiveness of the ratings.

Typically, AES systems are built on the basis of predefined linguistic features that are then given to a machine learning algorithm (Amorim and Veloso, 2017). Works that fall into this approach include (Srihari et al., 2008, 2007; Cummins et al., 2016; McNamaraa et al., 2015). Further, authors in (Dong and Zhang, 2016) presented an empirical analysis of features typically used for learning AES models. Authors in (Crossley et al., 2015) studied a broader category of features that can also be used to build AES models. There are also more recent approaches for learning AES models that do not assume a set of predefined features. These approaches are based on deep architectures, and include (Alikaniotis et al., 2016; Taghipour and Ng, 2016; Riordan et al., 2017; Dong et al., 2017). Finally, there also models based on domain adaptation (Phandi et al., 2015) and unsupervised learning (Chen et al., 2010).

Few works have investigated the subjective nature of essay scoring. An interesting exception is (Allen et al., 2015), in which the authors investigated the misalignment between students' and teachers' ratings of essay. Results revealed that students who were less accurate in their self-assessments produced essays that were more causal, contained less meaningful words, and had less argument overlap between sentences.

The work in this paper builds upon prior work on building subjectivity lexicons (Klebanov et al., 2012) and subjectivity detection (Recasens et al., 2013), but in our case applied to score agreement. In this respect, our work is more comparable to (Klebanov and Beigman, 2009; Beigman and Klebanov, 2009), where authors discussed and investigated the problem of learning in the presence of biased annotators. Other works that are also

close to ours include (Farra et al., 2015; Somasundaran et al., 2016; Song et al., 2014), in which the authors studied the problem of scoring persuasive and argumentative essays.

3 Method

Our aim in this work is to learn AES models that are less prone to the effects of biased ratings, that is, models that are able to perform highly objective and impartial judgements. Thus, we start this section by proposing features that are useful for building AES models. Then, we propose another set of features that are useful for identifying biased ratings based on subjectivity cues. Finally, we propose an approach to remove biased ratings from the training set, thus learning more objective AES models.

3.1 Features for Essay Scoring

As most existing AES systems, our models are built on the basis of predefined features (e.g. number of words, average word length, and number of spelling errors) that are given to a machine learning algorithm. The features used to build our AES models are discussed and evaluated in (Amorim and Veloso, 2017). They may fall into two broad categories:

Domain features: These are simple linguistic features, including the number of first-person pronouns, demonstrative pronouns and verbs. Features also include the number of pronouns and verbs normalized by the number of tokens in the corresponding sentence.

General features: Most of the general features are based on (Attali and Burstein, 2006). However, due to lack of tools for processing the Portuguese language, we implemented the following features, which are sub-divided as follows:

Grammar and style: Features include the number of grammar errors and misspellings. These numbers are also normalized by the number of tokens in the corresponding sentence. In order to evaluate style, we designed features based on the style rules suggested in (Martins, 2000). Features include the number of style errors and the number of style of errors per sentence.

Organization and development: Features include the number of discourse markers from the Portuguese grammar, and the number of discourse markers per sentence. Discourse markers are linguistic units that establish connections between sentences to build coherent and knit discourse.

Lexical complexity: Features include the Portuguese version for the Flesh score (Martins et al., 1996), the average word length (i.e., the number of syllables), the number of tokens in an essay, and the number of different words in an essay.

Prompt-specific vocabulary usage: Features include different distances between prompt and essay (i.e., cosine distance). In this case, both the prompt and the essay are treated as frequency vectors of words.

3.2 Features for Identifying Biased Ratings

We assume a scenario in which essay raters must ground the provided scores with specific comments. We also assume that we can identify biased ratings by detecting comments with biased language. In order to detect biased language, we developed subjectivity lexicons for the Portuguese language. Specifically, a linguist built a list of Portuguese lexicons based on the analysis of expressions that seem to express some subjectivity of the human evaluator. Our subjectivity lexicons are categorized into the following groups:

Argumentation: This lexicon includes markers of argumentative discourse. Argumentative markers include lexical expressions and connectives, such as: "even" (*até*), "by the way" (*aliás*), "as a consequence" (*como consequência*), "or else" (*ou então*), "as if" (*como se*), "rather than" (*em vez de*), "somehow" (*de certa forma*), "despite" (*apesar de*), among others.

Presupposition: This lexicon includes markers that suggest the rater assumes something is true. Some examples of such markers include: "nowadays" (*hoje em dia*), "to keep on doing" (*continuar a*), and factive verbs.

Modalization: This lexicon indicates that the writer exhibits a stance towards its own state-

ment. Some examples of such markers are adverbs, auxiliary verbs, modality clauses, and some type of verbs.

Sentiment: This lexicon also includes markers that indicate a state of mind or a sentiment of the rater while evaluating the essay. Some examples of such markers include: "with regret" (*infelizmente*), "with pleasure" (*felizmente*), and "it is preferable" (*preferencialmente*).

Valuation: This lexicon assigns a value to facts. Usually, adjectives are employed as valuation, but as adjectives are context dependent we use only in this class the markers related to intensification, such as: "absolutely" (*absolutamente*), "highly" (*altamente*), and "approximately" (*aproximadamente*).

3.3 Debiasing the Training Set

Bias is generally defined as a deviation from a norm. If the norm is unknown to us, then bias is hard to identify. Thus, our approach for debiasing the training set starts by finding the norm (in terms of the subjectivity within rater comments) for each score value. Intuitively, the amount of subjectivity within a comment should be similar to the amount of subjectivity within another comment, given that the scores associated with the corresponding essays are close to each other. So, we should not expect to find essays having discrepant scores, but for which the corresponding comments show a similar amount of subjectivity. Our debiasing approach is divided into three steps:

1. Rater comments are represented according to the amount of subjectivity cues. In order to represent a comment, we calculate the distance between it and each of the five subjectivity lexicons. More specifically, we learn word embeddings (Mikolov et al., 2013) for the Portuguese language, and then we employed the Word Mover's Distance function (Kusner et al., 2015) between a comment and the five subjectivity lexicons. As a result, each comment is finally represented by a five-dimensional subjectivity vector, where each dimension corresponds to the amount of a specific type of subjectivity. This results in a subjectivity space, where comments are placed according to their amount of subjectivity.

2. We group subjectivity vectors according to the score misalignment associated with the corresponding essay. Then, we calculate centroids for each group in order to find the prototypical subjectivity vector for each group (or misalignment level).

3. The distance to the prototypical subjectivity vector is used as a measure of deviation from the norm. Specifically, we sort essays according to the distance between the subjectivity vector and the corresponding centroid. Then, we define a number of essays to be removed from the training set. The relative number of essays to be removed from the training set is controlled by hyper-parameter α.

4 Experiments

In this section, we present the data we used to learn and evaluate different AES models. Then, we discuss our evaluation procedure and report the results obtained with our debiasing approach. In particular, our experiments aim to answer the following research questions:

RQ1: How scores are distributed across the essays? How aligned with human raters are different AES models?

RQ2: Does subjectivity in rater comments vary depending on the given score?

RQ3: Does subjectivity in rater comments vary depending on the misalignment between the AES model and the human rater?

RQ4: Can we mitigate the effects of biased ratings?

4.1 Corpus

Our corpus is composed of essays ($n = 1,840$) that were written by high-school students as part of a standardized Brazilian national exam. Each essay is evaluated according to the following five objective aspects:

Formal language: Mastering of the formal Portuguese language.

Relevance to the prompt: Understanding of essay prompt and application of concepts from different knowledge fields, to develop the theme in an argumentative dissertation format.

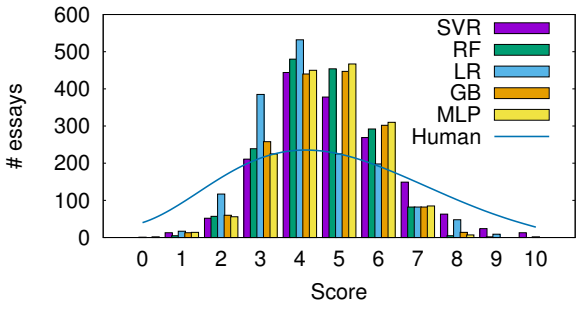


Figure 1: Distribution of the scores given by human raters. Also, distribution of the scores given by different AES models.

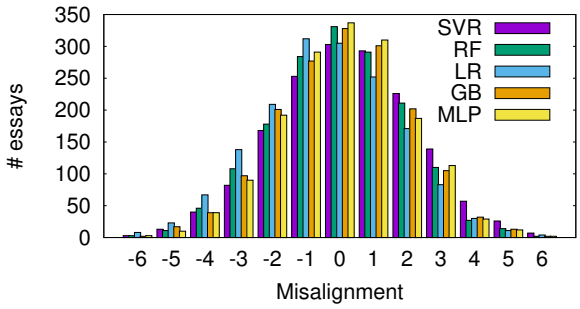


Figure 2: Distribution of misalignment for the different AES models.

Organization of information: Selecting, connecting, organizing, and interpreting information.

Argumentation: Demonstration of knowledge of linguistic mechanisms required to construct arguments.

Solution proposal: Formulation of a proposal to the problem presented.

The final score is given as the sum of the scores associated with each aspect. Raters are supposed to perform impartial and objective evaluations, and they must enter specific comments in order to ground their scores. Also, each essay was assessed by one rater.

Bias-free ratings: We also separate a number of essays ($n = 50$) which received similar scores by three expert raters who were directly instructed to perform impartial, objective, and unbiased evaluations. These raters are PhD-level in Linguistics with unlimited time to provide their ratings, and they do not participate on the creation of the training set. We assume the ratings given to these essays were not contaminated by biased judgements, and we will use these essays for evaluating the efficacy of AES models learned after the training set is debiased.

4.2 Setup

We implemented the different AES models using scikit-learn (Pedregosa et al., 2011). Specifically, we learn AES models using Support Vector Regression (SVR), Random Forests (RF), Logistic Regression (LR), Gradient Boosting (GB), and Multi-Layer Perceptron (MLP). All models are based on the same set of features, previously described in Section 3.1, and all models are trained in regression mode. The measure used to evaluate the effectiveness of the different models is the quadratic weighted kappa (κ) which measures the inter-agreement between human raters and AES models (Cohen, 1960). We conducted five-fold cross validation, where the dataset is arranged into five folds with approximately the same number of examples. At each run, four folds are used as training set, and the remaining fold is used as test set. We also kept a separate validation set. The training set is used to learn the models, the validation set is used to tune hyper-parameters and the test set is used to estimate κ numbers for the different the models. Unless otherwise stated, the results reported are the average of the five runs, and are used to assess the overall effectiveness of each model. To ensure the relevance of the results, we assess the statistical significance of our measurements by comparing each pair of models using a Welch's t-test with $\text{p}-\text{value} \leq 0.01$.

4.3 Results and Discussion

Next we report results obtained from the execution of the experiments, and discuss these results in the light of our research questions.

Score distribution: The first experiment is concerned with RQ1. Figure 1 shows how scores are distributed over the essays in our corpus. Although the distribution differs for each AES model, scores are centered around 4, and few essays received extreme scores. The LR model seems to have a preference for lower scores. The scores provided by the GB and MLP models are better distributed.

Figure 2 shows how aligned with human raters are the different AES models. For most of the essays, AES models are well aligned with human

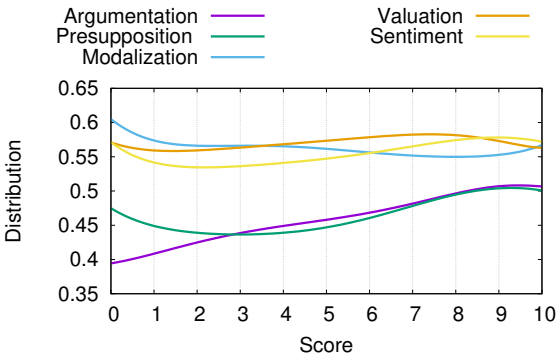


Figure 3: Subjectivity distribution for human raters.

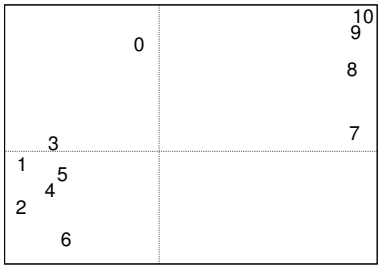


Figure 4: t-SNE representation for subjectivity vectors. Numbers correspond to the scores assigned to corresponding essays.

raters, showing misalignments that vary from -2 to $+2$. For some essays, the LR model tends to give scores that are much smaller than the score given by the human rater. The GB and MLP models perform very similary, but the MLP model shows a slightly better alignment.

Subjectivity vectors and biased ratings: The second experiment is concerned with RQ2. Figure 3 shows the average subjectivity vector grouped according to the score given to the corresponding essay (i.e., the centroid or prototypical vector of a score). More specifically, we first grouped subjectivity vectors according to the score associated with the corresponding essay, and then we calculated the average subjectivity vector for each group. As shown in Figure 3, the argumentation dimension increases with the score, while modalization tends to decrease. Presupposition, valuation and sentiment dimensions show a very similar trend with varying score values.

Figure 4 shows t-SNE representations (van ter Maaten and Hinton, 2008) for the average subjectivity vectors (centroids for each group of score). Three larger clusters emerged: subjectivity vectors associated with score 0, subjectivity vectors associated with scores between 1 and 6, and subjectivity vectors associated with scores between 6 and 10.

Subjectivity vectors and misalignment: The third experiment is concerned with RQ3. Figure 5 shows the average subjectivity vector considering different levels of misalignment. More specifically, we grouped essays according to the misalignment between the score provided by the AES model and the human rater. Then, we calculated the average subjectivity vector for each group. As we can see, subjectivity affects AES models in different ways. In general, however, subjectivity vectors within groups of essays associated with extreme misalignments are very different from subjectivity vectors associated with mild misalignments.

Figure 6 shows t-SNE representations for subjectivity vectors grouped by misalignment levels. Each cluster contains $\approx 80\%$ of the vectors associated with one of the misalignment levels inside the cluster. That is, 20% of the essays will be removed from the training set (i.e., $\alpha = 0.2$).

Debiasing the training set: The last experiment is concerned with RQ4. As described in Section 3.3, our debiasing approach works by removing from the training set a number of essays (controlled by α) that are more likely to be associated with biased ratings. Table 1 shows κ numbers for different α values. Clearly, the inter-agreement decreases as we remove essays with potentially biased ratings from the training set. This happens because the test set remains with essays that are potentially associated with biased ratings. In this case, removing biased ratings from the training set is always detrimental to the efficacy of AES models.

In order to properly evaluate our debiasing approach, we employ the 50 separate essays with bias-free ratings as our test set. In this case, biased ratings are removed from the training set, and the test set is composed by unbiased ratings. Table 2 shows κ numbers for different α values. As expected, the inter-agreement increases significantly with α, until a point in which keeping removing essays from the training set becomes detrimental. This happens either because we start to remove unbiased ratings, or the training set becomes too small. In all cases, the MLP model showed to be

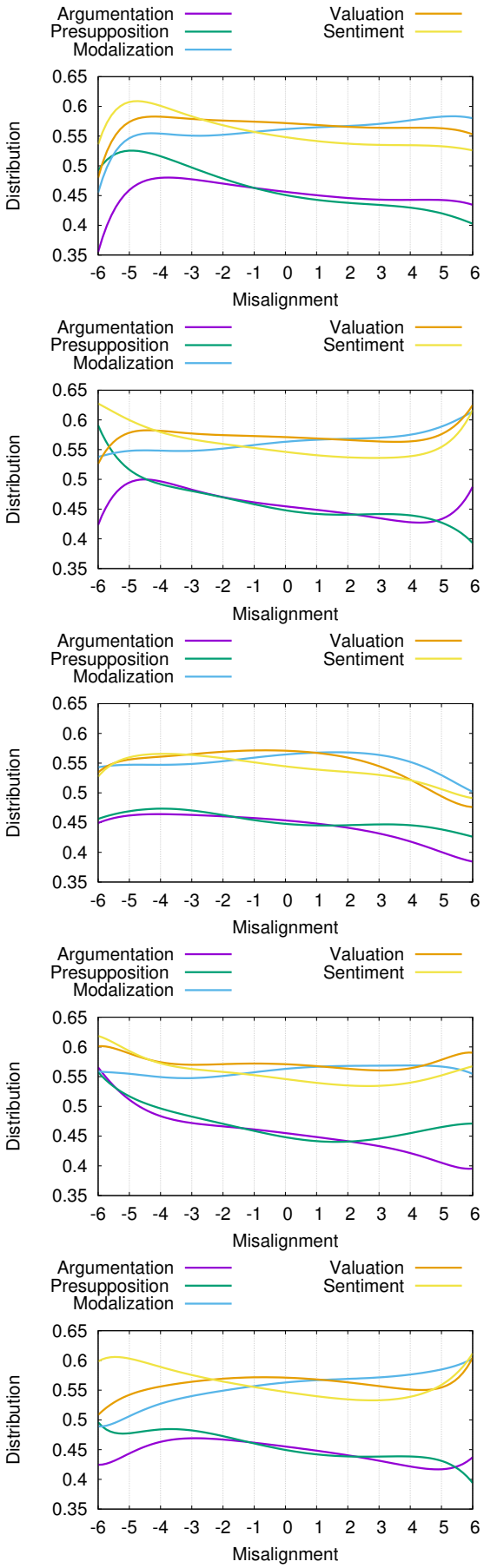


Figure 5: Subjectivity distribution. (Top to bottom) SVR, RF, LR, GB, and MLP.

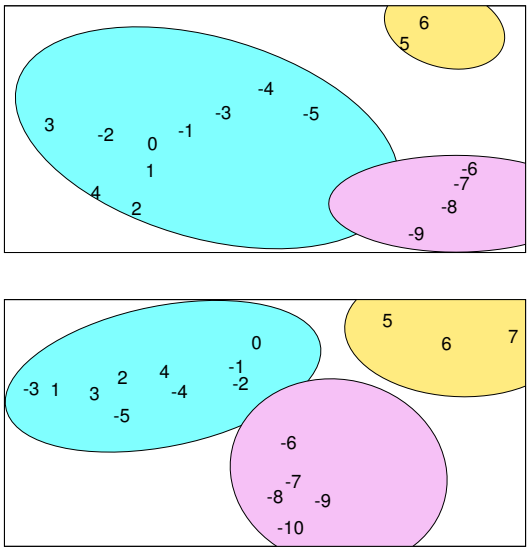


Figure 6: t-SNE representation for subjectivity vectors grouped by misalignment levels. The corresponding regions comprise essays associated with specific misalignment levels. (Top) GB model. (Bottom) MLP model.

	κ				
α	SVR	RF	LR	GB	MLP
–	.404	.410	.408	.432	**.446**
0.1	.390	.339	.364	.378	**.393**
0.2	.365	.331	.344	.370	**.393**
0.3	.345	.326	.338	.365	**.386**
0.4	.340	.324	.333	.361	**.384**
0.5	.307	.317	.328	.358	**.382**

Table 1: κ numbers for different models with varying α values. There are potentially biased ratings in the test set.

	κ				
α	SVR	RF	LR	GB	MLP
–	.451	.472	.466	.491	**.521**
0.1	.467	.491	.481	.505	**.544**
0.2	.481	.511	.490	.521	**.562**
0.3	.488	.526	.497	.542	**.571**
0.4	.491	.523	.499	.547	**.569**
0.5	.481	.518	.494	.545	**.560**

Table 2: κ numbers for different models with varying α values. Ratings in the the test set are likely to be unbiased.

statistically superior than the other models.

5 Conclusions

In this paper, we investigated the problem of automated essay scoring in the presence of biased ratings. Most of the existing work on automated essay scoring is devoted to maximize the agreement with the human rater. This is fairly dangerous, since human ratings may be biased. Overall, discussion about the quality of the ratings in automated essay scoring is lacking, and this was a central interest in this paper. Specifically, we create a subjectivity space from which potentially biased scores/ratings can be identified. We showed that removing biased scores from the training set results in improved AES models. Finally, the essay data as well as the subjectivity lexicons that we will release as part of this research could prove useful in other bias related tasks.

Acknowledgments

This work was partially funded by projects InWeb (grant MCT/CNPq 573871/2008-6) and MASWeb (grant FAPEMIG/PRONEX APQ-01400-14), and by the authors individual grants from CNPq and FAPEMIG. AV thanks the support received from Kunumi.

References

Dimitrios Alikaniotis, Helen Yannakoudakis, and Marek Rei. 2016. Automatic text scoring using neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. pages 715–725.

Laura Allen, Scott Crossley, and Danielle McNamara. 2015. Predicting misalignment between teachers' and students' essay scores using natural language processing tools. In *Proceedings of the 17th International Conference on Artificial Intelligence in Education*. pages 529–532.

Evelin Amorim and Adriano Veloso. 2017. A multi-aspect analysis of automatic essay scoring for brazilian portuguese. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (Student Research Workshop)*. pages 94–102.

Dan Ariely. 2008. *Predictably Irrational: The Hidden Forces That Shape Our Decisions*. HarperCollins.

Yigal Attali and Jill Burstein. 2006. Automated essay scoring with e-rater® v. 2. *The Journal of Technology, Learning and Assessment* 4(3).

Jonathan Baron. 2007. *Thinking and Deciding*, volume 4. Cambridge University Press.

Eyal Beigman and Beata Beigman Klebanov. 2009. Learning with annotation noise. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics*. pages 280–287.

Hongbo Chen and Ben He. 2013. Automated essay scoring by maximizing human-machine agreement. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. pages 1741–1752.

Yen-Yu Chen, Chien-Liang Liu, Chia-Hoang Lee, and Tao-Hsing Chang. 2010. An unsupervised automated essay scoring system. *IEEE Intelligent Systems* 25(5):61–67.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement* 20(1):37–46.

Scott Crossley, Laura Allen, Erica Snow, and Danielle McNamara. 2015. Pssst... textual features... there is more to automatic essay scoring than just you! In *Proceedings of the Fifth International Conference on Learning Analytics and Knowledge*. pages 203–207.

Ronan Cummins, Meng Zhang, and Ted Briscoe. 2016. Constrained multi-task learning for automated essay scoring. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. pages 789–799.

Alexandre Davis and Adriano Veloso. 2016. Subject-related message filtering in social media through context-enriched language models. *Trans. Computational Collective Intelligence* 21:97–138.

Fei Dong and Yue Zhang. 2016. Automatic features for essay scoring - an empirical study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 1072–1077.

Fei Dong, Yue Zhang, and Jie Yang. 2017. Attention-based recurrent convolutional neural network for automatic essay scoring. In *Proceedings of the 21st Conference on Computational Natural Language Learning*. pages 153–162.

Noura Farra, Swapna Somasundaran, and Jill Burstein. 2015. Scoring persuasive essays using opinions and their targets. In *Proceedings of the 10th Workshop on Innovative Use of NLP for Building Educational Applications*.

Pedro Henrique Calais Guerra, Adriano Veloso, Wagner Meira Jr., and Virgílio A. F. Almeida. 2011. From bias to opinion: a transfer-learning approach to real-time sentiment analysis. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. pages 150–158.

Daniel Kaheman and Amos Tversky. 1972. Subjective probability: A judgment of representativeness. *Cognitive Psychology* 3(3):430–454.

Beata Klebanov, Jill Burstein, Nitin Madnani, Adam Faulkner, and Joel Tetreault. 2012. Building subjectivity lexicon(s) from scratch for essay data. In *Proceedings of the 13th International Conference on Computational Linguistics and Intelligent Text Processing*. pages 591–602.

Beata Beigman Klebanov and Eyal Beigman. 2009. From annotator agreement to noise models. *Computational Linguistics* 35(4):495–503.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *Proceedings of the 32nd International Conference on Machine Learning*. pages 957–966.

E. Martins. 2000. *Manual de redação e estilo*. O Estado de São Paulo. https://books.google.com.br/books?id=CAkLnwEACAAJ.

Teresa BF Martins, Claudete M Ghiraldelo, Maria das Graças Volpe Nunes, and Osvaldo Novais de Oliveira Junior. 1996. *Readability formulas applied to textbooks in brazilian portuguese*. Icmsc-Usp.

Danielle McNamaraa, Scott Crossley, Rod Roscoe, Laura Allen, and Jianmin Dai. 2015. A hierarchical classification approach to automated essay scoring. *Assessing Writing* 23:35–59.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 27th Annual Conference on Neural Information Processing Systems*. pages 3111–3119.

C. Myford and E. Wolfe. 2003. Detecting and measuring rater effects using many-facet rasch measurement. *Journal Appl Meas.* 4(4):386–422.

Ellis Page. 1967. Grading essays by computer: progress report. In *Proceedings of the Invitational Conference on Testing Problems*. pages 87–100.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research* 12:2825–2830.

Peter Phandi, Kian Ming Adam Chai, and Hwee Tou Ng. 2015. Flexible domain adaptation for automated essay scoring using correlated linear regression. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 431–439.

Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2013. Linguistic models for analyzing and detecting biased language. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. pages 1650–1659.

Brian Riordan, Andrea Horbach, Aoife Cahill, Torsten Zesch, and Chong Min Lee. 2017. Investigating neural architectures for short answer scoring. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*. pages 159–168.

Swapna Somasundaran, Jill Burstein, and Martin Chodorow. 2014. Lexical chaining for measuring discourse coherence quality in test-taker essays. In *Proceedings of the 25th International Conference on Computational Linguistics*. pages 950–961.

Swapna Somasundaran, Brian Riordan, Binod Gyawali, and Su-Youn Yoon. 2016. Evaluating argumentative and narrative essays using graphs. In *Proceedings of the 26th International Conference on Computational Linguistics*. pages 1568–1578.

Yi Song, Michael Heilman, Beata Klebanov, and Paul Deane. 2014. Applying argumentation schemes for essay scoring. In *Proceedings of the 1st Workshop on Argument Mining*. pages 69–78.

Sargur Srihari, Jim Collins, Rohini Srihari, Harish Srinivasan, Shravya Shetty, and Janina Brutt-Griffler. 2008. Automatic scoring of short handwritten essays in reading comprehension tests. *Artif. Intell.* 172(2-3):300–324.

Sargur Srihari, Rohini Srihari, Pavithra Babu, and Harish Srinivasan. 2007. On the automatic scoring of handwritten essays. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*. pages 2880–2884.

Kaveh Taghipour and Hwee Tou Ng. 2016. A neural approach to automated essay scoring. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 1882–1891.

Amos Tversky and Daniel Kaheman. 1974. Judgement under uncertainty: Heuristics and biases. *Science* 185:1124–1131.

Laurens van ter Maaten and Geoffrey Hinton. 2008. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research* 9:2579–2605.

Content-Based Citation Recommendation

Chandra Bhagavatula
Allen Institute for AI
chandrab@allenai.org

Sergey Feldman
Data Cowboys *
sergey@data-cowboys.com

Russell Power
Independent Researcher †
russell.power@gmail.com

Waleed Ammar
Allen Institute for AI
waleeda@allenai.org

Abstract

We present a content-based method for recommending citations in an academic paper draft. We embed a given query document into a vector space, then use its nearest neighbors as candidates, and rerank the candidates using a discriminative model trained to distinguish between observed and unobserved citations. Unlike previous work, our method does not require metadata such as author names which can be missing, e.g., during the peer review process. Without using metadata, our method outperforms the best reported results on PubMed and DBLP datasets with relative improvements of over 18% in F1@20 and over 22% in MRR. We show empirically that, although adding metadata improves the performance on standard metrics, it favors self-citations which are less useful in a citation recommendation setup. We release an online portal for citation recommendation based on our method,[1] and a new dataset `OpenCorpus` of 7 million research articles to facilitate future research on this task.

1 Introduction

Due to the rapid growth of the scientific literature, conducting a comprehensive literature review has become challenging, despite major advances in digital libraries and information retrieval systems. Citation recommendation can help improve the quality and efficiency of this process by suggesting published scientific documents as likely citations for a query document, e.g., a paper draft to be submitted for ACL 2018. Existing citation recommendation systems rely on various information of the query documents such as author names and publication venue (Ren et al., 2014; Yu et al., 2012), or a partial list of citations provided by the author (McNee et al., 2002; Liu et al., 2015; Jia and Saule, 2017) which may not be available, e.g., during the peer review process or in the early stage of a research project.

Our method uses a neural model to embed all available documents into a vector space by encoding the textual content of each document. We then select the nearest neighbors of a query document as candidates and rerank the candidates using a second model trained to discriminate between observed and unobserved citations. Unlike previous work, we can embed new documents in the same vector space used to identify candidate citations based on their text content, obviating the need to re-train the models to include new published papers. Further, unlike prior work (Yang et al., 2015; Ren et al., 2014), our model is computationally efficient and scalable during both training and test time.

We assess the feasibility of recommending citations when some metadata for the query document is missing, and find that we are able to outperform the best reported results on two datasets while only using papers' textual content (i.e. its title and abstract). While adding metadata helps further improve the performance of our method on standard metrics, we found that it introduces a bias for self-citation which might not be desirable in a citation recommendation system. See §5 for details of our experimental results.

Our main contributions are:

- a content-based method for citation recommendation which remains robust when metadata are missing for query documents,
- large improvements over state of the art results on two citation recommendation datasets despite omitting the metadata,
- a new dataset of seven million research papers, addressing some of the limitations in

* Work done while on contract with AI2
† Work done while at AI2
[1] http://labs.semanticscholar.org/citeomatic/

Proceedings of NAACL-HLT 2018, pages 238–251
New Orleans, Louisiana, June 1 - 6, 2018.

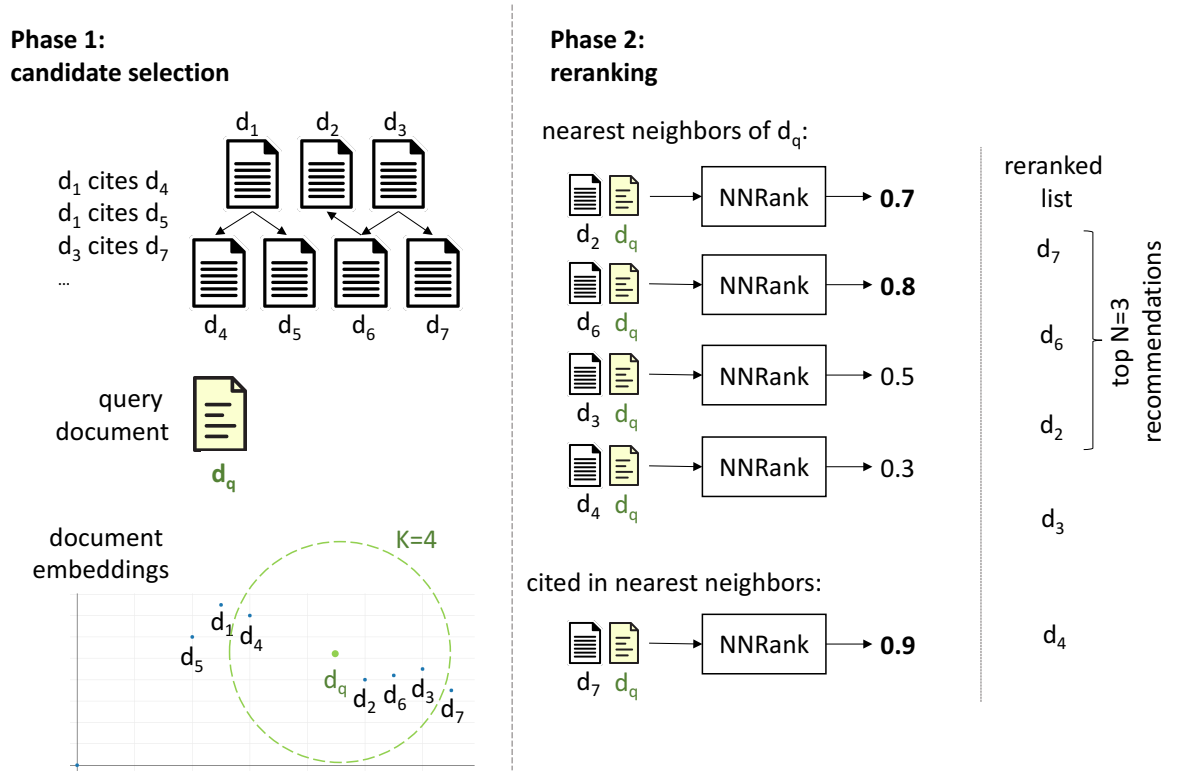


Figure 1: An overview of our Citation Recommendation system. In Phase 1 (`NNSelect`), we project all documents in the corpus (7 in this toy example) in addition to the query document d_q into a vector space, and use its (K=4) nearest neighbors: d_2, d_6, d_3, and d_4 as candidates. We also add d_7 as a candidate because it was cited in d_3. In Phase 2 (`NNRank`), we score each pair $(d_q, d_2), (d_q, d_6), (d_q, d_3), (d_q, d_4)$, and (d_q, d_7) separately to rerank the candidates and return the top 3 candidates: d_7, d_6 and d_2.

previous datasets used for citation recommendation, and

- a scalable web-based literature review tool based on this work.[2]

2 Overview

We formulate citation recommendation as a ranking problem. Given a query document d_q and a large corpus of published documents, the task is to rank documents which should be referenced in d_q higher than other documents. Following previous work on citation recommendation, we use standard metrics (precision, recall, F-measure and mean reciprocal rank) to evaluate our predictions against gold references provided by the authors of query documents.

Since the number of published documents in the corpus can be large, it is computationally expensive to score each document as a candidate reference with respect to d_q. Instead, we recommend citations in two phases: (i) a fast, recall-oriented candidate selection phase, and (ii) a feature rich, precision-oriented reranking phase. Figure 1 provides an overview of the two phases using a toy example.

Phase 1 - Candidate Selection: In this phase, our goal is to identify a set of candidate references for d_q for further analysis without explicitly iterating over all documents in the corpus.[3] Using a trained neural network, we first project all published documents into a vector space such that a document tends to be close to its references. Since the projection of a document is independent of the query document, the entire corpus needs to be embedded only once and can be reused for subsequent queries. Then, we project each query document d_q to the same vector space and identify its nearest neighbors as candidate references. See §3 for more details about candidate selection.

[2] https://github.com/allenai/citeomatic

[3] In order to increase the chances that all references are present in the list of candidates, the number of candidates must be significantly larger than the total number of citations of a document, but also significantly smaller than the number of documents in the corpus.

Phase 2 - Reranking: Phase 1 yields a manageable number of candidates making it feasible to score each candidate d_i by feeding the pair (d_q, d_i) into another neural network trained to discriminate between observed and unobserved citation pairs. The candidate documents are sorted by their estimated probability of being cited in d_q, and top candidates are returned as recommended citations. See §4 for more details about the reranking model and inference in the candidate selection phase.

3 Phase 1: Candidate Selection (`NNSelect`)

In this phase, we select a pool of candidate citations for a given query document to be reranked in the next phase. First, we compute a dense embedding of the query document d_q using the document embedding model (described next), and select K nearest neighbor documents in the vector space as candidates.[4] Following Strohman et al. (2007), we also include the outgoing citations of the K nearest neighbors as candidates.

The output of this phase is a list of candidate documents d_i and their corresponding scores $\texttt{NNSelect}(d_q, d_i)$, defined as the cosine similarity between d_q and d_i in the document embedding space.

Document embedding model. We use a supervised neural model to project any document d to a dense embedding based on its textual content. We use a bag-of-word representation of each textual field, e.g., $d[\text{title}] = \{\text{'content-based'}, \text{'citation'}, \text{'recommendation'}\}$, and compute the feature vector:

$$\mathbf{f}_{d[\text{title}]} = \sum_{t \in d[\text{title}]} w_t^{\text{mag}} \frac{\mathbf{w}_t^{\text{dir}}}{\|\mathbf{w}_t^{\text{dir}}\|_2}, \quad (1)$$

where $\mathbf{w}_t^{\text{dir}}$ is a dense direction embedding and w_t^{mag} is a scalar magnitude for word type t.[5] We then normalize the representation of each field and compute a weighted average of fields to get the document embedding, $\mathbf{e}_d$. In our experiments, we use the title and abstract fields of a document d:

$$\mathbf{e}_d = \lambda^{\text{title}} \frac{\mathbf{f}_{d[\text{title}]}}{\|\mathbf{f}_{d[\text{title}]}\|_2} + \lambda^{\text{abstract}} \frac{\mathbf{f}_{d[\text{abstract}]}}{\|\mathbf{f}_{d[\text{abstract}]}\|_2},$$

where λ^{title} and $\lambda^{\text{abstract}}$ are scalar model parameters.

Training. We learn the parameters of the document embedding model (i.e., $\lambda^*, w_*^{\text{mag}}, \mathbf{w}_*^{\text{dir}}$) using a training set $\mathcal{T}$ of triplets $\langle d_q, d^+, d^- \rangle$ where d_q is a query document, d^+ is a document cited in d_q, and d^- is a document not cited in d_q. The model is trained to predict a high cosine similarity for the pair (d_q, d^+) and a low cosine similarity for the pair (d_q, d^-) using the per-instance triplet loss (Wang et al., 2014):

$$\text{loss} = \max\big(\alpha + s(d_q, d^-) - s(d_q, d^+), 0\big), \quad (2)$$

where $s(d_i, d_j)$ is defined as the cosine similarity between document embeddings $\text{cos-sim}(\mathbf{e}_{d_i}, \mathbf{e}_{d_j})$. We tune the margin α as a hyperparameter of the model (see Appendix B for more details). Next, we describe how negative examples are selected.

Selecting negative examples. Defining positive examples is straight-forward; we use any (d_q, d^+) pair where a document d_q in the training set cites d^+. However, a careful choice of negative training examples is critical for model performance. We use three types of negative examples:

1. **Random:** any document not cited by d_q.
2. **Negative nearest neighbors:** documents that are close to d_q in the embedding space, but are not cited in it.[6]
3. **Citation-of-citation:** documents referenced in positive citations of d_q, but are not cited directly in d_q.

Negative examples belong to at least one of these types that serve different, and complementary purposes. Selecting a paper from the corpus at random as a negative example typically results in easy negative examples. Selecting nearest neighbor documents in the embedding space used for candidate selection enables the re-ranking phase (described in §4) to fix some of the mistakes made in the candidate selection step. Finally, using citations-of-citations as negative examples is based on the assumption that the authors would have included them as positive examples if they were relevant for the query paper. In Appendix §A, we describe the number of negative examples of each type used for training. Next, we describe how to rerank the candidate documents.

[4] We tune K as a hyperparameter of our method.

[5] The magnitude-direction representation is based on Salimans and Kingma (2016) and was found to improve results in preliminary experiments, compared to the standard "direction-only" word representation.

[6] Since the set of approximate neighbors depend on model parameters, we recompute a map from each query document to its K nearest neighbors before each epoch while training the document embedding model.

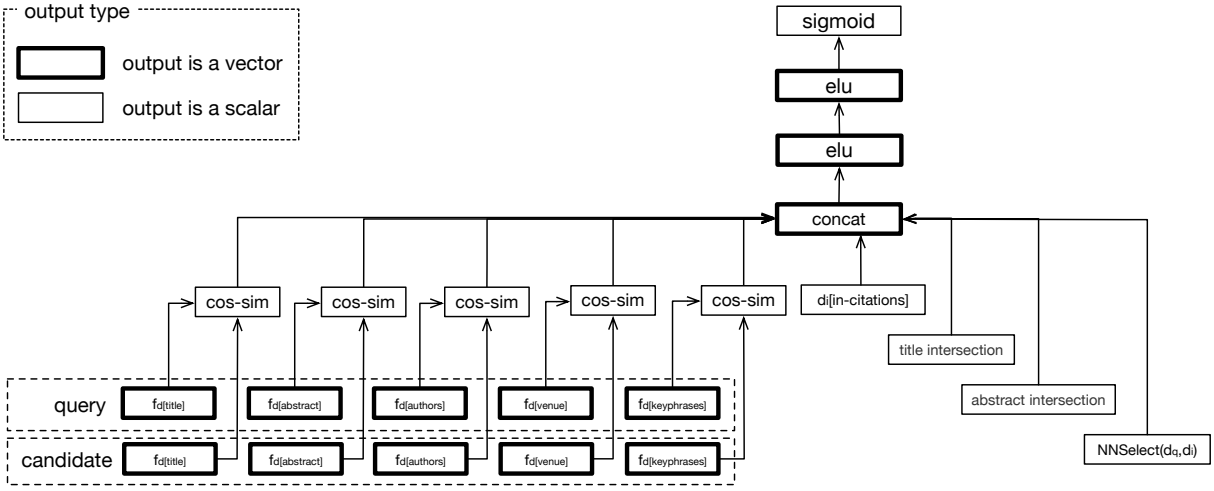


Figure 2: NNRank architecture. For each of the textual and categorical fields, we compute the cosine similarity between the embedding for d_q and the corresponding embedding for d_i. Then, we concatenate the cosine similarity scores, the numeric features and the summed weights of the intersection words, followed by two dense layers with ELU non-linearities The output layer is a dense layer with sigmoid non-linearity, which estimates the probability that d_q cites d_i.

4 Phase 2: Reranking Candidates (NNRank)

In this phase, we train another model which takes as input a pair of documents (d_q, d_i) and estimates the probability that d_i should be cited in d_q.

Input features. A key point of this work is to assess the feasibility of recommending citations without using metadata, but we describe all features here for completeness and defer this discussion to §5. For each document, we compute dense feature vectors $\mathbf{f}_{d[\text{field}]}$ as defined in Eq. 1 for the following fields: title, abstract, authors, venue and keyphrases (if available). For the title and abstract, we identify the subset of word types which appear in both documents (intersection), and compute the sum of their scalar weights as an additional feature, e.g., $\sum_{t \in \cap \text{title}} w_t^{\cap}$. We also use *log* number of times the candidate document d_i has been cited in the corpus, i.e., $log(d_i[\text{in-citations}])$. Finally, we use the cosine similarity between d_q and d_i in the embedding space, i.e., cos-sim$(\mathbf{e}_{d_q}, \mathbf{e}_{d_i})$.

Model architecture. We illustrate the NNRank model architecture in Figure 2. The output layer is defined as:

$$s(d_i, d_j) = \text{FeedForward}(\mathbf{h}), \quad (3)$$

$$\mathbf{h} = \Big[\mathbf{g}_{\text{title}}; \mathbf{g}_{\text{abstract}}; \mathbf{g}_{\text{authors}}; \mathbf{g}_{\text{venue}}; \mathbf{g}_{\text{keyphrases}}; \text{cos-sim}(\mathbf{e}_{d_q}, \mathbf{e}_{d_i}); \textstyle\sum_{t \in \cap_{\text{title}}} w_t^{\cap}; \sum_{t \in \cap_{\text{abstract}}} w_t^{\cap}; d_i[\text{in-citations}]\Big],$$

$$\mathbf{g}_{\text{field}} = \text{cos-sim}(\mathbf{f}_{d_q[\text{field}]}, \mathbf{f}_{d_i[\text{field}]}),$$

where 'FeedForward' is a three layer feed-forward neural network with two exponential linear unit layers (Clevert et al., 2015) and one sigmoid layer. ';' indicates concatenation.

Training. The parameters of the NNRank model are $w_*^{\text{mag}}, \mathbf{w}_*^{\text{dir}}, w_*^{\cap}$ and parameters of the three dense layers in 'FeedForward'. We reuse the triplet loss in Eq. 2 to learn these parameters, but redefine the similarity function $s(d_i, d_j)$ as the sigmoid output described in Eq. 3.

At test time, we use this model to recommend candidates d_i with the highest $s(d_q, d_i)$ scores.

5 Experiments

In this section, we describe experimental results of our citation recommendation method and compare it to previous work.

Datasets. We use the DBLP and PubMed datasets (Ren et al., 2014) to compare with previous work on citation recommendation. The DBLP

dataset contains over 50K scientific articles in the computer science domain, with an average of 5 citations per article. The PubMed dataset contains over 45K scientific articles in the medical domains, with an average of 17 citations per article. In both datasets, a document is accompanied by its title, abstract, venue (i.e. journal or conference where the document was published), authors, citations (i.e. other documents in the corpus that are referenced in the given document) and keyphrases (i.e. phrases considered important by automated extraction methods). We replicate the experimental setup of Ren et al. (2014) by excluding papers with fewer than 10 citations and using the standard train, dev and test splits.[7]

We also introduce `OpenCorpus`,[8] a new dataset of 7 million scientific articles primarily drawn from the computer science and neuroscience domain. Due to licensing constraints, documents in the corpus do not include the full text of the scientific articles, but include the title, abstract, year, author, venue, keyphrases and citation information. The mutually exclusive training, development, and test splits were selected such that no document in the development or test set has a publication year less than that of any document in the training set. Papers with zero citations were removed from the development and test sets. We describe the key characteristics of `OpenCorpus` in Table 1.

Statistic	Value
# of documents in corpus	6.9 million
# of unique authors	8.3 million
# of unique keyphrases	823,677
# of unique venues	23,672
avg. # of incoming citations	7.4 ($\pm$ 38.1)
avg. # of outgoing citations	8.4 ($\pm$ 14.4)
size of training set [years 1991 to 2014]	5.5 million
size of dev set [years 2014 to 2015]	689,000
size of test set [years 2015 to 2016]	20,000

Table 1: Characteristics of the `OpenCorpus`.

Baselines. We compare our method to two baseline methods for recommending citations: ClusCite and BM25. ClusCite (Ren et al., 2014) clusters nodes in a heterogeneous graph of terms, authors and venues in order to find related documents which should be cited. We use the ClusCite results as reported in Ren et al. (2014), which compared it to several other citation recommendation methods and found that it obtains state of the art results on the PubMed and DBLP datasets. The BM25 results are based on our implementation of the popular ranking function Okapi BM25 used in many information retrieval systems. See Appendix §D for details of our BM25 implementation.

Evaluation. We use Mean Reciprocal Rank (MRR) and F1@20 to report the main results in this section. In Appendix §F, we also report additional metrics (e.g., precision and recall at 20) which have been used in previous work. We compute F1@20 as the harmonic mean of the corpus-level precision and recall at 20 (P@20 and R@20). Following (Ren et al., 2014), precision and recall at 20 are first computed for each query document then averaged over query documents in the test set to compute the corpus-level P@20 and R@20.

Configurations. To find candidates in `NNSelect`, we use the approximate nearest neighbor search algorithm `Annoy`[9], which builds a binary-tree structure that enables searching for nearest neighbors in $O(\log n)$ time. To build this tree, points in a high-dimensional space are split by choosing random hyperplanes. We use 100 trees in our approximate nearest neighbors index, and retrieve documents using the cosine distance metric.

We use the `hyperopt` library[10] to optimize various hyperparameters of our method such as size of hidden layers, regularization strength and learning rate. To ensure reproducibility, we provide a detailed description of the parameters used in both `NNSelect` and `NNRank` models, our hyperparameter optimization method and parameter values chosen in Appendix §A.

Main results. Table 2 reports the F1@20 and MRR results for the two baselines and three variants of our method. Since the `OpenCorpus` dataset is much bigger, we were not able to train the ClusCite baseline for it. Totti et al. (2016) have also found it difficult to scale up ClusCite to larger datasets. Where available, we report the mean $\pm$ standard deviation based on five trials.

The first variant, labeled "`NNSelect`," only uses the candidate selection part of our method (i.e., phase 1) to rank candidates by their cosine

[7] The dataset characteristics reported here are different from those in Table 3 in (Ren et al., 2014) because we report the size of the filtered datasets while they report the size of the datasets before filtering.

[8] http://labs.semanticscholar.org/corpus/

[9] https://github.com/spotify/annoy

[10] https://github.com/hyperopt/hyperopt

Method	DBLP		PubMed		OpenCorpus	
	F1@20	MRR	F1@20	MRR	F1@20	MRR
BM25	0.119	0.425	0.209	0.574	0.058	0.218
ClusCite	0.237	0.548	0.274	0.578	–	–
NNSelect	0.282±0.002	0.579±0.007	0.309±0.001	0.699±0.001	0.109	0.221
+ NNRank	0.302±0.001	0.672±0.015	0.325±0.001	0.754±0.003	**0.126**	0.330
+ metadata	**0.303**±0.001	**0.689**±0.011	**0.329**±0.001	**0.771**±0.003	0.125	**0.330**

Table 2: F1@20 and MRR results for two baselines and three variants of our method. BM25 results are based on our implementation of this baseline, while ClusCite results are based on the results reported in Ren et al. (2014). "NNSelect" ranks candidates using cosine similarity between the query and candidate documents in the embedding space (phase 1). "NNSelect + NNRank" uses the discriminative reranking model to rerank candidates (phase 2), without encoding any of the metadata features. "+ metadata" encodes the metadata features (i.e., keyphrases, venues and authors), achieving the best results on all datasets. Mean and standard deviations are reported based on five trials.

similarity to the query document in the embedding space as illustrated in Fig. 1. Although the document embedding space was designed to efficiently select candidates for further processing in phase 2, recommending citations directly based on the cosine distance in this space outperforms both baselines.

The second variant, labeled "NNSelect + NNRank," uses the discriminative model (i.e., phase 2) to rerank candidates selected by NNSelect, without encoding metadata (venues, authors, keyphrases). Both the first and second variants show that improved modeling of paper text can significantly outperform previous methods for citation recommendation, without using metadata.

The third variant, labeled "NNSelect + NNRank + metadata," further encodes the metadata features in the reranking model, and gives the best overall results. On both the DBLP and PubMed datasets, we obtain relative improvements over 20% (for F1@20) and 25% (for MRR) compared to the best reported results of ClusCite.

In the rest of this section, we describe controlled experiments aimed at analyzing different aspects of our proposed method.

Choice of negative samples. As discussed in §3, we use different types of negative samples to train our models. We experimented with using only a subset of the types, while controlling for the total number of negative samples used, and found that using negative nearest neighbors while training the models is particularly important for the method to work. As illustrated in Table 3, on the PubMed dataset, adding negative nearest neighbors while training the models improves the F1@20 score from 0.306 to 0.329, and improves the MRR score from 0.705 to 0.771. Intuitively, using nearest neighbor negative examples focuses training on the harder cases on which the model is more likely to make mistakes.

	F1@20	Δ	MRR	Δ
Full model	0.329		0.771	
without intersection	0.296	**0.033**	0.653	**0.118**
without -ve NNs	0.306	0.016	0.705	0.066
without numerical	0.314	0.008	0.735	0.036

Table 3: Comparison of PubMed results of the full model with model without (i) intersection features, (ii) negative nearest neighbors in training samples, and (iii) numerical features.

Valuable features. We experimented with different subsets of the optional features used in NNRank in order to evaluate the contribution of various features. We found intersection features, NNSelect scores, and the number of incoming citations to be the most valuable feature. As illustrated in Table 3, the intersection features improves the F1@20 score from 0.296 to 0.329, and the MRR score from 0.653 to 0.771, on the PubMed dataset. The numerical features (NNSelect score and incoming citations) improve the F1@20 score from 0.314 to 0.329, and improves the MRR score from 0.735 to 0.771. This shows that, in some applications, feeding engineered features to neural networks can be an effective strategy to improve their performance.

Performance across venues We studied the variability of performance of our model for papers from different venues. Figure 3 shows the F1@20 score of NNRank for papers belonging to the top

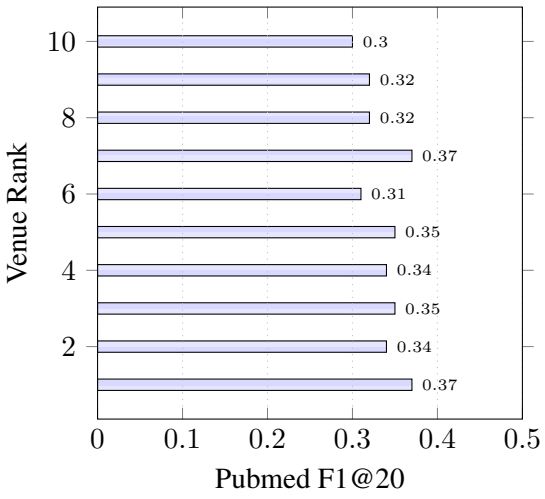


Figure 3: F1@20 of NNRank on the Pubmed dataset across top ten venues

ten venues (by their paper count) in the Pubmed corpus. NNRank's performance is robust across venues.

Encoding textual features. We also experimented with using recurrent and convolutional neural network to encode the textual fields of query and candidate documents, instead of using a weighted sum as described in Eq. 1. We found that recurrent and convolutional encoders are much slower, and did not observe a significant improvement in the overall performance as measured by the F1@20 and MRR metrics. This result is consistent with previous studies on other tasks, e.g., Iyyer et al. (2015).

Number of nearest neighbors. As discussed in §3, the candidate selection step is crucial for the scalability of our method because it reduces the number of computationally expensive pairwise comparisons with the query document at runtime. We did a controlled experiment on the OpenCorpus dataset (largest among the three datasets) to measure the effect of using different numbers of nearest neighbors, and found that both P@20 and R@20 metrics are maximized when NNSelect fetches five nearest neighbors using the approximate nearest neighbors index (and their out-going citations), as illustrated in Table 4.

Self-citation bias. We hypothesized that a model trained with the metadata (e.g., authors) could be biased towards self-citations and other well-cited authors. To verify this hypothesis, we compared two NNRank models – one with metadata, and one without. We measured the mean and max rank of predictions that had at least one author in common with the query document. This experiment was performed with the OpenCorpus dataset.

# of neighbors	R@20	P@20	Time(ms)
1	0.123	0.079	131
5	**0.142**	**0.080**	144
10	0.138	0.069	200
50	0.081	0.040	362

Table 4: OpenCorpus results for NNSelect step with varying number of nearest neighbors on 1,000 validation documents.

A lower mean rank for NNRank + Metadata indicates that the model trained with metadata tends to favor documents authored by one of the query document's authors. We verified the prevalence of this bias by varying the number of predictions for each model from 1 to 100. Figure 4 shows that the mean and max rank of the model trained with metadata is always lower than those for the model that does not use metadata.

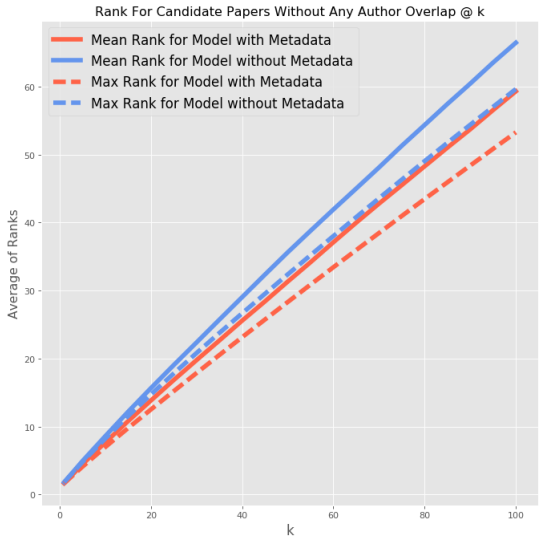


Figure 4: Mean and Max Rank of predictions with varying number of candidates.

6 Related Work

Citation recommendation systems can be divided into two categories – *local* and *global*. A local citation recommendation system takes a few sentences (and an optional placeholder for the candidate citation) as input and recommends citations based on the local context of the input sen-

tences (Huang et al., 2015; He et al., 2010; Tang and Zhang, 2009; Huang et al., 2012; He et al., 2011). A global citation recommendation system takes the entire scholarly article as input and recommends citations for the paper (McNee et al., 2002; Strohman et al., 2007; Nallapati et al., 2008; Kataria et al., 2010; Ren et al., 2014). We address the global citation recommendation problem in this paper.

A key difference of our proposed method compared to previous work is that our method is content-based and works well even in the absence of metadata (e.g. authors, venues, key phrases, seed list of citations). Many citation recommendation systems crucially rely on a query document's metadata. For example, the collaborative filtering based algorithms of McNee et al. (2002); Jia and Saule (2017); Liu et al. (2015) require seed citations for a query document. (Ren et al., 2014; Yu et al., 2012) require authors, venues and key terms of the query documents to infer interest groups and to extract features based on paths in a heterogeneous graph. In contrast, our model performs well solely based on the textual content of the query document.

Some previous work (e.g. (Ren et al., 2014; Yu et al., 2012)) have addressed the citation recommendation problem using graph-based methods. But, training graph-based citation recommendation models has been found to be expensive. For example, the training complexity of the ClusCite algorithm (Ren et al., 2014) is cubic in the number of edges in the graph of authors, venues and terms. This can be prohibitively expensive for datasets as large as `OpenCorpus`. On the other hand our model is a neural network trained via batched stochastic gradient descent that scales very well to large datasets (Bottou, 2010).

Another crucial difference between our approach and some prior work in citation prediction is that we build up a document representation using its constituent words only. Prior algorithms (Huang et al., 2015, 2012; Nallapati et al., 2008; Tanner and Charniak, 2015) learn an explicit representation for each training document separately that isn't a deterministic function of the document's words. This makes the model effectively transductive since a never-before-seen document does not have a ready-made representation. Similarly, Huang et al. (2012)'s method needs a candidate document to have at least one in-coming citation to be eligible for citation – this disadvantages newly published documents. Liu et al. (2015) form document representations using citation relations, which are not available for unfinished or new documents. In contrast, our method does not need to be re-trained as the corpus of potential candidates grows. As long as the new documents are in the same domain as that of the model's training documents, they can simply be added to the corpus and are immediately available as candidates for future queries.

While the citation recommendation task has attracted a lot of research interest, a recent survey paper (Beel et al., 2016) has found three main concerns with existing work: (i) limitations in evaluation due to strongly pruned datasets, (ii) lack of details for re-implementation, and (iii) variations in performance across datasets. For example, the average number of citations per document in the DBLP dataset is 5, but Ren et al. (2014) filtered out documents with fewer than 10 citations from the test set. This drastically reduced the size of the test set. We address these concerns by releasing a new large scale dataset for future citation recommendation systems. In our experiments on the `OpenCorpus` dataset, we only prune documents with zero outgoing citations. We provide extensive details of our system (see Appendix §A) to facilitate reproducibility and release our code[11]. We also show in experiments that our method consistently outperforms previous systems on multiple datasets.

Finally, recent work has combined graph node representations and text-based document representations using CCA (Gupta and Varma, 2017). This sort of approach can enhance our text-based document representations if a technique to create graph node representations at test-time is available.

7 Conclusion

In this paper, we present a content-based citation recommendation method which remains robust when metadata is missing for query documents, enabling researchers to do an effective literature search early in their research cycle or during the peer review process, among other scenarios. We show that our method obtains state of the art results on two citation recommendation datasets, even without the use of metadata available to the

[11] https://github.com/allenai/citeomatic

baseline method. We make our system publicly accessible online. We also introduce a new dataset of seven million scientific articles to facilitate future research on this problem.

Acknowledgements

We would like to thank Oren Etzioni, Luke Zettlemoyer, Doug Downey and Iz Beltagy for participating in discussions and for providing helpful comments on the paper draft; Hsu Han and rest of the Semantic Scholar team at AI2 for creating the OpenCorpus dataset. We also thank Xiang Ren for providing the data used in their experiments on the DBLP and Pubmed datasets. Finally, we thank the anonymous reviewers for insightful comments on the draft.

References

Joeran Beel, Bela Gipp, Stefan Langer, and Corinna Breitinger. 2016. Research-paper recommender systems: a literature survey. *International Journal on Digital Libraries* 17(4):305–338.

Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, Springer, pages 177–186.

Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a siamese time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence* 7(04):669–688.

Cornelia Caragea, Florin Adrian Bulgarov, Andreea Godea, and Sujatha Das Gollapalli. 2014a. Citation-enhanced keyphrase extraction from research papers: A supervised approach. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 1435–1446.

Cornelia Caragea, Florin Adrian Bulgarov, Andreea Godea, and Sujatha Das Gollapalli. 2014b. Citation-enhanced keyphrase extraction from research papers: A supervised approach. In *EMNLP*. volume 14, pages 1435–1446.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289* .

Shashank Gupta and Vasudeva Varma. 2017. Scientific article recommendation by using distributed representations of text and graph. In *WWW*.

Qi He, Daniel Kifer, Jian Pei, Prasenjit Mitra, and C. Lee Giles. 2011. Citation recommendation without author supervision. In *WSDM*.

Qi He, Jian Pei, Daniel Kifer, Prasenjit Mitra, and Lee Giles. 2010. Context-aware citation recommendation. In *Proceedings of the 19th international conference on World wide web*. ACM, pages 421–430.

Wenyi Huang, Saurabh Kataria, Cornelia Caragea, Prasenjit Mitra, C Lee Giles, and Lior Rokach. 2012. Recommending citations: translating papers into references. In *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, pages 1910–1914.

Wenyi Huang, Zhaohui Wu, Liang Chen, Prasenjit Mitra, and C Lee Giles. 2015. A neural probabilistic model for context based citation recommendation. In *AAAI*. pages 2404–2410.

Mohit Iyyer, Varun Manjunatha, Jordan L. Boyd-Graber, and Hal Daumé. 2015. Deep unordered composition rivals syntactic methods for text classification. In *ACL*.

Haofeng Jia and Erik Saule. 2017. An analysis of citation recommender systems: Beyond the obvious. In *ASONAM*.

Saurabh Kataria, Prasenjit Mitra, and Sumit Bhatia. 2010. Utilizing context in generative bayesian models for linked corpus. In *AAAI*.

Haifeng Liu, Xiangjie Kong, Xiaomei Bai, Wei Wang, Teshome Megersa Bekele, and Feng Xia. 2015. Context-based collaborative filtering for citation recommendation. *IEEE Access* 3:1695–1703.

Patrice Lopez and Laurent Romary. 2010. Humb: Automatic key term extraction from scientific articles in grobid. In *Proceedings of the 5th international workshop on semantic evaluation*. Association for Computational Linguistics, pages 248–251.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL*.

Sean M McNee, Istvan Albert, Dan Cosley, Prateep Gopalkrishnan, Shyong K Lam, Al Mamunur Rashid, Joseph A Konstan, and John Riedl. 2002. On the recommending of citations for research papers. In *Proceedings of the 2002 ACM conference on Computer supported cooperative work*. ACM, pages 116–125.

Ramesh Nallapati, Amr Ahmed, Eric P. Xing, and William W. Cohen. 2008. Joint latent topic models for text and citations. In *KDD*.

Xiang Ren, Jialu Liu, Xiao Yu, Urvashi Khandelwal, Quanquan Gu, Lidan Wang, and Jiawei Han. 2014. Cluscite: Effective citation recommendation by information network-based clustering. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 821–830.

Tim Salimans and Diederik P. Kingma. 2016. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *CoRR* abs/1602.07868. http://arxiv.org/abs/1602.07868.

Trevor Strohman, W. Bruce Croft, and David D. Jensen. 2007. Recommending citations for academic papers. In *SIGIR*.

Jie Tang and Jing Zhang. 2009. A discriminative approach to topic-based citation recommendation. In *PAKDD*.

Chris Tanner and Eugene Charniak. 2015. A hybrid generative/discriminative approach to citation prediction. In *HLT-NAACL*.

Luam C. Totti, Prasenjit Mitra, Mourad Ouzzani, and Mohammed J. Zaki. 2016. A query-oriented approach for relevance in citation networks. In *WWW*.

Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. 2014. Learning fine-grained image similarity with deep ranking. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, Washington, DC, USA, CVPR '14, pages 1386–1393. https://doi.org/10.1109/CVPR.2014.180.

Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. 2015. Network representation learning with rich text information. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence, Buenos Aires, Argentina*. pages 2111–2117.

Xiao Yu, Quanquan Gu, Mianwei Zhou, and Jiawei Han. 2012. Citation prediction in heterogeneous bibliographic networks. In *SDM*.

A Hyperparameter Settings

Neural networks are complex and have a large number of hyperparameters to tune. This makes it challenging to reproduce experimental results. Here, we provide details of how the hyperparameters of the `NNSelect` and `NNRank` models were chosen or otherwise set. We chose a subset of hyperparameters for tuning, and left the rest at manually set default values. Due to limited computational resources, we were only able to perform hyperparameter tuning on the development split of the smaller DBLP and Pubmed datasets.

For DBLP and PubMed, we first ran Hyperopt[12] with 75 trials. Each trial was run for five epochs of 500,000 triplets each. The ten top performing of these models were trained for a full 50 epochs, and the best performing model's hyperparameters are selected. Hyperparameters for `NNSelect` were optimized for Recall@20 and those for the `NNRank` model were optimized for F1@20 on the development set. The selected values for DBLP are reported in Table 6 and for PubMed are reported in Table 7.

`OpenCorpus` hyperparameters were set via informal hand-tuning, and the results are in Table 9. A few miscellaneous parameters (not tuned) that are necessary for reproducibility are in Table 8.

We briefly clarify the meaning of some parameters below:

- **Margin Multiplier** - The triplet loss has variable margins for the three types of negatives: $0.1\gamma, 0.2\gamma$, and 0.3γ. We treat γ as a hyperparameter and refer to it as the margin multiplier.
- **Use Siamese Embeddings** - For the majority of our experiments, we use a *Siamese* model (Bromley et al., 1993). That is, the textual embeddings for the query text and abstract share the same weights. However, we had a significantly larger amount of data to train `NNRank` on `OpenCorpus`, and found that non-Siamese embeddings are beneficial.
- **Use Pretrained** - We estimate word embeddings on the titles and abstracts of `OpenCorpus` using Word2Vec implemented by the gensim Python package[13].

[12]https://github.com/hyperopt/hyperopt
[13]https://radimrehurek.com/gensim/

B Margin Loss Details

When computing the margins for the triplet loss, we use a boosting function for highly cited documents. The full triplet loss function is as follows:

$$\max\big(\gamma\alpha(d^-) + s(d_q, d^-) + B(d^-) - s(d_q, d^+) - B(d^+), 0\big)$$

where γ is the margin multiplier, and $\alpha(d^-)$ varies based on the type of negative document:

- $\alpha(d^-) = 0.3$ for random negatives
- $\alpha(d^-) = 0.2$ for nearest neighbor negatives
- $\alpha(d^-) = 0.1$ for citation-of-citation negatives.

The boosting function is defined as follows:

$$B(d) = \frac{\sigma\left(\frac{d[\text{in-citations}]}{100}\right)}{50}$$

where σ is the sigmoid function and $d[\text{in-citations}]$ is the number of times document d was cited in the corpus. The boosting function allows the model to slightly prefer candidates that are cited more frequently, and the constants were set without optimization.

C Nearest Neighbors for Training Details

When obtaining nearest neighbors for negative examples during training, we use a heuristic to find a subset of the fetched nearest neighbors that are sufficiently wrong. That is, these are non-citation samples that look dissimilar in the original text but similar in the embedding space. This procedure is as follows for each training query:

1. Compute the Jaccard similarities between a training query and all of its true citations using the concatenation of title and abstract texts.
2. Compute the bottom fifth percentile Jaccard similarity value. I.e. the value below which only the bottom 5% most least textually similar true citations fall. For example, if the Jaccard similarities range from 0.2 to 0.9, the fifth percentile might plausibly be 0.3.
3. Use the Annoy index computed at the end of the previous epoch to fetch nearest neighbors for the query document.

4. Compute the textual Jaccard similarity between all of the nearest neighbors and the query document.

5. Retain nearest neighbors that have a smaller Jaccard similarity than the fifth percentile. Using the previous example, retain the nearest neighbors that have a lower Jaccard similarity than 0.3.

D BM25 Details

BM25 Implementation	DBLP		PubMed	
	F@20	MRR	F@20	MRR
Ren et al. (2014)	0.111	0.411	0.153	0.497
Our approximation	0.119	0.425	0.209	0.574

Table 5: Results of our BM25 implementation on DBLP and Pubmed datasets.

Okapi-BM25 is a popular ranking function. We use BM25 as an IR-based baseline for the task of citation recommendation. For the DBLP and Pubmed datasets, BM25 performance is provided in Ren et al. (2014). To create a competitive BM25 baseline for `OpenCorpus`, we first created indexes for the DBLP and Pubmed datasets and tuned the query to approximate the performance reported in previous work. We used Whoosh[14] to create an index. We extract the key terms (using Whoosh's `key_terms_from_text` method) from the title and abstract of each query document. The key terms from the document are concatenated to form the query string. Table 5 shows that our BM25 is a close approximation to the BM25 implementation of previous work and can be reliably used as a strong IR baseline for `OpenCorpus`. In Table 2, we report results on all three datasets using our BM25 implementation.

E Key Phrases for `OpenCorpus`

In the `OpenCorpus` dataset, some documents are accompanied by automatically extracted key phrases. Our implementation of automatic key phrase extraction is based on standard key phrase extraction systems – e.g. (Caragea et al., 2014a,b; Lopez and Romary, 2010). We first extract noun phrases using the Stanford CoreNLP package (Manning et al., 2014) as candidate key phrases. Next, we extract corpus level and document level features (e.g. term frequency, document frequency, n-gram probability etc.) for each candidate key phrase. Finally, we rank the candidate key phrases using a ranking model that is trained on author-provided key phrases as gold labels.

F Detailed Experimental Results

Table 10 compares `NNRank` with previous work in detail on DBLP and Pubmed datasets. ClusCite (Ren et al., 2014) clusters nodes in a heterogeneous graph of terms, authors and venues in order to find related documents which should be cited. ClusCite obtains the previous best results on these two datasets. L2-LR (Yu et al., 2012) uses a linear combination of meta-path based linear features to classify candidate citations. We show that `NNRank` (with and without metadata) consistently outperforms ClusCite and other baselines on all metrics on both datasets.

[14] https://pypi.python.org/pypi/Whoosh/

Hyperparameter	NNSelect Range	NNSelect Chosen Value	NNRank Range	NNRank Chosen Value
learning rate	[1e-5, 1e-4, . . . , 1e-1]	0.01	[1e-5, 1e-4, . . . , 1e-1]	0.01
l2 regularization	[0, 1e-7, 1e-6, . . . , 1e-2]	0	[0, 1e-7, 1e-6, . . . , 1e-2]	1e-3
l1 regularization	[0, 1e-7, 1e-6, . . . , 1e-2]	1e-7	[0, 1e-7, 1e-6, . . . , 1e-2]	1e-4
word dropout	[0, 0.05, 0.1, . . . , 0.75]	0.60	[0, 0.05, 0.1, . . . , 0.75]	0.35
margin multiplier	[0.5, 0.75, 1.0, 1.25, 1.5]	1.0	[0.5, 0.75, 1.0, 1.25, 1.5]	0.5
dense dimension	[25, 50, . . . , 325]	300	[25, 50, . . . , 325]	175
metadata dimension	-	-	[5, 10, . . . , 55]	45
use pretrained	[true, false]	true	[true, false]	false
finetune pretrained	[true, false]	true	[true, false]	-
number ANN neighbors	-	10	-	-
triplets per batch size	-	256	-	256
triplets per epoch	-	500000	-	500000
triplets per training	-	2500000	-	2500000
use Siamese embeddings	-	true	-	true

Table 6: DBLP hyperparameter tuning results. Note that the dense dimension when using pretrained vectors is fixed to be 300. A '-' indicates that the variable was not tuned.

Hyperparameter	NNSelect Range	NNSelect Chosen Value	NNRank Range	NNRank Chosen Value
learning rate	[1e-5, 1e-4, . . . , 1e-1]	0.001	[1e-5, 1e-4, . . . , 1e-1]	0.001
l2 regularization	[0, 1e-7, 1e-6, . . . , 1e-2]	0	[0, 1e-7, 1e-6, . . . , 1e-2]	0
l1 regularization	[0, 1e-7, 1e-6, . . . , 1e-2]	1e-6	[0, 1e-7, 1e-6, . . . , 1e-2]	1e-6
word dropout	[0, 0.05, 0.1, . . . , 0.75]	0.55	[0, 0.05, 0.1, . . . , 0.75]	0.1
margin multiplier	[0.5, 0.75, 1.0, 1.25, 1.5]	0.5	[0.5, 0.75, 1.0, 1.25, 1.5]	1.5
dense dimension	[25, 50, . . . , 325]	325	[25, 50, . . . , 325]	150
metadata dimension	-	-	[5, 10, . . . , 55]	40
use pretrained	[true, false]	false	[true, false]	false
finetune pretrained	[true, false]	-	[true, false]	-
number ANN neighbors	-	10	-	-
triplets per batch size	-	256	-	256
triplets per epoch	-	500000	-	500000
triplets per training	-	2500000	-	2500000
use Siamese embeddings	-	true	-	true

Table 7: PubMed hyperparameter tuning results. Note that the dense dimension when using pretrained GloVe vectors is fixed to be 300. A '-' indicates that the variable was not tuned.

Hyperparameter	PubMed/DBLP Value	`OpenCorpus` Value
title/abstract vocabulary size	200000	200000
maximum title length	50	50
maximum abstract length	500	500
training triplets per query	6	6
min # of papers per author included	1	10
min # of papers per venue included	1	10
min # of papers per keyphrases included	5	10
max authors per document	8	8
max keyphrases per document	20	20
minimum true citations per document	2/1	2
maximum true citations per document	100	100
optimizer	LazyAdamOptimizer*	Nadam**
use magnitude-direction embeddings	true	true
reduce learning rate upon plateau	false	true

Table 8: Per-dataset parameters. These were hand-specified. *LazyAdamOptimizer is part of TensorFlow. **Nadam is part of Keras.

Hyperparameter	`NNSelect` Value	`NNRank` Value
learning rate	0.001	0.001
l2 regularization	1e-5	1e-5
l1 regularization	1e-7	1e-7
word dropout	0.1	0.1
margin multiplier	1.0	1.0
dense dimension	75	75
metadata dimension	-	25
use pretrained	false	false
number ANN neighbors	5	-
triplets per batch size	256	32
triplets per epoch	2500000	2500000
triplets per training	25000000	100000000
use Siamese embeddings	true	false

Table 9: Hyperparameters used for `OpenCorpus`

	DBLP					PubMed				
Method	**P@10**	**P@20**	**R@20**	**F1@20**	**MRR**	**P@10**	**P@20**	**R@20**	**F1@20**	**MRR**
BM25	0.126	0.0902	0.1431	0.11	0.4107	0.1847	0.1349	0.1754	0.15	0.4971
L2-LR	0.2274	0.1677	0.2471	0.200	0.4866	0.2527	0.1959	0.2504	0.2200	0.5308
ClusCite	0.2429	0.1958	0.2993	0.237	0.5481	0.3019	0.2434	0.3129	0.274	0.5787
`NNSelect`	0.287	0.230	0.363	0.282	0.579	0.388	0.316	0.302	0.309	0.699
+ `NNRank`	0.339	0.247	0.390	0.302	0.672	0.421	0.332	0.318	0.325	0.754
+ metadata	**0.345**	**0.247**	**0.390**	**0.303**	**0.689**	**0.429**	**0.337**	**0.322**	**0.329**	**0.771**

Table 10: Comparing `NNRank` with ClusCite. (Ren et al., 2014) have presented results on several other topic-based, link-based and network-based citation recommendation methods as baselines. For succinctness, we show results for the best system, Cluscite, and two baselines BM25 and L2-LR.

Looking Beyond the Surface: A Challenge Set for Reading Comprehension over Multiple Sentences

Daniel Khashabi[1], **Snigdha Chaturvedi**[2], **Michael Roth**[3], **Shyam Upadhyay**[1], **Dan Roth**[1]
[1]University of Pennsylvania, [2]University of California, Santa Cruz, [3]Saarland University
{danielkh,shyamupa,danroth}@cis.upenn.edu, snigdha@ucsc.edu, mroth@coli.uni-sb.de

Abstract

We present a reading comprehension challenge in which questions can only be answered by taking into account information from multiple sentences. We solicit and verify questions and answers for this challenge through a 4-step crowdsourcing experiment. Our challenge dataset contains $\sim 6k$ questions for $+800$ paragraphs across 7 different domains (elementary school science, news, travel guides, fiction stories, etc) bringing in linguistic diversity to the texts and to the questions wordings. On a subset of our dataset, we found human solvers to achieve an F1-score of 86.4%. We analyze a range of baselines, including a recent state-of-art reading comprehension system, and demonstrate the difficulty of this challenge, despite a high human performance. The dataset is the first to study multi-sentence inference at scale, with an open-ended set of question types that requires reasoning skills.

1 Introduction

Machine Comprehension of natural language text is a fundamental challenge in AI and it has received significant attention throughout the history of AI (Greene, 1959; McCarthy, 1976; Reiter, 1976; Winograd, 1980). In particular, in natural language processing (NLP) it has been studied under various settings, such as multiple-choice Question-Answering (QA) (Green Jr. et al., 1961), Reading Comprehension (RC) (Hirschman et al., 1999), Recognizing Textual Entailment (RTE) (Dagan et al., 2013) etc. The area has seen rapidly increasing interest, thanks to the existence of sizable datasets and standard benchmarks. CNN/Daily Mail (Hermann et al., 2015), SQuAD (Rajpurkar et al., 2016) and NewsQA (Trischler et al., 2016) to name a few, are some of the datasets that were released recently with the goal of facilitating research in machine comprehension. Despite all the excitement fueled by that large data sets and the ability to directly train statistical learning models, current QA systems do not have capabilities comparable to elementary school or younger children (Clark and Etzioni, 2016). For many of these datasets, researchers point out that models neither need to 'comprehend' in order to correctly predict an answer, nor do they learn to 'reason' in a way that generalizes across datasets. For example, Khashabi et al. (2016) showed that adversarial perturbation in candidate answers results in a significant drop in performance of a few state-of-art science QA systems. Similarly, Jia and Liang (2017) show that adding an adversarially selected sentence to the instances in the SQuAD datasets drastically reduces the performance of many of the existing baselines. Chen et al. (2016) show that in the CNN/Daily Mail datasets, "the required reasoning and inference level ... is quite simple" and that a relatively simple algorithm can get almost close to the upper-bound. We believe that one key reason that simple algorithms can deal with the existing large datasets but, nevertheless, fail at generalization, is that the datasets do not actually require a deep understanding.

We propose to address this shortcoming by developing a reading comprehension challenge in which answering each of the questions requires reasoning over multiple sentences.

There is evidence that answering 'single-sentence questions', i.e. questions that can be answered from a single sentence of the given paragraph, is easier than answering multi-sentence questions', which require multiple sentences to answer a given question. For example, Richardson et al. (2013) released a reading comprehension dataset that contained both single-sentence and multi-sentence questions; models proposed for this task yielded considerably better performance on the single-sentence questions than on the multi-

Proceedings of NAACL-HLT 2018, pages 252–262
New Orleans, Louisiana, June 1 - 6, 2018.

sentence questions (according to Narasimhan and Barzilay (2015) accuracy of about 83% and 60% on these two types of questions, respectively).

There could be multiple reasons for this. First, multi-sentence reasoning seems to be inherently a difficult task. Research has shown that while complete-sentence construction emerges as early as first grade for many children, their ability to integrate sentences emerges only in fourth grade (Berninger et al., 2011). Answering multi-sentence questions might be more challenging for an automated system because it involves more than just processing individual sentences but rather combining linguistic, semantic and background knowledge across sentences—a computational challenges in itself. Despite these challenges, multi-sentence questions can be answered by humans and hence present an interesting yet reasonable goal for AI systems (Davis, 2014).

In this work, we propose a multi-sentence QA challenge in which questions can be answered only using information from multiple sentences. Specifically, we present MultiRC (Multi-Sentence Reading Comprehension)[1]—a dataset of short paragraphs and multi-sentence questions that can be answered from the content of the paragraph. Each question is associated with several choices for answer-options, out of which *one or more* correctly answer the question. Figure 1 shows two examples from our dataset. Each instance consists of a multi-sentence paragraph, a question, and answer-options. All instances were constructed such that it is not possible to answer a question correctly without gathering information from multiple sentences. Due to space constraints, the figure shows only the relevant sentences from the original paragraph. The entire corpus consists of 871 paragraphs and about $\sim$ 6k multi-sentence questions.

The goal of this dataset is to encourage the research community to explore approaches that can do more than sophisticated lexical-level matching. To accomplish this, we designed the dataset with three key challenges in mind. (i) The number of correct answer-options for each question is not pre-specified. This removes the over-reliance of current approaches on answer-options and forces them to decide on the correctness of each candidate answer independently of others. In other words, unlike previous work, the task here is not to simply identify the best answer-option, but to evaluate the correctness of each answer-option individually. For example, the first question in Figure 1 can be answered by combining information from sentences 3, 5, 10, 13 and 15. It requires not only understanding that the stalker's name is Timothy but also that he is the man who Mary had hit. (ii) The correct answer(s) is not required to be a span in the text. For example, the correct answer, A, of the second question in Figure 1 is not present in the paragraph verbatim. It is instead a combination of two spans from 2 sentences: 12 and 13. Such answer-options force models to process and understand not only the paragraph and the question but also the answer-options. (iii) The paragraphs in our dataset have diverse provenance by being extracted from 7 different domains such as news, fiction, historical text etc., and hence are expected to be more diverse in their contents as compared to single-domain datasets. We also expect this to lead to diversity in the types of questions that can be constructed from the passage.

Overall, we introduce a reading comprehension dataset that significantly differs from most other datasets available today in the following ways:

S3: Hearing noises in the garage, Mary Murdock finds a bleeding man, mangled and impaled on her jeep's bumper.
S5: Panicked, she hits him with a golf club.
S10: Later the news reveals the missing man is kindergarten teacher, Timothy Emser.
S12: It transpires that Rick, her boyfriend, gets involved in the cover up and goes to retrieve incriminatory evidence off the corpse, but is killed, replaced in Emser's grave.
S13: It becomes clear Emser survived.
S15: He stalks Mary many ways.

Who is stalking Mary?

A)* Timothy	D) Rick
B) Timothy's girlfriend	E) Murdock
C)* The man she hit	F) Her Boyfriend

S1: Most young mammals, including humans, play.
S2: Play is how they learn the skills that they will need as adults.
S6: Big cats also play.
S8: At the same time, they also practice their hunting skills.
S11: Human children learn by playing as well.
S12: For example, playing games and sports can help them learn to follow rules.
S13: They also learn to work together.

What do human children learn by playing games and sports?
A)* They learn to follow rules and work together
B) hunting skills
C)* skills that they will need as adult

Figure 1: Examples from our MultiRCcorpus. Each example shows relevant excerpts from a paragraph; multi-sentence question that can be answered by combining information from multiple sentences of the paragraph; and corresponding answer-options. The correct answer(s) is indicated by a *. Note that there can be multiple correct answers per question.

[1] http://cogcomp.org/multirc/

- $\sim 6k$ high-quality multiple-choice RC questions that are generated (and manually verified via crowdsourcing) to require integrating information from multiple sentences.
- The questions are not constrained to have a *single* correct answer, generalizing existing paradigms for representing answer-options.
- Our dataset is constructed using 7 different sources, allowing more diversity in content, style, and possible question types.
- We show a significant performance gap between current solvers and human performance, indicating an opportunity for developing sophistical reasoning systems.

2 Relevant Work

Automated reasoning is arguably one of the major problems in contemporary AI research. Brachman et al. (2005) suggest challenges for developing AI program that can pass the SAT exams. In similar spirit Clark and Etzioni (2016) advocate elementary-school tests as a new test for AI. Davis (2014) proposes hand-construction of multiple-choice challenge sets that are easy for children but difficult for computers. Despite Davis' claim on simplicity of his target questions, it is not clear how easy it is to generate such questions, as he doesn't provide any reasonably-sized dataset matching his proposal. Weston et al. (2015) present a relatively small dataset of 10 reasoning categories, and propose to build a system that uses a world model and a linguistic model. The fundamental limitation of the dataset is that it is generated according to a restricted set of reasoning categories, which possibly limits the complexity and diversity of questions.

Some other recent datasets proposed for machine comprehension also pay attention to type of questions and reasoning required. For example, RACE (Lai et al., 2017) attempts to incorporate different types of reasoning phenomena, and MCTest (Richardson et al., 2013) attempted to contain at least 50% multi-sentence reasoning questions. However, since the crowdsourced workers who created the dataset were only encouraged, and not required, to write such questions, it is not clear how many of these questions actually require multi-sentence reasoning (see Sec. 3.5). Similarly, only about 25% of question in the RACE dataset require multi-sentence reasoning as reported in their paper. Remedia (Hirschman et al., 1999) also contains 5 different types of questions (based on question words) but is a much smaller dataset. Other datasets which do not deliberately attempt to include multi-sentence reasoning, like SQuAD (Rajpurkar et al., 2016) and the CNN/Daily Mail dataset (Hermann et al., 2015), suffer from even lower percentage of such questions (12% and 2% respectively (Lai et al., 2017)). There are several other corpora which do not guarantee specific reasoning types, including MS MARCO (Nguyen et al., 2016), WikiQA (Yang et al., 2015), and TriviaQA (Joshi et al., 2017).

The complexity of reasoning required for a reading comprehension dataset would depend on several factors such as the source of questions or paragraphs; the way they are generated; and the order in which they are generated (i.e. questions from paragraphs, or the reverse). Specifically, paragraphs' source could influence the complexity and diversity of the language of the paragraphs and questions, and hence the required level of reasoning capabilities. Unlike most current datasets which rely on only one or two sources for their paragraphs (e.g. CNN/Daily Mail and SQuAD rely only on news and Wikipedia articles respectively) our dataset uses 7 different domains.

Another factor that distinguishes our dataset from previously proposed corpora is the way answers are represented. Several datasets represent answers as multiple-choices with a single correct answer. While multiple-choice questions are easy to grade, coming up with non-trivial correct and incorrect answers can be challenging. Also, assuming exactly one correct answer (e.g., as in MCTest and RACE) inadvertently changes the task from choosing the correct answer to choosing the most likely answer. Other datasets (e.g MS-MARCO and SQuAD) represent answers as a contiguous substring within the passage. This assumption of the answer being a span of the paragraph, limits the questions to those whose answer is contained verbatim in the paragraph. Unfortunately, it rules out more complicated questions whose answers are only implied by the text and hence require a deeper understanding. Because of these limitations, we designed our dataset to use multiple-choice representations, but without specifying the number of correct answers for each question.

3 Construction of MultiRC

In this section we describe our principles and methodology of dataset collection. This includes automatically collecting paragraphs, composing questions and answer-options through crowdsourcing platform, and manually curating the collected data. We also summarize a pilot study that helped us design this process, and end with a summary of statistics of the collected corpus.

3.1 Principles of design

Questions and answers in our dataset are designed based on the following key principles:

Multi-sentenceness. Questions in our challenge require models to use information from multiple sentences of a paragraph. This is ensured through explicit validation. We exclude any question that can be answered based on a single sentence from a paragraph.

Open-endedness. Our dataset is not restricted to questions whose answer can be found verbatim in a paragraph. Instead, we provide a set of handcrafted answer-options for each question. Notably, they can represent information that is not explicitly stated in the text but is only inferable from it (e.g. implied counts, sentiments, and relationships).

Answers to be judged independently. The total number of answer options per question is variable in our data and we explicitly allow multiple correct and incorrect answer options (e.g. 2 correct and 1 incorrect options). As a consequence, correct answers cannot be guessed solely by a process of elimination or by simply choosing the best candidates out of the given options.

Through these principles, we encourage users to explicitly model the semantics of text beyond individual words and sentences, to incorporate extra-linguistic reasoning mechanisms, and to handle answer options independently of one another.

Variability. We encourage variability on different levels. Our dataset is based on paragraphs from multiple domains, leading to linguistically diverse questions and answers. Also, we do not impose any restrictions on the questions, to encourage different forms of reasoning.

3.2 Sources of documents

The paragraphs used in our dataset are extracted from various sources. Here is the complete list of the text types and sources used in our dataset, and the number of paragraphs extracted from each category (indicated in square brackets on the right):

1. News: [121]
   - CNN (Hermann et al., 2015)
   - WSJ (Ide et al., 2008)
   - NYT (Ide et al., 2008)
2. Wikipedia articles [92]
3. Articles on society, law and justice (Ide and Suderman, 2006) [91]
4. Articles on history and anthropology (Ide et al., 2008) [65]
5. Elementary school science textbooks [2] [153]
6. 9/11 reports (Ide and Suderman, 2006) [72]
7. Fiction: [277]
   - Stories from the Gutenberg project
   - Children stories from MCTest (Richardson et al., 2013)
   - Movie plots from CMU Movie Summary corpus (Bamman et al., 2013)

From each of the above-mentioned sources we extracted paragraphs that had enough content. To ensure this we followed a 3-step process. In the first step we selected top few sentences from paragraphs such that they contained 1k-1.5k characters. To ensure coherence, all sentences were contiguous and extracted from the same paragraph. In this process we also discarded paragraphs that seemed to deviate too much from third person narrative style. For example, while processing Gutenberg corpus we considered files that had at least 5k lines because we found that most of them were short poetic texts. In the second step, we annotated (Khashabi et al., 2018b) the paragraphs and automatically filtered texts using conditions such as the average number of words per sentence; number of named entities; number of discourse connectives in the paragraph. These were designed by the authors of this paper after reviewing a small sample of paragraphs. A complete set of conditions is listed in Table 1. Finally in the last step, we manually verified each paragraph and filtered out the ones that had formatting issues or other concerns that seemed to compromise their usability.

[2] https://www.ck12.org

Condition	bound
Number of sentences	≥ 6 & ≤ 18
Number of NER(CoNLL) mentions	≥ 2
Avg. number of NER(CoNLL) mentions	≥ 0.2
Number of NER(Ontonotes) mentions	≥ 4
Avg. number of NER(Ontonotes) mentions	≥ 0.25
Avg. number of words per sentence	≥ 5
Number of coreference mentions	≥ 3
Avg. number of coreference mentions	≥ 0.1
Number of coreference relations	≥ 3
Avg. number of coreference relations	≥ 0.08
Number of coreference chains	≥ 2
Avg. number of coreference chains	≥ 0.1
Number of discourse markers	≥ 2

Table 1: Bounds used to select paragraphs for dataset creation.

3.3 Pipeline of question extraction

In this section, we delineate details of the process for collecting questions and answers. Figure 2 gives a high-level idea of the process. The first two steps deal with creating multi-sentence questions, followed by two steps for construction of candidate answers. Interested readers can find more details on set-ups of each step in Appendix I.

Step 1: Generating questions. The goal of the first step of our pipeline is to collect multi-sentence questions. We show each paragraph to 5 turkers and ask them to write 3-5 questions such that: (1) the question is answerable from the passage, and (2) only those questions are allowed whose answer cannot be determined from a single sentence. We clarify this point by providing example paragraphs and questions. In order to encourage turkers to write meaningful questions that fit our criteria, we additionally ask them for a correct answer and for the sentence indices required to answer the question. To ensure the grammatical quality of the questions collected in this step, we limit the turkers to the countries with English as their major language. After the acquisition of questions in this step, we filter out questions which required less than 2 or more than 4 sentences to be answered; we also run them through an automatic spell-checker[3] and manually correct questions regarding typos and unusual wordings.

Step 2: Verifying multi-sentenceness of questions. In a second step, we verify that each question can only be answered using more than one sentence. For each question collected in the previous step, we create question-sentence pairs by pairing it with each of the sentences necessary for answering it as indicated in the previous step. For a given question-sentence pair, we then ask turkers to annotate if they could answer the question from the sentence it is paired with (binary annotation). The underlying idea of this step is that a multi-sentence question would not be answerable from a single sentence, hence turkers should not be able to give a correct answer for any of the question-sentence pair. Accordingly, we determine a question as requiring multiple sentences only if the correct answer cannot be guessed from any single question-sentence pair. We collected at least 3 annotations per pair, and to avoid sharing of information across sentences, no two pairs shown to a turker came from the same paragraph. We aggregate the above annotations for each question-answer pair and retain only those questions for which no pair was judged as answerable by a majority of turkers.

Step 3: Generating answer-options. In this step, we collect answer-options that will be shown with each question. Specifically, for each verified question from the previous steps, we ask 3 turkers to write as many correct and incorrect answer options as they can think of. In order to not curb creativity, we do not place a restriction on the number of options they have to write. We explicitly ask turkers to design difficult and non-trivial incorrect answer-options (e.g. if the question is about a person, a non-trivial incorrect answer-option would be other people mentioned in the paragraph).

After this step, we perform a light clean up of the candidate answers by manually correcting minor errors (such as typos), completing incomplete sentences and rephrasing any ambiguous sentences. We further make sure there is not much repetition in the answer-options, to prevent potential exploitation of correlation between some candidate answers in order to find the correct answer. For example, we drop obviously duplicate answer-options (i.e. identical options after lower-casing, lemmatization, and removing stop-words).

Step 4: Verifying quality of the dataset. This step serves as the final quality check for both questions and the answer-options generated in the previous steps. We show each paragraph, its questions, and the corresponding answer-options to 3 turkers, and ask them to indicate if they find any errors (grammatical or otherwise), in the questions and/or answer-options. We then manually review,

[3]Grammarly: www.grammarly.com

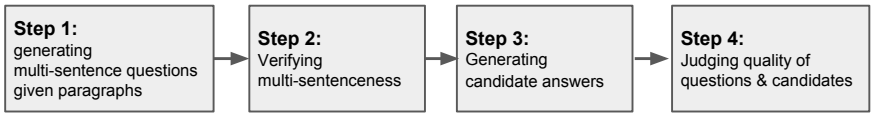


Figure 2: Pipeline of our dataset construction.

and correct if needed, all erroneous questions and answer-options. This ensures that we have meaningful questions and answer-options. In this step, we also want to verify that the correct (or incorrect) options obtained from Step 3 were indeed correct (or incorrect). For this, we additionally ask the annotators to select all correct answer-options for the question. If their annotations did not agree with the ones we had after Step 3 (e.g. if they unanimously selected an 'incorrect' option as the answer), we manually reviewed and corrected (if needed) the annotation.

3.4 Pilot experiments

The 4-step process described above was a result of detailed analysis and substantial refinement after two small pilot studies.

In the first pilot study, we ran a set of 10 paragraphs extracted from the CMU Movie Summary Corpus through our pipeline. Our then pipeline looked considerably different from the one described above. We found the steps that required turkers to write questions and answer-options to often have grammatical errors, possibly because a large majority of turkers were non-native speakers of English. This probslem was more prominent in questions than in answer-options. Because of this, we decided to limit the task to native speakers. Also, based on the results of this pilot, we overhauled the instructions of these steps by including examples of grammatically correct—but undesirable (not multi-sentence)—questions and answer-options, in addition to several minor changes.

Thereafter, we decided to perform a manual validation of the verification steps (current Steps 2 and 4). For this, we (the authors of this paper) performed additional annotations ourselves on the data shown to turkers, and compared our results with those provided by the turkers. We found that in the verification of answer-options, our annotations were in high agreement (98%) with those obtained from mechanical turk. However, that was not the case for the verification of multi-sentence questions. We made several further changes to the first two steps. Among other things, we clarified in the instructions that turkers should not use their background knowledge when writing and verifying questions, and also included negative examples of such questions. Additionally, when turkers judged a question to be answerable using a single sentence, we decided to encourage (but not require) them to guess the answer to the question. This improved our results considerably, possibly because it forced annotators to think more carefully about what the answer might be, and whether they *actually* knew the answer or they just *thought* that they knew it (possibly because of background knowledge or because the sentence contained a lot of information relevant to the question). Guessed answers in this step were only used to verify the validity of multi-sentence questions. They were not used in the dataset or subsequent steps.

After revision, we ran a second pilot study in which we processed a set of 50 paragraphs through our updated pipeline. This second pilot confirmed that our revisions were helpful, but thanks to its larger size, also allowed us to identify a couple of borderline cases for which additional clarifications were required. Based on the results of the second pilot, we made some additional minor changes and then decided to apply the pipeline for creating the final dataset.

3.5 Verifying multi-sentenceness

While collecting our dataset, we found that, even though Step 1 instructed turkers to write multi-sentence questions, not all generated questions indeed required multi-sentence reasoning. This happened even after clarifications and revisions to the corresponding instructions, and we attribute it to honest mistakes. Therefore, we designed the subsequent verification step (Step 2).

There are other datasets which aim to include multi-sentence reasoning questions, especially MCTest. Using our verification step, we systematically verify their multi-sentenceness. For this, we conducted a small pilot study on about 60 multi-sentence questions from MCTest. As for our own verification, we created question-sentence pairs for each question and asked annotators to judge whether they can answer a question from the single sentence shown. Because we did not know

which sentences contain information relevant to a question, we created question-sentence pairs using all sentences from a paragraph. After aggregation of turker annotations, we found that about half of the questions annotated as multi-sentence could be answered from a single sentence of the paragraph. This study, though performed on a subset of the data, underscores the necessity of rigorous verification step for multi-sentence reasoning when studying this phenomenon.

3.6 Statistics on the dataset

We now provide a brief summary of MultiRC. Overall, it contains roughly $\sim 6k$ multi-sentence questions collected for about $+800$ paragraphs.[4] The median number of correct and total answer options for each question is 2 and 5, respectively. Additional statistics are given in Table 2.

In Step 1, we also asked annotators to identify sentences required to answer a given question. We found that answering each question required 2.4 sentences on average. Also, required sentences are often not contiguous, and the average distance between sentences is 2.4. Next, we analyze the types of questions in our dataset. Figure 4 shows the count of first word(s) for our questions. We can see that while the popular question words (*What*, *Who*, etc.) are very common, there is a wide variety in the first word(s) indicating a diversity in question types. About 28% of our questions require binary decisions (true/false or yes/no).

We randomly selected 60 multi-sentence questions from our corpus and asked two independent annotators to label them with the type of reasoning phenomenon required to answer them.[5] During this process, the annotators were shown a list of common reasoning phenomena (shown below), and they had to identify one or more of the phenomena relevant to a given question. The list of phenomena shown to the annotators included the following categories: mathematical and logical reasoning, spatio-temporal reasoning, list/enumeration, coreference resolution (including implicit references, abstract pronouns, event coreference, etc.), causal relations, paraphrases and contrasts (including lexical relations such as synonyms, antonyms), commonsense knowledge, and 'other'. The categories were selected after a manual inspection of a subset of questions by two of the authors. The annotation process revealed that answering questions in our corpus requires a broad variety of reasoning phenomena. The left plot in Figure 3 provides detailed results.

The figure shows that a large fraction of questions require coreference resolution, and a more careful inspection revealed that there were different types of coreference phenomena at play here. To investigate these further, we conducted a follow-up experiment in which manually annotated all questions that required coreference resolution into finer categories. Specifically, each question was shown to two annotators who were asked to select one or more of the following categories: entity coreference (between two entities), event coreference (between two events), set inclusion coreference (one item is part of or included in the other) and 'other'. Figure 3 (right) shows the results of this experiment. We can see that, as expected, entity coreference is the most common type of coreference resolution needed in our corpus. However, a significant number of questions also require other types of coreference resolution. We provide some examples of questions along with the required reasoning phenomena in Appendix II.

Parameter	Value
# of paragraphs	871
# of questions	9,872
# of multi-sentence questions	5,825
avg # of candidates (per question)	5.44
avg # of correct answers (per question)	2.58
avg paragraph length (in sentences)	14.3 (4.1)
avg paragraph length (in tokens)	263.1 (92.4)
avg question length (in tokens)	10.9 (4.8)
avg answer length (in tokens)	4.7 (5.5)
% of yes/no/true/false questions	27.57%
avg # of sent. used for questions	2.37 (0.63)
avg distance between the sent.'s used	2.4 (2.58)
% of correct answers verbatim in paragraph	34.96%
% of incorrect answers verbatim in paragraph	25.84%

Table 2: Various statistics of our dataset. Figures in parentheses represent standard deviation.

4 Analysis

In this section, we provide a quantitative analysis of several baselines for our challenge.

Evaluation Metrics. We define precision and recall for a question q as: $\text{Pre}(q) = \frac{|A(q) \cap \hat{A}(q)|}{|\hat{A}(q)|}$ and $\text{Rec}(q) = \frac{|A(q) \cap \hat{A}(q)|}{|A(q)|}$, where $A(q)$ and $\hat{A}(q)$ are the sets of correct and selected answer-options.

[4] We will also release the $3.7k$ questions that did not pass Step 2. Though not multi-sentence questions, they could be a valuable resource on their own.

[5] The annotations were adjudicated by two authors of this paper.

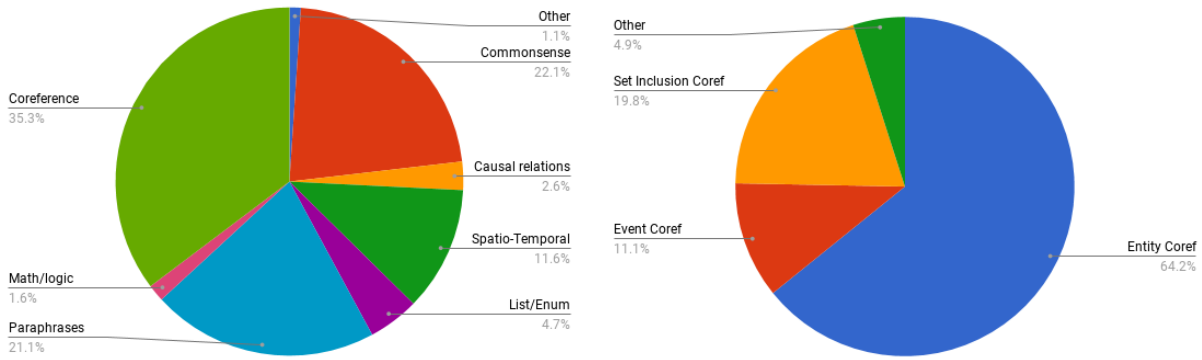


Figure 3: Distribution of (left) general phenomena; (right) variations of the "coreference" phenomena.

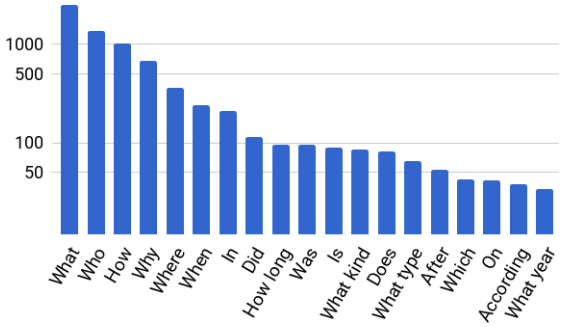


Figure 4: Most frequent first chunks of the questions (counts in log scale).

We define (macro-average) F1_m as the harmonic mean of average-precision $avg_{q \in Q}(\text{Pre}(q))$ and average-recall $avg_{q \in Q}(\text{Rec}(q))$ with Q as the set of all questions.

Since by design, each answer-option can be judged independently, we consider another metric, F1_a, evaluating binary decisions on all the answer-options in the dataset. We define F1_a to be the harmonic mean of $\text{Pre}(Q)$ and $\text{Rec}(Q)$, with $\text{Pre}(Q) = \frac{|A(Q) \cap \hat{A}(Q)|}{|\hat{A}(Q)|}$; $A(Q) = \bigcup_{q \in Q} A(q)$; and similar definitions for $\hat{A}(Q)$ and $\text{Rec}(Q)$.

4.1 Baselines

Human. Human performance provides us with an estimate of the best achievable results on datasets. Using mechanical turk, we ask 4 people (limited to native speakers) to solve our data. We evaluate score of each label by averaging the decision of the individuals.

Random. To get an estimate on the lower-bound we consider a random baseline, where each answer option is selected as correct with a probability of 50% (an unbiased coin toss). The numbers reported for this baseline represent the expected outcome (statistical expectation).

IR (information retrieval baseline). This baseline selects answer-options that best match sentences in a text corpus (Clark et al., 2016). Specifically, for each question q and answer option a_i, the IR solver sends $q + a_i$ as a query to a search engine (we use Lucene) on a corpus, and returns the search engine's score for the top retrieved sentence s, where s must have at least one non-stopword overlap with q, and at least one with a_i.

We create two versions of this system. In the first variation IR(paragraphs) we create a corpus of sentences extracted from all the paragraphs in the dataset. In the second variation, IR(web) in addition to the knowledge of the paragraphs, we use extensive external knowledge extracted from the web (Wikipedia, science textbooks and study guidelines, and other webpages), with 5×10^{10} tokens (280GB of plain text).

SurfaceLR (logistic regression baseline). As a simple baseline that makes use of our small training set, we reimplemented and trained a logistic regression model using word-based overlap features. As described in (Merkhofer et al., 2018), this baseline takes into account the lengths of a text, question and each answer candidate, as well as indicator features regarding the (co-)occurrences of any words in them.

SemanticILP (semi-structured baseline). This state-of-the-art solver, originally proposed for science questions and biology tests, uses a semi-structured representation to formalize the scoring problem as a subgraph optimization problem over multiple layers of semantic abstrac-

	Dev		Test	
	$F1_m$	$F1_a$	$F1_m$	$F1_a$
Random	44.3	43.8	47.1	47.6
IR(paragraphs)	64.3	60.0	54.8	53.9
SurfaceLR	66.1	63.7	66.7	63.5
Human	86.4	83.8	84.3	81.8

Table 3: Performance comparison for different baselines tested on a subset of our dataset (in percentage). There is a significant gap between the human performance and current statistical methods.

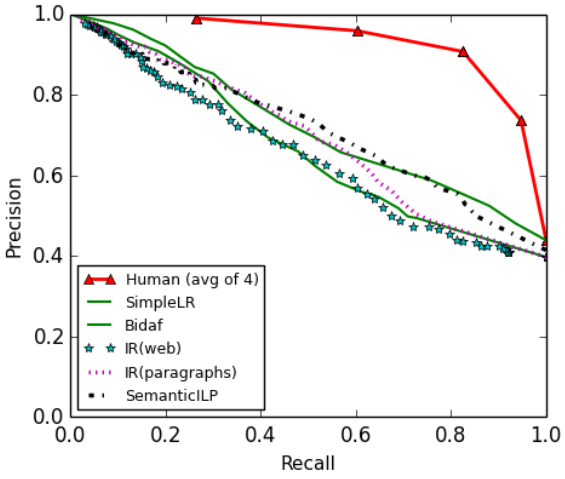


Figure 5: PR curve for each of the baselines. There is a considerable gap with the baselines and human.

tions (Khashabi et al., 2018a). Since the solver is designed for multiple-choice with single-correct answer, we adapt it to our setting by running it for each answer-option. Specifically for each answer-option, we create a single-candidate question, and retrieve a real-valued score from the solver.

BiDAF (neural network baseline). As a neural baseline, we apply this solver by Seo et al. (2017), which was originally proposed for SQuAD but has been shown to generalize well to another domain (Min et al., 2017). Since BiDAF was designed for cloze style questions, we apply it to our multiple-choice setting following the procedure by Kembhavi et al. (2017): Specifically, we score each answer-option by computing the similarity value of it's output span with each of the candidate answers, computed by phrasal similarity tool of Wieting et al. (2015).

4.2 Results

To get a sense of our dataset's hardness, we evaluate both human performance and multiple computational baselines. Each baseline scores an answer-option with a real-valued score, which we threshold to decide whether an answer option is selected or not, where the threshold is tuned on the development set. Table 3 shows performance results for different baselines. The significantly high human performance shows that humans do not have much difficulties in answering the questions. Similar observations can be made in Figure 5 where we plot $avg_{q \in Q}(\text{Pre}(q))$ vs. $avg_{q \in Q}(\text{Rec}(q))$, for different threshold values.

5 Conclusion

In this paper we have presented MultiRC, a reading comprehension dataset in which questions require reasoning over multiple sentences to be answered. Our dataset contains $\sim 6k$ questions extracted from about +800 paragraphs. For each question, it contains multiple answer-options out of which one or more can be correct. The paragraphs (and questions) originate from different domains and hence are amenable to a wide variety and complexity of required reasoning phenomena. We found human performance on this corpus to be about 88% while state-of-the-art machine comprehension models do not exceed a F1-score of 60%. We hope that this significant difference in performance will encourage the community to work towards more sophisticated reasoning systems.

6 Acknowledgement

The authors would like to thank all the contributors to the project. This work was supported by Contract HR0011-15-2-0025 with the US Defense Advanced Research Projects Agency (DARPA). Approved for Public Release, Distribution Unlimited. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. This work was partly funded by grants from the German Research Foundation (DFG EXC 284 and RO 4848/1-1), by the Allen Institute for Artificial Intelligence (allenai.org); by Google; and by IBM-ILLINOIS Center for Cognitive Computing Systems Research (C3SR)—a research collaboration as part of the IBM AI Horizons Network.

References

David Bamman, Brendan O'Connor, and Noah A. Smith. 2013. Learning latent personas of film characters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, Volume 1: Long Papers*. pages 352–361. http://aclweb.org/anthology/P/P13/P13-1035.pdf.

Virginia W. Berninger, William Nagy, and Scott Beers. 2011. Child writers construction and reconstruction of single sentences and construction of multi-sentence texts: Contributions of syntax and transcription to translation. *Reading and writing* 24(2):151–182.

Ronald Brachman, David Gunning, Selmer Bringsjord, Michael Genesereth, Lynette Hirschman, and Lisa Ferro. 2005. Selected grand challenges in cognitive science. Technical report, MITRE Technical Report 05-1218.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, Volume 1: Long Papers*. http://aclweb.org/anthology/P/P16/P16-1223.pdf.

Peter Clark and Oren Etzioni. 2016. My computer is an honor student - but how intelligent is it? standardized tests as a measure of AI. *AI Magazine* 37(1):5–12. http://www.aaai.org/ojs/index.php/aimagazine/article/view/2636.

Peter Clark, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter D. Turney, and Daniel Khashabi. 2016. Combining retrieval, statistics, and inference to answer elementary science questions. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. pages 2580–2586. http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/11963.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzoto. 2013. *Recognizing Textual Entailment: Models and Applications*. Morgan and Claypool.

Ernest Davis. 2014. The limitations of standardized science tests as benchmarks for artificial intelligence research: Position paper. *CoRR* abs/1411.1629. http://arxiv.org/abs/1411.1629.

Bert F. Green Jr., Alice K. Wolf, Carol Chomsky, and Kenneth Laughery. 1961. Baseball: An automatic question-answerer. In *Papers Presented at the May 9-11, 1961, Western Joint IRE-AIEE-ACM Computer Conference*. ACM, IRE-AIEE-ACM '61 (Western), pages 219–224. https://doi.org/10.1145/1460690.1460714.

Peter H. Greene. 1959. An approach to computers that perceive, learn, and reason. In *Papers Presented at the the March 3-5, 1959, Western Joint Computer Conference*. ACM, IRE-AIEE-ACM '59 (Western), pages 181–186. https://doi.org/10.1145/1457838.1457870.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015*. pages 1693–1701.

Lynette Hirschman, Marc Light, Eric Breck, and John D. Burger. 1999. Deep read: A reading comprehension system. In *27th Annual Meeting of the Association for Computational Linguistics, ACL 1999*. http://www.aclweb.org/anthology/P99-1042.

Nancy Ide, Collin F. Baker, Christiane Fellbaum, Charles J. Fillmore, and Rebecca J. Passonneau. 2008. MASC: the manually annotated sub-corpus of american english. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2008*. http://www.lrec-conf.org/proceedings/lrec2008/summaries/617.html.

Nancy Ide and Keith Suderman. 2006. Integrating linguistic resources: The american national corpus model. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation, LREC 2006*. pages 621–624. http://www.lrec-conf.org/proceedings/lrec2006/pdf/560_pdf.pdf.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017*. pages 2021–2031. https://aclanthology.info/papers/D17-1215/d17-1215.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Volume 1: Long Papers*. pages 1601–1611. https://doi.org/10.18653/v1/P17-1147.

Aniruddha Kembhavi, Min Joon Seo, Dustin Schwenk, Jonghyun Choi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Are you smarter than a sixth grader? textbook question answering for multimodal machine comprehension. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. pages 5376–5384. https://doi.org/10.1109/CVPR.2017.571.

Daniel Khashabi, Tushar Khot, Ashish Sabharwal, Peter Clark, Oren Etzioni, and Dan Roth. 2016. Question answering via integer programming over semi-structured knowledge. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016*. pages 1145–1152. http://www.ijcai.org/Abstract/16/166.

Daniel Khashabi, Tushar Khot, Ashish Sabharwal, and Dan Roth. 2018a. Question answering as global reasoning over semantic abstractions. In *Proceedings of The Thirty-Second AAAI Conference on Artificial Intelligence, AAAI 2018 (to appear)*. http://cogcomp.org/page/publication_view/824.

Daniel Khashabi, Mark Sammons, Ben Zhou, Tom Redman, Christos Christodoulopoulos, Vivek Srikumar, Nicholas Rizzolo, Lev Ratinov, Guanheng Luo, Quang Do, Chen-Tse Tsai, Subhro Roy, Stephen Mayhew, Zhilli Feng, John Wieting, Xiaodong Yu, Yangqiu Song, Shashank Gupta, Shyam Upadhyay, Naveen Arivazhagan, Qiang Ning, Shaoshi Ling, and Dan Roth. 2018b. CogCompNLP: your swiss army knife for nlp. In *Proceedings of International Conference on Language Resources and Evaluation (LREC)*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard H. Hovy. 2017. RACE: large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017*. pages 785–794. https://aclanthology.info/papers/D17-1082/d17-1082.

J. McCarthy. 1976. An example for natural language understanding and the AI problems it raises. Available at http://jmc.stanford.edu/articles/mrhug/mrhug.pdf.

Elizabeth Merkhofer, John Henderson, David Bloom, Laura Strickhart, and Guido Zarrella. 2018. Mitre at semeval-2018 task 11: Commonsense reasoning without commonsense knowledge. In *Proceedings of the International Workshop on Semantic Evaluation (SemEval-2018)*. New Orleans, LA, USA.

Sewon Min, Min Joon Seo, and Hannaneh Hajishirzi. 2017. Question answering through transfer learning from large fine-grained supervision data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Volume 2: Short Papers*. pages 510–517. https://doi.org/10.18653/v1/P17-2081.

Karthik Narasimhan and Regina Barzilay. 2015. Machine comprehension with discourse relations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, Volume 1: Long Papers*. pages 1253–1262. http://aclweb.org/anthology/P/P15/P15-1121.pdf.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. *CoRR* abs/1611.09268. http://arxiv.org/abs/1611.09268.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016*. pages 2383–2392. http://aclweb.org/anthology/D/D16/D16-1264.pdf.

Raymond Reiter. 1976. A semantically guided deductive system for automatic theorem proving. *IEEE Transactions on Computers* 25:328–334. https://doi.org/10.1109/TC.1976.1674613.

Matthew Richardson, Christopher J. C. Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013*. pages 193–203. http://aclweb.org/anthology/D/D13/D13-1020.pdf.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations, ICLR 2017*.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. Newsqa: A machine comprehension dataset. *CoRR* abs/1611.09830. http://arxiv.org/abs/1611.09830.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *CoRR* abs/1502.05698. http://arxiv.org/abs/1502.05698.

J. Wieting, M. Bansal, K. Gimpel, K. Livescu, and D. Roth. 2015. From paraphrase database to compositional paraphrase model and back. *TACL* 3:345–358.

Terry Winograd. 1980. Extended inference modes in reasoning by computer systems. *Artificial Intelligence* 13:5–26.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*. pages 2013–2018. http://aclweb.org/anthology/D/D15/D15-1237.pdf.

Neural Automated Essay Scoring and Coherence Modeling for Adversarially Crafted Input

Youmna Farag Helen Yannakoudakis Ted Briscoe

Department of Computer Science and Technology
The ALTA Institute
University of Cambridge
United Kingdom
{youmna.farag,helen.yannakoudakis,ted.briscoe}@cl.cam.ac.uk

Abstract

We demonstrate that current state-of-the-art approaches to Automated Essay Scoring (AES) are not well-suited to capturing adversarially crafted input of grammatical but incoherent sequences of sentences. We develop a neural model of local coherence that can effectively learn connectedness features between sentences, and propose a framework for integrating and jointly training the local coherence model with a state-of-the-art AES model. We evaluate our approach against a number of baselines and experimentally demonstrate its effectiveness on both the AES task and the task of flagging adversarial input, further contributing to the development of an approach that strengthens the validity of neural essay scoring models.

1 Introduction

Automated Essay Scoring (AES) focuses on automatically analyzing the quality of writing and assigning a score to the text. Typically, AES models exploit a wide range of manually-tuned shallow and deep linguistic features (Shermis and Hammer, 2012; Burstein et al., 2003; Rudner et al., 2006; Williamson et al., 2012; Andersen et al., 2013). Recent advances in deep learning have shown that neural approaches to AES achieve state-of-the-art results (Alikaniotis et al., 2016; Taghipour and Ng, 2016) with the additional advantage of utilizing features that are automatically learned from the data. In order to facilitate interpretability of neural models, a number of visualization techniques have been proposed to identify textual (superficial) features that contribute to model performance (Alikaniotis et al., 2016).

To the best of our knowledge, however, no prior work has investigated the robustness of neural AES systems to adversarially crafted input that is designed to trick the model into assigning desired missclassifications; for instance, a high score to a low quality text. Examining and addressing such validity issues is critical and imperative for AES deployment. Previous work has primarily focused on assessing the robustness of "standard" machine learning approaches that rely on manual feature engineering; for example, Powers et al. (2002); Yannakoudakis et al. (2011) have shown that such AES systems, unless explicitly designed to handle adversarial input, can be susceptible to subversion by writers who understand something of the systems' workings and can exploit this to maximize their score.

In this paper, we make the following contributions:

i. We examine the robustness of state-of-the-art neural AES models to adversarially crafted input,[1] and specifically focus on input related to *local coherence*; that is, grammatical but incoherent sequences of sentences.[2] In addition to the superiority in performance of neural approaches against "standard" machine learning models (Alikaniotis et al., 2016; Taghipour and Ng, 2016), such a setup allows us to investigate their potential superiority / capacity in handling adversarial input without being explicitly designed to do so.

ii. We demonstrate that state-of-the-art neural AES is not well-suited to capturing adversarial input of grammatical but incoherent sequences of sentences, and develop a neural model of local coherence that can effectively learn connectedness features between sentences.

[1] We use the terms 'adversarially crafted input' and 'adversarial input' to refer to text that is designed with the intention to trick the system.

[2] Coherence can be assessed locally in terms of transitions between adjacent sentences.

Proceedings of NAACL-HLT 2018, pages 263–271
New Orleans, Louisiana, June 1 - 6, 2018.

iii. A local coherence model is typically evaluated based on its ability to rank coherently ordered sequences of sentences higher than their incoherent / permuted counterparts (e.g., Barzilay and Lapata (2008)). We focus on a stricter evaluation setting in which the model is tested on its ability to rank coherent sequences of sentences higher than *any* incoherent / permuted set of sentences, and not just its own permuted counterparts. This supports a more rigorous evaluation that facilitates development of more robust models.

iv. We propose a framework for integrating and jointly training the local coherence model with a state-of-the-art AES model. We evaluate our approach against a number of baselines and experimentally demonstrate its effectiveness on both the AES task and the task of flagging adversarial input, further contributing to the development of an approach that strengthens AES validity.

At the outset, our goal is to develop a framework that strengthens the validity of state-of-the-art neural AES approaches with respect to adversarial input related to local aspects of coherence. For our experiments, we use the Automated Student Assessment Prize (ASAP) dataset,[3] which contains essays written by students ranging from Grade 7 to Grade 10 in response to a number of different prompts (see Section 4).

2 Related Work

AES Evaluation against Adversarial Input One of the earliest attempts at evaluating AES models against adversarial input was by Powers et al. (2002) who asked writing experts – that had been briefed on how the e-Rater scoring system works – to write essays to trick e-Rater (Burstein et al., 1998). The participants managed to fool the system into assigning higher-than-deserved grades, most notably by simply repeating a few well-written paragraphs several times. Yannakoudakis et al. (2011) and Yannakoudakis and Briscoe (2012) created and used an adversarial dataset of well-written texts and their random sentence permutations, which they released in the public domain, together with the grades assigned by a human expert to each piece of text. Unfortunately, however, the dataset is quite small, consisting of 12 texts in total. Higgins and Heilman (2014) proposed a framework for evaluating the susceptibility of AES systems to gaming behavior.

Neural AES Models Alikaniotis et al. (2016) developed a deep bidirectional Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) network, augmented with score-specific word embeddings that capture both contextual and usage information for words. Their approach outperformed traditional feature-engineered AES models on the ASAP dataset. Taghipour and Ng (2016) investigated various recurrent and convolutional architectures on the same dataset and found that an LSTM layer followed by a Mean over Time operation achieves state-of-the-art results. Dong and Zhang (2016) showed that a two-layer Convolutional Neural Network (CNN) outperformed other baselines (e.g., Bayesian Linear Ridge Regression) on both in-domain and domain-adaptation experiments on the ASAP dataset.

Neural Coherence Models A number of approaches have investigated neural models of coherence on news data. Li and Hovy (2014) used a window approach where a sliding kernel of weights was applied over neighboring sentence representations to extract local coherence features. The sentence representations were constructed with recursive and recurrent neural methods. Their approach outperformed previous methods on the task of selecting maximally coherent sentence orderings from sets of candidate permutations (Barzilay and Lapata, 2008). Lin et al. (2015) developed a hierarchical Recurrent Neural Network (RNN) for document modeling. Among others, they looked at capturing coherence between sentences using a sentence-level language model, and evaluated their approach on the sentence ordering task. Tien Nguyen and Joty (2017) built a CNN over entity grid representations, and trained the network in a pairwise ranking fashion. Their model outperformed other graph-based and distributed sentence models.

We note that our goal is not to identify the "best" model of local coherence on randomly permuted grammatical sentences in the domain of AES, but rather to propose a framework that strengthens the validity of AES approaches with respect to adversarial input related to local aspects of coherence.

[3] https://www.kaggle.com/c/asap-aes/

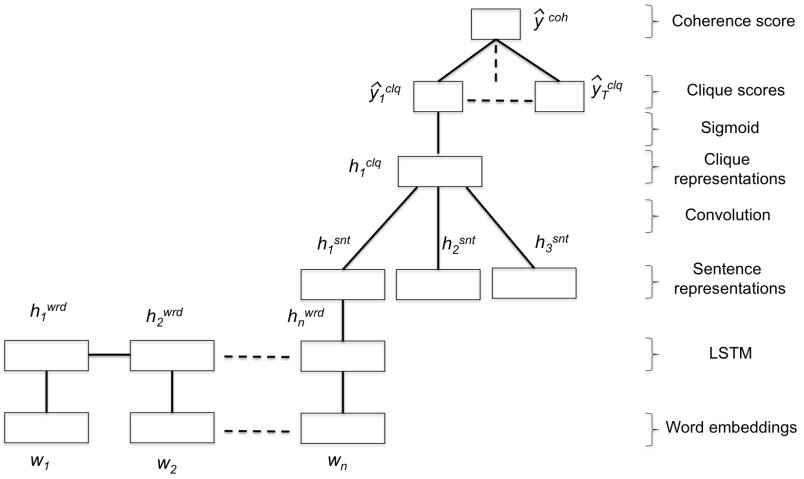


Figure 1: Local Coherence (LC) model architecture using a window of size 3. All h^{snt} representations are computed the same way as h_1^{snt}. The figure depicts the process of predicting the first clique score, which is applied to all the cliques in the text. The output coherence score is the average of all the clique scores. T is the number of cliques.

3 Models

3.1 Local Coherence (LC) Model

Our local coherence model is inspired by the model of Li and Hovy (2014) which uses a window approach to evaluate coherence.[4] Figure 1 presents a visual representation of the network architecture, which is described below in detail.

Sentence Representation This part of the model composes the sentence representations that can be utilized to learn connectedness features between sentences. Each word in the text is initialized with a k-dimensional vector w from a pre-trained word embedding space. Unlike Li and Hovy (2014), we use an LSTM layer[5] to capture sentence compositionality by mapping words in a sentence $s = \{w_1, w_2, ..., w_n\}$ at each time step t (w_t, where $t \leq n$) onto a fixed-size vector $h_t^{wrd} \in \mathbb{R}^{d_{lstm}}$ (where d_{lstm} is a hyperparameter). The sentence representation h^{snt} is then the representation of the last word in the sentence:

$$h^{snt} = h_n^{wrd} \tag{1}$$

Clique Representation Each window of sentences in a text represents a clique $q = \{s_1, ..., s_m\}$, where m is a hyperparameter indicating the window size. A clique is assigned a score of 1 if it is coherent (i.e., the sentences are *not* shuffled) and 0 if it is incoherent (i.e., the sentences are shuffled). The clique embedding is created by concatenating the representations of the sentences it contains according to Equation 1. A convolutional operation – using a filter $W^{clq} \in \mathbb{R}^{m \times d_{lstm} \times d_{cnn}}$, where d_{cnn} denotes the convolutional output size – is then applied to the clique embedding, followed by a non-linearity in order to extract the clique representation $h^{clq} \in \mathbb{R}^{d_{cnn}}$:

$$h_j^{clq} = \tanh([h_j^{snt}; ..; h_{j+m-1}^{snt}] * W^{clq}) \tag{2}$$

Here, $j \in \{1, ..., N - m + 1\}$, N is the number of sentences in the text, and $*$ is the linear convolutional operation.

Scoring The cliques' predicted scores are calculated via a linear operation followed by a sigmoid function to project the predictions to a $[0, 1]$ probability space:

$$\hat{y}_j^{clq} = \text{sigmoid}(h_j^{clq}.V) \tag{3}$$

where $V \in \mathbb{R}^{d_{cnn}}$ is a learned weight. The network optimizes its parameters to minimize the negative log-likelihood of the cliques' gold scores y^{clq}, given the network's predicted scores:

$$L_{local} = \frac{1}{T}\sum_{j=1}^{T}[-y_j^{clq}\log(\hat{y}_j^{clq}) - (1 - y_j^{clq})\log(1 - \hat{y}_j^{clq})] \tag{4}$$

[4] We note that Li and Jurafsky (2017) also present an extended version of the work by Li and Hovy (2014), evaluated on different domains.

[5] LSTMs have been shown to produce state-of-the-art results in AES (Alikaniotis et al., 2016; Taghipour and Ng, 2016).

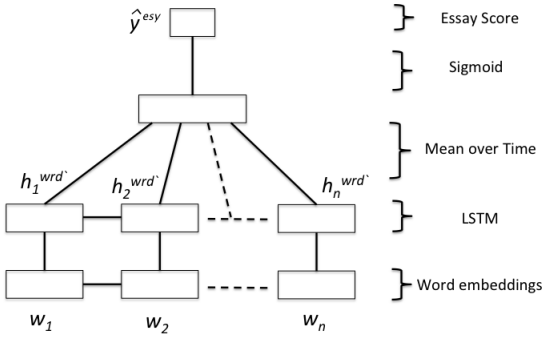


Figure 2: AES $\text{LSTM}_{\text{T\&N}}$ model of Taghipour and Ng (2016). The $\hat{y}^{esy}$ is the final predicted essay score.

where $T = N - m + 1$ (number of cliques in text). The final prediction of the text's coherence score is calculated as the average of all of its clique scores:

$$\hat{y}^{coh} = \frac{1}{T} \sum_{j=1}^{T} \hat{y}_j^{clq} \quad (5)$$

This is in contrast to Li and Hovy (2014), who multiply all the estimated clique scores to generate the overall document score. This means that if only one clique is misclassified as incoherent and assigned a score of 0, the whole document is regarded as incoherent. We aim to soften this assumption and use the average instead to allow for a more fine-grained modeling of degrees of coherence.[6]

We train the LC model on synthetic data automatically generated by creating random permutations of highly-scored ASAP essays (Section 4).

3.2 LSTM AES Model

We utilize the LSTM AES model of Taghipour and Ng (2016) shown in Figure 2 ($\text{LSTM}_{\text{T\&N}}$), which is trained, and yields state-of-the-art results on the ASAP dataset. The model is a one-layer LSTM that encodes the sequence of words in an essay, followed by a Mean over Time operation that averages the word representations generated from the LSTM layer.[7]

[6] Our experiments showed that using the multiplicative approach gives poor results, as presented in Section 6.

[7] We note that the authors achieve slightly higher results when averaging ensemble results of their LSTM model together with CNN models. We use their main LSTM model which, for the purposes of our experiments, does not affect our conclusions.

3.3 Combined Models

We propose a framework for integrating the $\text{LSTM}_{\text{T\&N}}$ model with the Local Coherence (LC) one. Our goal is to have a robust AES system that is able to correctly flag adversarial input while maintaining a high performance on essay scoring.

3.3.1 Baseline: Vector Concatenation (VecConcat)

The baseline model simply concatenates the output representations of the pre-prediction layers of the trained $\text{LSTM}_{\text{T\&N}}$ and LC networks, and feeds the resulting vector to a machine learning algorithm (e.g., Support Vector Machines, SVMs) to predict the final overall score. In the $\text{LSTM}_{\text{T\&N}}$ model, the output representation (hereafter referred to as the *essay representation*) is the vector produced from the Mean Over Time operation; in the LC model, we use the generated clique representations (Figure 1) aggregated with a max operation;[8] (hereafter referred to as the *clique representation*). Although the LC model is trained on permuted ASAP essays (Section 4) and the $\text{LSTM}_{\text{T\&N}}$ model on the original ASAP data, essay and clique representations are generated for both the ASAP and the synthetic essays containing reordered sentences.

3.3.2 Joint Learning

Instead of training the $\text{LSTM}_{\text{T\&N}}$ and LC models separately and then concatenating their output representations, we propose a framework where both models are trained jointly, and where the final network has then the capacity to predict AES scores and flag adversarial input (Figure 3).

Specifically, the $\text{LSTM}_{\text{T\&N}}$ and LC networks predict an essay and coherence score respectively (as described earlier), but now they both share the word embedding layer. The training set is the aggregate of both the ASAP and permuted data to allow the final network to learn from both simultaneously. Concretely, during training, for the ASAP essays, we assume that both the gold essay and coherence scores are the same and equal to the gold ASAP scores. This is not too strict an assumption, as overall scores of writing competence tend to correlate highly with overall coherence. For the synthetic essays, we set the "gold" coher-

[8] We note that max aggregation outperformed other aggregation functions.

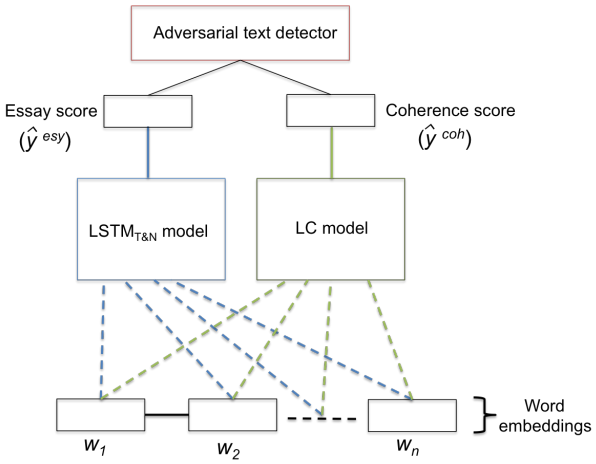


Figure 3: A joint network for scoring essays as well as detecting adversarial input. The LSTM$_{T\&N}$ model is the one depicted in Figure 2, and the LC in Figure 1.

ence scores to zero, and the "gold" essay scores to those of their original non-permuted counterparts in the ASAP dataset. The intuition is as follows: firstly, setting the "gold" essay scores of synthetic essays to zero would bias the model into over-predicting zeros; secondly, our approach reinforces the LSTM$_{T\&N}$'s inability to detect adversarial input, and forces the overall network to rely on the LC branch to identify such input.[9]

The two sub-networks are trained together and the error gradients are back-propagated to the word embeddings. To detect whether an essay is adversarial, we further augment the system with an *adversarial text detection* component that simply captures adversarial input based on the difference between the predicted essay and coherence scores. Specifically, we use our development set to learn a threshold for this difference, and flag an essay as adversarial if the difference is larger than the threshold. We experimentally demonstrate that this approach enables the model to perform well on both original ASAP and synthetic essays. During model evaluation, the texts flagged as adversarial by the model are assigned a score of zero, while the rest are assigned the predicted essay score ($\hat{y}^{esy}$ in Figure 3).

4 Data and Evaluation

We use the ASAP dataset, which contains 12,976 essays written by students ranging from Grade 7 to Grade 10 in response to 8 different prompts. We follow the ASAP data split by Taghipour and Ng (2016), and apply 5-fold cross validation in all experiments using the same train/dev/test splits. For each prompt, the fold predictions are aggregated and evaluated together. In order to calculate the overall system performance, the results are averaged across the 8 prompts.

To create adversarial input, we select high scoring essays per prompt (given a pre-defined score threshold, Table 1)[10] that are assumed coherent, and create 10 permutations per essay by randomly shuffling its sentences. In the joint learning setup, we augment the original ASAP dataset with a subset of the synthetic essays. Specifically, we randomly select 4 permutations per essay to include in the training set,[11] but include all 10 permutations in the test set. Table 1 presents the details of the datasets.

We test performance on the ASAP dataset using Quadratic Weighted Kappa (QWK), which was the official evaluation metric in the ASAP competition, while we test performance on the synthetic dataset using pairwise ranking accuracy (PRA) between an original non-permuted essay and its permuted counterparts. PRA is typically used as an evaluation metric on coherence assessment tasks on other domains (Barzilay and Lapata, 2008), and is based on the fraction of correct pairwise rankings in the test data (i.e., a coherent essay should be ranked higher than its permuted counterpart). Herein, we extend this metric and furthermore evaluate the models by comparing each original essay to all adversarial / permuted essays in the test data, and not just its own permuted counterparts – we refer to this metric as *total pairwise ranking accuracy* (TPRA).

5 Model Parameters and Baselines

Coherence models We train and test the LC model described in Section 3.1 on the synthetic dataset and evaluate it using PRA and TPRA. During pre-processing, words are lowercased and initialized with pre-trained word embeddings (Zou et al., 2013). Words that occur only once in the training set are mapped to a special *UNK* embed-

[9] We note that, during training, the scores are mapped to a range between 0 and 1 (similarly to Taghipour and Ng (2016)), and then scaled back to their original range during evaluation.

[10] We note that this threshold is different than the one mentioned in Section 3.3.2.

[11] This is primarily done to keep the data balanced: initial experiments showed that training with all 10 permutations per essay harms AES performance, but has negligible effect on adversarial input detection.

Prompt	#ASAP essays	Score Range	Synthetic Dataset		
			threshold	#ASAP essays	total size
1	1,783	2–12	10	472	5,192
2	1,800	1–6	5	82	902
3	1,726	0–3	3	407	4,477
4	1,772	0–3	3	244	2,684
5	1,805	0–4	4	258	2,838
6	1,800	0–4	4	367	4,037
7	1,569	0–30	23	179	1,969
8	723	0–60	45	72	792

Table 1: Statistics for each dataset per prompt. For the synthetic dataset, the high scoring ASAP essays are selected based on the indicated score threshold (inclusive). "total size" refers to the number of the ASAP essays selected + their 10 different permutations.

ding. All network weights are initialized to values drawn randomly from a uniform distribution with scale $= 0.05$, and biases are initialized to zeros. We apply a learning rate of 0.001 and RMSProp (Tieleman and Hinton, 2012) for optimization. A size of 100 is chosen for the hidden layers (d_{lstm} and d_{cnn}), and the convolutional window size (m) is set to 3. Dropout (Srivastava et al., 2014) is applied for regularization to the output of the convolutional operation with probability 0.3. The network is trained for 60 epochs and performance is monitored on the development sets – we select the model that yields the highest PRA value.[12]

We use as a baseline the LC model that is based on the multiplication of the clique scores (similarly to Li and Hovy (2014)), and compare the results (LC_{mul}) to our averaged approach. As another baseline, we use the entity grid (EGrid) (Barzilay and Lapata, 2008) that models transitions between sentences based on sequences of entity mentions labeled with their grammatical role. EGrid has been shown to give competitive results on similar coherence tasks in other domains. Using the Brown Coherence Toolkit (Eisner and Charniak, 2011),[13] we construct the entity transition probabilities with length = 3 and salience = 2. The transition probabilities are then used as features that are fed as input to an SVM classifier with an *RBF* kernel and penalty parameter $C = 1.5$ to predict a coherence score.

$\text{LSTM}_{\text{T\&N}}$ model We replicate and evaluate the LSTM model of Taghipour and Ng (2016)[14] on ASAP and our synthetic data.

Combined models After training the LC and $\text{LSTM}_{\text{T\&N}}$ models, we concatenate their output vectors to build the **Baseline: Vector Concatenation (VecConcat)** model as described in Section 3.3.1, and train a Kernel Ridge Regression model.[15]

The **Joint Learning** network is trained on both the ASAP and synthetic dataset as described in Section 3.3.2. Adversarial input is detected based on an estimated threshold on the difference between the predicted essay and coherence scores (Figure 3). The threshold value is empirically calculated on the development sets, and set to be the average difference between the predicted essay and coherence scores in the synthetic data:

$$\text{threshold} = \frac{\sum_{i=1}^{M} \hat{y}_i^{esy} - \hat{y}_i^{coh}}{M}$$

where M is the number of synthetic essays in the development set.

We furthermore evaluate a baseline where the joint model is trained *without* sharing the word embedding layer between the two submodels, and report the effect on performance ($\text{Joint Learning}_{\text{no\_layer\_sharing}}$). Finally, we evaluate a baseline where for the joint model we set the "gold" essay scores of synthetic data to zero ($\text{Joint Learning}_{\text{zero\_score}}$), as opposed to our proposed approach of setting them to be the same as the score of their original non-permuted counterpart in the ASAP dataset.

6 Results

The state-of-the-art $\text{LSTM}_{\text{T\&N}}$ model, as shown in Table 2, gives the highest performance on the ASAP data, but is not robust to adversarial input and therefore unable to capture aspects of local coherence, with performance on synthetic data that is less than 0.5. On the other hand, both

[12] Our implementation is available at https://github.com/Youmna-H/Coherence_AES

[13] https://bitbucket.org/melsner/browncoherence

[14] https://github.com/nusnlp/nea

[15] We use scikit-learn with the following parameters: alpha=0.1, coef0=1, degree=3, gamma=0.1, kernel='rbf'.

Model	ASAP	Synthetic	
	QWK	PRA	TPRA
EGrid	–	0.718*	0.706*
LC	–	0.946*	0.689*
LC_{mul}	–	**0.948***	0.620*
$LSTM_{T\&N}$	**0.739**	0.430	0.473
VecConcat	0.719	0.614*	0.567*
Joint Learning	0.724	0.770*	**0.777***

Table 2: Model performance on ASAP and synthetic test data. Evaluation is based on the average QWK, PRA and TRPA across the 8 prompts. * indicates significantly different results compared to $LSTM_{T\&N}$ (two-tailed test with $p < 0.01$).

our LC model and the EGrid significantly outperform $LSTM_{T\&N}$ on synthetic data. While EGrid is slightly better in terms of TPRA compared to LC (0.706 vs. 0.689), LC is substantially better on PRA (0.946 vs. 0.718). This could be attributed to the fact that LC is optimised using PRA on the development set. The LC_{mul} variation has a performance similar to LC in terms of PRA, but is significantly worse in terms of TPRA, which further supports the use of our proposed LC model.

Our Joint Learning model manages to exploit the best of both the $LSTM_{T\&N}$ and LC approaches: performance on synthetic data is significantly better compared to $LSTM_{T\&N}$ (and in particular gives the highest TPRA value on synthetic data compared to all models), while manages to maintain the high performance of $LSTM_{T\&N}$ on ASAP data (performance slighly drops from 0.739 to 0.724 though not significantly). When the Joint Learning model is compared against the VecConcat baseline, we can again confirm its superiority on both datasets, giving significant differences on synthetic data.

7 Further Analysis

We furthermore evaluate the performance of the the Joint Learning model when trained using different parameters (Table 3). When assigning "gold" essay scores of zero to adversarial essays ($\text{Joint Learning}_{\text{zero\_score}}$), AES performance on the ASAP data drops to 0.449 QWK, and the results are statistically significant.[16] This is partly explained by the fact that the model, given the training data gold scores, is biased towards predicting zeros. The result, however, further supports our hypothesis that forcing the Joint Learning model to rely on the coherence branch for adversarial input detection further improves performance. Importantly, we need something more than just training a state-of-the-art AES model (in our case, $LSTM_{T\&N}$) on both original and synthetic data.

Model	ASAP	Synthetic	
	QWK	PRA	TPRA
Joint Learning	**0.724**	**0.770**	**0.777**
$\text{Joint Learning}_{\text{no\_layer\_sharing}}$	0.720	0.741	0.753*
$\text{Joint Learning}_{\text{zero\_score}}$	0.449*	–	–

Table 3: Evaluation of different Joint Learning model parameters. * indicates significantly different results compared to our Joint Learning approach.

We also compare Joint Learning to $\text{Joint Learning}_{\text{no\_layer\_sharing}}$ in which the the two sub-models are trained separately without sharing the first layer of word representations. While the difference in performance on the ASAP test data is small, the differences are much larger on synthetic data, and are significant in terms of TPRA. By examining the false positives of both systems (i.e., the coherent essays that are misclassified as adversarial), we find that when the embeddings are not shared, the system is biased towards flagging long essays as adversarial, while interestingly, this bias is not present when the embeddings are shared. For instance, the average number of words in the false positive cases of $\text{Joint Learning}_{\text{no\_layer\_sharing}}$ on the ASAP data is 426, and the average number of sentences is 26; on the other hand, with the Joint Learning model, these numbers are 340 and 19 respectively.[17] A possible explanation for this is that training the words with more contextual information (in our case, via embeddings sharing), is advantageous for longer documents with a large number of sentences.

Ideally, no essays in the ASAP data should be flagged as adversarial as they were not designed to trick the system. We calculate the number of ASAP texts incorrectly detected as adversarial, and find that the average error in the Joint Learning model is quite small (0.382%). This increases with $\text{Joint Learning}_{\text{no\_layer\_sharing}}$ (1%), although still remains relatively small.

[16] Note that we do not report performance of this model on synthetic data. In this case, the thresholding technique cannot be applied as both sub-models are trained with the same "gold" scores and thus have very similar predictions on synthetic data.

[17] Adversarial texts in the synthetic dataset have an average number of 306 words and an average number of 18 sentences.

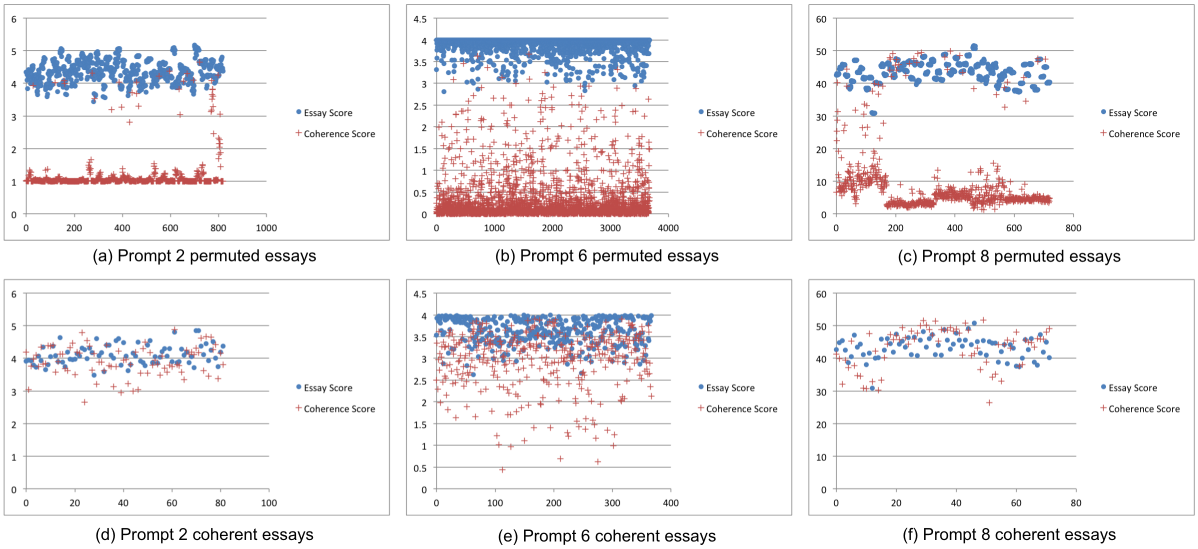


Figure 4: Joint Learning model predictions on the synthetic test set for 3 randomly selected prompts. The upper graphs ((a), (b) and (c)) show the predicted essay and coherence scores on adversarial text, while the bottom ones ((d), (e) and (f)) show the predicted scores for highly scored / coherent ASAP essays. The blue circles represent the essay scores, and the red pluses the coherence scores. All predicted scores are mapped to their original scoring scale.

We further investigate the essay and coherence scores predicted by our best model, Joint Learning, for the permuted and original ASAP essays in the *synthetic* dataset (for which we assume that the selected, highly scored ASAP essays are coherent, Section 4), and present results for 3 randomly selected prompts in Figure 4. The graphs show a large difference between predicted essay and coherence scores on permuted / adversarial data ((a), (b) and (c)), where the system predicts high essay scores for permuted texts (as a result of our training strategy), but low coherence scores (as predicted by the LC model). For highly scored ASAP essays ((d), (e) and (f)), the system predictions are less varied and positively contributes to the performance of our proposed approach.

8 Conclusion

We evaluated the robustness of state-of-the-art neural AES approaches on adversarial input of grammatical but incoherent sequences of sentences, and demonstrated that they are not well-suited to capturing such cases. We created a synthetic dataset of such adversarial examples and trained a neural local coherence model that is able to discriminate between such cases and their coherent counterparts. We furthermore proposed a framework for jointly training the coherence model with a state-of-the-art neural AES model, and introduced an effective strategy for assigning "gold" scores to adversarial input during training. When compared against a number of baselines, our joint model achieves better performance on randomly permuted sentences, while maintains a high performance on the AES task. Among others, our results demonstrate that it is not enough to simply (re-)train neural AES models with adversarially crafted input, nor is it sufficient to rely on "simple" approaches that concatenate output representations from different neural models. Finally, our framework strengthens the validity of neural AES approaches with respect to adversarial input designed to trick the system.

Acknowledgements

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X Pascal GPU used for this research. We are also grateful to Cambridge Assessment for their support of the ALTA Institute. Special thanks to Christopher Bryant and Marek Rei for their valuable feedback.

References

Dimitrios Alikaniotis, Helen Yannakoudakis, and Marek Rei. 2016. Automatic text scoring using neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 715–725.

Øistein E Andersen, Helen Yannakoudakis, Fiona Barker, and Tim Parish. 2013. Developing and testing a self-assessment and tutoring system. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications, BEA*. Association for Computational Linguistics, pages 32–41.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics* 34(1):1–34.

Jill Burstein, Martin Chodorow, and Claudia Leacock. 2003. Criterion: Online essay evaluation: An application for automated evaluation of student essays. In *Proceedings of the fifteenth annual conference on innovative applications of artificial intelligence*. American Association for Artificial Intelligence, pages 3–10.

Jill Burstein, Karen Kukich, Susanne Wolff, Chi Lu, Martin Chodorow, Lisa Braden-Harder, and Mary Dee Harris. 1998. Automated scoring using a hybrid feature identification technique. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*. pages 206–210.

Fei Dong and Yue Zhang. 2016. Automatic features for essay scoring – an empirical study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 1072–1077.

Micha Eisner and Eugene Charniak. 2011. Extending the entity grid with entity-specific features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*. Association for Computational Linguistics, pages 125–129.

Derrick Higgins and Michael Heilman. 2014. Managing what we can measure: Quantifying the susceptibility of automated scoring systems to gaming behavior. *Educational Measurement: Issues and Practice* 33:3646.

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

Jiwei Li and Eduard Hovy. 2014. A model of coherence based on distributed sentence representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 2039–2048.

Jiwei Li and Dan Jurafsky. 2017. Neural net models for open-domain discourse coherence. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 198–209.

Rui Lin, Shujie Liu, Muyun Yang, Mu Li, Ming Zhou, and Sheng Li. 2015. Hierarchical recurrent neural network for document modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 899–907.

Donald E. Powers, Jill Burstein, Martin Chodorow, Mary E. Fowles, and Karen Kukich. 2002. Stumping e-rater: challenging the validity of automated essay scoring. *Computers in Human Behavior* 18(2):103–134.

LM Rudner, Veronica Garcia, and Catherine Welch. 2006. An evaluation of IntelliMetric essay scoring system. *The Journal of Technology, Learning, and Assessment* 4(4):1 – 22.

M Shermis and B Hammer. 2012. Contrasting state-of-the-art automated scoring of essays: analysis. In *Annual National Council on Measurement in Education Meeting*. pages 1–54.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.

Kaveh Taghipour and Hwee Tou Ng. 2016. A neural approach to automated essay scoring. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1882–1891.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5 - rmsprop. *Technical report* .

Dat Tien Nguyen and Shafiq Joty. 2017. A neural local coherence model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1320–1330.

DM Williamson, Xiaoming Xi, and FJ Breyer. 2012. A framework for evaluation and use of automated scoring. *Educational Measurement: Issues and Practice* 31(1):2–13.

Helen Yannakoudakis and Ted Briscoe. 2012. Modeling coherence in esol learner texts. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*. Association for Computational Linguistics, pages 33–43.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading esol texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pages 180–189.

Will Y Zou, Richard Socher, Daniel M Cer, and Christopher D Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *EMNLP*. pages 1393–1398.

QuickEdit: Editing Text & Translations by Crossing Words Out

David Grangier **Michael Auli**
Facebook AI Research
Menlo Park, CA

Abstract

We propose a framework for computer-assisted text editing. It applies to translation post-editing and to paraphrasing. Our proposal relies on very simple interactions: a human editor modifies a sentence by marking tokens they would like the system to change. Our model then generates a new sentence which reformulates the initial sentence by avoiding marked words. The approach builds upon neural sequence-to-sequence modeling and introduces a neural network which takes as input a sentence along with change markers. Our model is trained on translation bitext by simulating post-edits. We demonstrate the advantage of our approach for translation post-editing through simulated post-edits. We also evaluate our model for paraphrasing through a user study.

1 Introduction

Computers can help humans edit text more efficiently. In particular, statistical models are used for that purpose, for instance to help correct spelling mistakes (Brill and Moore, 2000) or suggest likely completions of a sentence (Bickel et al., 2005). In this work, we rely on statistical learning to enable a computer to rephrase a sentence by only pointing at words that should be avoided. Specifically, we consider the task of reformulating either a sentence, i.e. *paraphrasing* (Quirk et al., 2004), or a translation, i.e. *translation post-editing* (Koehn, 2009b). Paraphrasing reformulates a sentence with different words preserving its meaning, while translation post-editing takes a candidate translation along with the corresponding source sentence and improves it.

Our proposal relies on very simple interactions: a human editor modifies a sentence by selecting tokens they would like the system to replace and no other feedback. Our system then generates a new sentence which reformulates the initial sentence by avoiding the word types from the selected tokens. Our approach builds upon neural sequence-to-sequence and introduces a neural network which takes as input a sentence along with token markers. We introduce a novel attention-based architecture suited to this goal and propose a training procedure based on simulated post-edits on translation bitext (§3). This approach allows to get substantial modifications of the initial sentence – including deletion, reordering and insertion of multiple words – with limited user effort.

Our experiments (§4) relies on large scale simulated post-edits. They show that our model outperforms our post-editing baseline by up to 5 BLEU points on WMT'14 English-German and WMT'14 German-English translation. The advantage of our method is also highlighted in monolingual settings, where we analyze the quality of the paraphrases generated by our model in a user study.

Before introducing our method (§3) and its empirical evaluation (§4), we describe related work in the next section.

2 Related Work

Our work builds upon previous research on neural machine translation, machine translation post-editing, and computer-assisted editing.

2.1 Neural Machine Translation

Statistical machine translation systems models automatically translate text relying on large corpora of bitext, i.e. corresponding pairs of sentences in the source and target language (Koehn, 2009a). Recently, machine translation systems based on neural networks have emerged as an effective approach to this problem (Sutskever et al., 2014). Neural networks are a departure from count-based translation systems, e.g. phrase-based systems, which used to dominate the field (Koehn, 2009a).

Research in Neural Machine Translation (NMT) focuses notably on identifying appropri-

Proceedings of NAACL-HLT 2018, pages 272–282
New Orleans, Louisiana, June 1 - 6, 2018.

ate neural architecture. Cho et al. (2014) and Suskever et al. (2014) proposed encoder/decoder models. These models consist of a Recurrent Neural Network (RNN) mapping the source sentence sentence into a latent vector (encoder). This vector conditions an RNN language model (decoder) which generates the target sentence (Mikolov et al., 2010; Graves, 2013). Bahdanau et al. (2014) adds attention to these models, which leverages that the explanation for a given target word in generally localized around a few source words. Recently, new architectures have proposed to replace recurrent modules with convolutions (Gehring et al., 2017) or self-attention (Vaswani et al., 2017) to further increase accuracy. These architecture also perform attention at more than one decoder layer, allowing for more complex attention patterns. In this work, we build upon the architecture of Gehring et al. (2017) since this model offers a good trade-off between high accuracy and fast decoding.

2.2 Translation Post-Editing

Post-editing leverages a machine translation system and enable human translators to edit its output with different levels of computer assistance. This enables improving machine translation outputs with lesser effort than purely manual translation.

Green et al. (2014) implement such a system relying on a phrase-based translation system. The system presents an initial translation to the user who can accept a prefix and select among the most likely postfix iteratively. Similar ideas relying on decoding with prefix constrains are common in post-translation (Langlais et al., 2000; Koehn, 2009b; Barrachina et al., 2009). Recently, these approaches based on left-to-right decoding have been extended to neural machine translation (Peris et al., 2017).

Closer to our work, Marie and Max (2015) propose light-weight interactions based on accepting/rejecting spans from the output of a statistical machine translation system. The user labels each span that should appear in the final translation. Unmarked spans are assumed to be undesirable and the system removes any entries that could generate those spans from the phrase table. The phrase table is modified such that only positively marked target spans are allowed to explain the corresponding source phrases.

Compared to their work, we rely on similar interactions but we do not require the user to label every token as either accepted or rejected. The user only needs to mark a few rejections. Also, we build on a more accurate neural translation model which is not amenable to phrase table editing. Finally, our method is equally applicable to the monolingual editing of regular text.

Automatic post-editing (APE) (Lagarda et al., 2009), i.e. a process which automatically modifies an MT output without human guidance (Lagarda et al., 2009), is also an active area of research. Although APE shares similarities to classical post-editing, it is beyond the scope of this paper.

2.3 Computer-Assisted Text Editing

Computer assisted text editing has been introduced with interactive computer terminals (Irons and Djorup, 1972). Its first achievement was to simplify the insertion, deletion, and copy of text compared to typewriters. Computers then enabled the emergence of computerized language assistance tools such as spelling correctors (Brill and Moore, 2000) or next word suggestions (Bickel et al., 2005).

More recently, research has focused on generating paraphrases (Bannard and Callison-Burch, 2005; Mallinson et al., 2017), compressing sentences (Rush et al., 2015) or simplifying sentences (Nisioi et al., 2017). This type of work expands the possibilities for *interactive* text generation tools, like our work.

Related to our work, Filippova et al. (2015) considers the task of predicting which tokens can be removed from a sentence without modifying its meaning relying on a recurrent neural network. Our work pursues a different goal since our model does not predict which token to remove, as the user provides this information. Our generation is more involved as our model rephrases the sentences, which includes introducing new words, reordering text, inflecting nouns and verbs, etc. Guu et al. (2017) considers generating text with latent edits. Their goal is not to enable users to control which words need to be changed in an initial sentence but to enable sampling valid English sentences with high lexical overlap around a starting sentence. Contrary to paraphrasing, such samples might introduce negations and other changes impacting meaning.

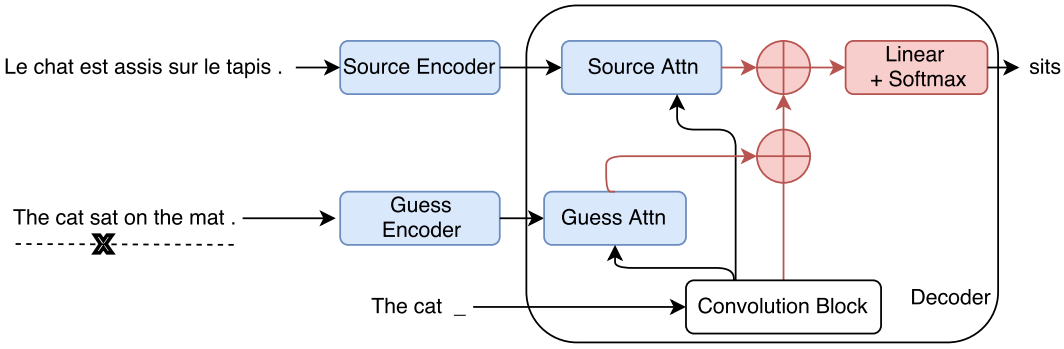


Figure 1: QuickEdit architecture for translation post-editing. The decoder attends to both encodings, one for the source and one for the initial translation (guess) with deletion markers (X on the diagram). Our simplified schema shows one convolutional block and single-hop attention for readability.

3 QuickEdit

QuickEdit is our sequence-to-sequence model for post-editing via delete actions. This model takes as input a *source* sentence and an initial *guess* target sentence annotated with change markers. It then aims to improve upon the guess by generating a better target sentence which avoids the marked tokens.

3.1 Model Architecture

Our model builds upon the architecture of Gehring et al. (2017). This model is a sequence to sequence neural model with attention. Both the encoder and decoder are deep convolutional networks with residual connections. The model performs *multi-hop* attention, i.e. each layer of the decoder attends to the encoder outputs. Our architecture choice is motivated by the accuracy of this model along with its computational efficiency.

QuickEdit adds a *second* encoder to represent the annotated guess sentence. It also duplicates every attention layer to allow the decoder to attend both to the source and the guess sentences. Dual attention has been introduced recently in the context of automatic post-editing (Novak et al., 2016; Libovický and Helcl, 2017). Our work is however the first work to introduce dual attention in a multi-hop architecture. Figure 1 illustrates our architecture.

The encoder of the initial guess takes as input a target sentence t annotated with binary change labels c, i.e.

$$g = \{g_i\}_{i=1}^{l_g} \text{ where } \forall i, g_i = (t_i, c_i)$$

in which l_g denotes the length of the guess, t_i is an index in the target vocabulary and c_i is a binary variable with 1 indicating a request to change the token by the user and 0 indicating no user preference. The first layer of the encoder maps this sequence to two embedding sequences, i.e. a sequence of target word embeddings and a sequence of positional embeddings. Compared to (Gehring et al., 2017), we extend the positional embedding to contain two types of vectors, positional vectors associated with positions i where $c_i = 0$ and positional vectors associated with positions i where $c_i = 1$. Like all parameters in the system, both sets of embeddings are learned to maximize the log-likelihood of the training reference sentences conditioned on the source, annotated guess pairs.

The attention over two sentences is simple. Both source and guess encoders produce a sequence of key and value pairs. We denote the output of the source encoder as $\{(k_i^s, v_i^s)\}_{i=1}^{l_s}$ and the output of the guess encoder as $\{(k_i^g, v_i^g)\}_{i=1}^{l_g}$. At each decoder layer k and time step j, the decoder produces a latent state vector h_j^k, this vector attends to the output of the source encoder,

$$a_i^s = \exp\left(h_j^k \cdot k_i^s\right) / \sum_l \exp\left(h_j^k \cdot k_l^s\right)$$

and the guess encoder,

$$a_i^g = \exp\left(h_j^k \cdot k_i^g\right) / \sum_l \exp\left(h_j^k \cdot k_l^g\right).$$

This attention weights are used to summarize the values of the source $\sum_i a_i^s v_i^s$ and the guess $\sum_i a_i^s v_i^g$ respectively. The attention module then averages these two vectors $\frac{1}{2}\sum_i a_i^s v_i^s + \frac{1}{2}\sum_i a_i^g v_i^g$ and uses this average instead of the source attention output in the next layer (Gehring et al., 2017).

3.2 Training & Inference

Our model is trained on translation bitext by simulating post-edits. Given a bitext corpus, we first train an initial translation system and we then rely on this system to translate the training corpus. This strategy results in three sentences for each example: the source, the guess (i.e. the sentence decoded from the initial system) and the reference sentence. Post-edits are simulated by marking guess tokens which do not appear in the corresponding reference sentence.

The dual attention model presented in the above section is then trained. We maximize the log-likelihood of the training reference sentences y given each corresponding source sentence x and the annotated guess g, i.e. we maximize

$$\mathcal{L}_{\text{Train}} : \theta \rightarrow \sum_{(x,y,g) \in \text{Train}} \log P(y|x, g, \theta)$$

where y refers to the reference sentence, x refers to the source sentence and g is the annotated guess sentence as defined above. Training relies on stochastic gradient descent (Bottou, 1991), using Nesterov's accelerated gradient with momentum (Nesterov, 1983; Sutskever et al., 2013). At inference time, we decode through standard left-to-right beam search (Sutskever et al., 2014). Our decoding strategy for QuickEdit also incorporates hard constraints that prevent the decoder from outputting tokens which are marked in the guess.

3.3 Extension to Monolingual Editing

The extension of QuickEdit to a monolingual setting is straightforward: we remove the source encoder and the corresponding attention path. This results in a single encoder model which takes only an annotated guess as input. This model can be trained from pairs of sentences consisting of a machine translation output along with the corresponding reference sentence. Although machine translation bitext are used to create this model training data, it operates solely on target language sentences without requiring a source sentence at test time. In our experiments, we train distinct models for the monolingual setting. We do not consider sharing parameters with the translation models at this point.

4 Experiments & Results

We evaluate on three translation datasets of increasing size and we report results in both language directions: IWSLT'14 German-English (Cettolo et al., 2014), WMT'14 German-English (Luong et al., 2015), and WMT'14 English-French (Bojar et al., 2014). Our post-editing baseline is our initial neural translation system, complemented with decoding constraints to disallow marked guess words to be considered in the beam. For paraphrasing, we compare our model trained on WMT'14 fr-en to the model of (Mallinson et al., 2017) on the MTC dataset (Huang et al., 2002) following their setup. We relied on WMT'14 fr-en training data motivated by its size[1].

For IWSLT'14 we train on 160K sentence pairs and we validate on a random subset of 7,250 sentence-pairs held-out from the original training corpus. We test on the concatenation of *tst2010, tst2011, tst2012, tst2013, dev2010* and *dev2012* comprising 6,750 sentence pairs. The vocabulary for this dataset is 24k for English and 36k for German. For WMT'14 English to German and German to English, we use the same setup as Luong et al. (2015) which comprises 4.5M sentence pairs for training and we test on newstest2014.[2] We took 45k sentences out of the training set for validation purpose. As vocabulary, we learn a joint source and target byte-pair encoding (BPE) with 44k types from the training set (Sennrich et al., 2016b,a). Note that even when using BPE, we solely rely on full word markers, i.e. all the BPE tokens of a given word carry the same binary indication (to be changed/no preference). For WMT'14 English to French and French to English (Bojar et al., 2014), we also rely on BPE with 44k types. This dataset is larger with 35.4M sentences for training and 26k sentences for validation. We rely on newstest2014 for testing[3].

The model architecture settings are borrowed from (Gehring et al., 2017). For IWSLT'14 de-en and IWSLT'14 en-de, we rely on 4-layer encoders and 3-layer decoders, both with 256 hidden units and kernel width 3. The word embedding for source and target as well as the output matrix have 256 dimensions. For WMT'14 en-de and WMT'14 de-en, both encoders and decoders have 15 layers (9 layers with 512 hidden units, 4

[1] Posterior to our experiments, (Wieting and Gimpel, 2017) released an even large dataset that might be used in our setting.

[2] http://nlp.stanford.edu/projects/nmt

[3] http://www.statmt.org/wmt14/translation-task.html

layers with 1,024 units followed by 2 layers with 2,048 units). Input embeddings have 768 dimensions, output embedding have 512. For WMT'14 en-fr and WMT'14 fr-en, both encoders and decoders have 15 layers (6 layers with 512 hidden units, 4 layers with 768 units, 3 layers with 1024 units, followed by two larger layers with 2048 and 4096 units). Similar to the German model, input embeddings have 768 dimensions, output embedding have 512 dimensions. For all datasets, we decode using beam search with a beam of size 5.

4.1 Post-editing

Our study is based on simulated post-edits, i.e. simulated token deletion actions. We start from machine translation outputs from an initial system in which we label tokens to change automatically. For initial translation, we rely on the convolutional translation system from (Gehring et al., 2017)[4] learned from the training portion of the dataset. For each system output, any word which does not belong to the reference translation is marked to be changed. We perform this operation for the train, validation and test portion of each dataset. The training and validation portion can be used for learning and developing our post-editing system. The test portion is used for evaluation.

Table 1 reports our result on this task. Our QuickEdit method strongly outperforms the baseline post-editing system. Both systems access the same information, i.e. a list of deleted word types, which constrains the decoding. QuickEdit adds attention over the initial sentence with rejection marks. This has a big impact on BLEU. On the larger WMT'14 en-de benchmark, the advantage is over 5 BLEU point for both directions. We conjecture that the improvement is lower on the smaller IWSLT data due to over-fitting, i.e. the base system is excellent on the training set which reduces the post-editing opportunities on the training data, therefore limiting the amount of supervised data for training our post-editing system. We show examples of post-editing from the test set of WMT-14 de-en in Table 2. These examples show the ability of the model to rephrase sentences avoiding the marked tokens while preserving the source meaning. Similar to our experiments on WMT'14 en-de, QuickEdit also reports large improvement with respect to the baseline model on WMT'14 en-fr, with $+5.6$ points (53.4 vs 47.8).

One should note that the simulated edits rely on gold information, i.e. crossed-out words are always absent from the reference. Our aim is to simulate a post-editor which might have a sentence close to the reference in mind. This evaluation method allows to conduct large scale experiments without labeling burden. Conducting an interactive post-editing study requires trained editors and interface consideration beyond the scope of this initial work.

4.2 Partial Feedback

So far, our post-editing setting marked all incorrect words in the guess. We now consider a setting where the simulated post-editor performs less work by marking only a subset of these tokens. This is analogous to a hypothetical online translation service which offers a feature enabling the user to mark parts of a translation to be improved. In addition to marking only a subset of the incorrect tokens at inference time, we also train new models for which the training data also only had a subset of incorrect tokens marked. Specifically, we train three models QE25, QE50, QE100 for which either 25%, 50% or 100% of incorrect guess tokens were marked.

In this setting, we also compare with the baseline model, i.e. the initial translation system augmented with decoding constraints to avoid marked words. Figure 2 plots BLEU as a function of the number of marked words on the validation set of WMT'14 German to English. This curve is obtained by marking at most 1, 2, ..., 8 words to be changed per sentence, taking into account that the actual number of marked word in a sentence cannot be higher than the number of guess words not present in the reference sentence.

Compared to the baseline, there is a small advantage for QuickEdit for 1-2 marked words and a larger improvement when more words are marked. Unsurprisingly, the model trained with fewer marked words (QE25, QE50) performs better when tested with fewer marked words, while QE100 gives the largest improvement with 4 or more marked words.

4.3 Monolingual Editing

Table 1 also reports monolingual results. In that case, the system is not given the source sentence, only a sentence in the target language along with change markers. Even if the model is not given the

[4] https://github.com/facebookresearch/fairseq-py.

	IWSLT'14		WMT'14 (de)		WMT'14 (fr)	
	de→en	en→de	de→en	en→de	fr→en	en→fr
initial translation	27.4	24.2	29.7	25.2	37.0	40.2
post-edit baseline	33.0	30.2	34.6	30.7	45.4	47.8
post-edit QuickEdit	34.6	30.8	41.3	36.6	49.7	53.4
monolingual QuickEdit	29.3	26.7	39.5	34.2	47.7	51.3

Table 1: Editing results (BLEU4) when all incorrect tokens are requested to be changed.

source	Schauspieler Orlando Bloom hat sich zur Trennung von seiner Frau, Topmodel Miranda Kerr, geäußert.
guess	Actor Orlando Bloom ~~has spoken of the~~ separation ~~of~~ his wife, ~~Topmodel~~ Miranda Kerr.
output	Actor Orlando Bloom **spoke about** separation **from** his wife, **Top Model** Miranda Kerr.
source	Die heutigen elektronischen Geräte geben im Allgemeinen wesentlich weniger Funkstrahlung ab als frühere Generationen.
guess	Today's electronic devices generally ~~give far less~~ radio ~~radiation~~ than previous generations.
output	Today's electronic devices generally **emit significantly fewer** radio **frequencies** than previous generations.
source	Statt sich von der Zahlungsunfähigkeit der US-Regierung verunsichern zu lassen, konzentrierten sich Investoren auf das, was vermutlich mehr zählt: die Federal Reserve.
guess	~~Instead of~~ being ~~obscured~~ by the ~~US~~ government's ~~inability to pay~~, investors ~~focused~~ on what ~~is~~ probably more ~~important~~: the Federal Reserve.
output	**Rather than** being **insane** by the **United States** government's **insolvency**, investors **concentrated** on what probably **counts** more: the Federal Reserve.
source	Boeing bestreitet die Zahlen von Airbus zu den Sitzmaßen und sagt, es stehe nicht im Ermessen der Hersteller zu entscheiden, wie Fluggesellschaften die Balance zwischen Flugtarifen und Einrichtung gestalten.
guess	Boeing is ~~denying~~ the figures ~~from~~ Airbus to the ~~seats~~ and says that it is not ~~left to the discretion of~~ the manufacturers to decide how airlines ~~are~~ to balance air fares and ~~set~~ up.
output	Boeing is **contesting** Airbus**'s seating** figures and says it is not **up to** manufacturers to **determine** how airlines balance fares and **equipment**.

Table 2: Post-editing examples from WMT'14 en-de. Examples originate from news sentences of the newstest2014 dataset. Strike-through text indicates the tokens marked to be changed. Bold text indicates tokens introduced by the model, i.e. tokens not present in the original guess.

source, it manages to generate sentences which are closer to the reference than the initial sentences, as shown by the BLEU improvement. This shows the ability of the model to paraphrase from deletion constraints. Table 3 shows examples of the system in action from the English test set of WMT-14 fr-en. This examples show that the model can provide synonyms, e.g. *essential* → *vital*, or *came after* → *followed*. The model can also replace tenses when appropriate, e.g. *have not waited* → *did not wait*, or *wrote* → *had written*.

4.4 Paraphrasing

Although it is not our primary goal, monolingual QuickEdit can also be used for paraphrasing by pairing it with another model to automatically generate change markers. In that case, the generative model of edit markers replaces the human instructions. Basically, given an input sentence x, the edit model generate a sequence c of binary variables, which indicates whether each word x_i of x should

input	And while ~~the members of~~ Congress ~~cannot~~ agree on whether to ~~continue~~, several ~~States have~~ not ~~waited~~.
output	And while **there is no way for** Congress to agree on whether to **go ahead**, several **states did** not **wait**.
input	This is ~~truly essential~~ for our nation.
output	This is **really vital** for our nation.
input	His case ~~came after~~ that of Corporal Glen Kirkland, who ~~told~~ a parliamentary ~~committee~~ last month that he had been ~~pushed out~~ before ~~being~~ ready because he did not meet the universality of service rule.
output	His case **followed** that of Corporal Glen Kirkland, who **said to** a parliamentary **panel** last month that he had been **forced to go** before he **was** ready because he did not meet the rule of universality of service.
input	Since the ~~beginning~~ of major ~~fighting~~ in Afghanistan, the army has ~~been struggling~~ to determine what latitude it can ~~grant~~ to injured soldiers who want to ~~remain~~ in the ~~ranks~~, but who are not ~~fit for battle~~.
output	Since the **start** of major **battles** in Afghanistan, the army has **had a hard time** to determine what latitude it can **give** to injured soldiers who want to **stay** in the army, but who are not **capable** of **battling**.
input	~~Mr.~~ Snowden ~~wrote~~ in his letter that he had ~~been subjected to~~ a ~~serious~~ and sustained campaign of persecution ~~,~~ ~~which~~ forced him ~~to leave~~ his ~~country~~.
output	**Mr** Snowden had **written** in his letter that he had **suffered** a **severe** and sustained campaign of persecution that forced him **out** of his **homeland**.
input	Spirit Airlines Inc. ~~applied~~ the first ~~hand baggage charges~~ three years ago, and ~~low-cost~~ Allegiant followed ~~a little~~ later.
output	Spirit Airlines Inc. **introduced** the first **hand-luggage charge** three years ago, and the **inexpensive** Allegiant followed **somewhat** later.
input	"I~~'ve never~~ seen ~~such a fluid~~ boarding ~~procedure~~ in my ~~entire~~ career"; he ~~says~~.
output	"I **have not** seen **this kind of seamless** boarding in my career"; he **said**.
input	~~As a result ,~~ there ~~will~~ be ~~no more~~ employees in the ~~plant~~.
output	**This means that** there **won't** be **any** employees in the **factory**.
input	Pierre Beaudoin ~~,~~ ~~President and CEO ,~~ is confident ~~that~~ Bombardier ~~will~~ meet its target ~~of~~ 300 firm ~~orders before~~ the first ~~aircraft enters~~ commercial ~~service~~.
output	**Chief Executive Officer** Pierre Beaudoin is confident Bombardier **can** meet its 300 firm **order** target **prior to** the first **airplane entering** commercial **services**.
input	~~Another 35 persons~~ involved in ~~trafficking~~ were ~~sentenced to~~ a total of 153 years~~'~~ ~~imprisonment~~ for ~~drug trafficking~~.
output	**Thirty-five other people** involved in **the traffic** were **punished with** a total of 153 years in **prison** for **drug-related offenses**.

Table 3: Monolingual editing examples from the WMT'14 fr-en test set. Examples originate from news sentences of the newstest2014 dataset. Strike-through text indicates the tokens marked to be changed. Bold text indicates tokens introduced by the model, i.e. tokens not present in the original guess.

	Accuracy	Fluency	Boldness
Source	100%	100%	0%
ParaNet	56%	37%	16%
QuickEdit	72%	53%	21%

Table 4: Paraphrasing experiments on the MTC dataset.

be edited out ($c_i = 1$) or not ($c_i = 0$). QuickEdit then takes (x, c) and generate a sentence y that paraphrases x following the change markers c.

We use the monolingual QuickEdit model for English trained on WMT-14 fr-en for our paraphrase experiments. We rely on the simplest possible model to generate change markers: for each word type w, we estimate its probability to be

reference	He said that Sino-Kenyan news agencies had long-term cooperative ties and hoped that the ties could further develop in the new century.
human	He said the two News Agencies of China and Kenya have friendly relationship over a long period of time. He hoped that this relation could further develop in the new century.
paranet	He said the two **news outlets** in China and Kenya have **amicably similar relationships to** a long period of time.
QuickEdit	He said that the two **news agencies** of China and Kenya **were friends for** a long period of **time** and hoped that the relationship **would continue** in the new century.
reference	Annan urged sharon to ensure israeli forces will "adopt military tactic and weapons that cause a minimum possible threat to safety of palestinian people and personal properties. "
human	Annan called on Sharon to ensure that Israeli security forces " use weapons and fighting methods that will cause minimum threat to the safety and property of the Palestinian civilians. "
paranet	Annan called **for** Sharon to " ensure that Israeli security forces **deploy** weapons and **combat** methods that **endanger** security and the property of Palestinian civilians. "
QuickEdit	Annan **calls** on Sharon to " use weapons and **combat practices** that will **pose a** minimum threat to the safety and property of Palestinian civilians. "
reference	[Shuttleworth']s space travel has drawn great publicity in South Africa and won the honor of being the most important news event since Mandela's release from prison.
human	Shuttleworth's space journey has received enormous attention in South Africa and is praised as the most important news since the release of Nelson Mandela from prison.
paranet	Shuttleworth's journey has received enormous attention in South Africa and is **considered** the most important news since the release of Nelson Mandela.
QuickEdit	**The** Shuttleworth space **trip attracted considerable** attention in South Africa and is **lauded** as the most important news since Nelson Mandela **was released** from **jail**.

Table 5: Paraphrasing experiments on news data from the MTC dataset. Bold indicates tokens introduced by the the models, i.e. tokens which are not in the human source given as input.

edited out $P(c_i = 1|x_i = w)$ on the QuickEdit training data based on relative frequency counts. For inference, we simply threshold this probability $P(c_i = 1|x_i = w) > \tau$ to assign change markers. τ is selected to control how *bold* paraphrasing should be, i.e. large τ would yield minor changes, while small τ would edit the input sentence substantially.

We compare our paraphrasing approach with ParaNet (Mallinson et al., 2017), a paraphrasing neural model based on translation pivoting[5]. We conduct our evaluation on the MTC dataset (Huang et al., 2002) following the setup introduced in the ParaNet paper. This setup consists of 75 human paraphrase pairs (excluding duplicate MTC sentences as well as erroneous paraphrases). The evaluation considers each pair of human paraphrases (x, y). Each paraphrasing model (QuickEdit and ParaNet) generates a paraphrase given x. Then human judgments are collected by showing y and three versions of x, i.e. the original version x, its paraphrase from ParaNet $x^{(p)}$ and its paraphrase from QuickEdit $x^{(q)}$. For each example, the three sentences $x, x^{(p)}, x^{(q)}$ are shuffled and do not carry any information about their origin. The assessor should label whether each version of x is a valid paraphrase of y and should rank them by fluency from 1 most fluent to 3 least fluent.

We can evaluate paraphrasing performance at various levels of boldness which we control with the parameter τ. Bold paraphrasing means that the model needs to generate sentences which differ more from the input x than conservative paraphrasing. In this work, our evaluation relies on a level of boldness comparable to ParaNet

[5]We are thankful to the authors of ParaNet for sharing their generations for our evaluation.

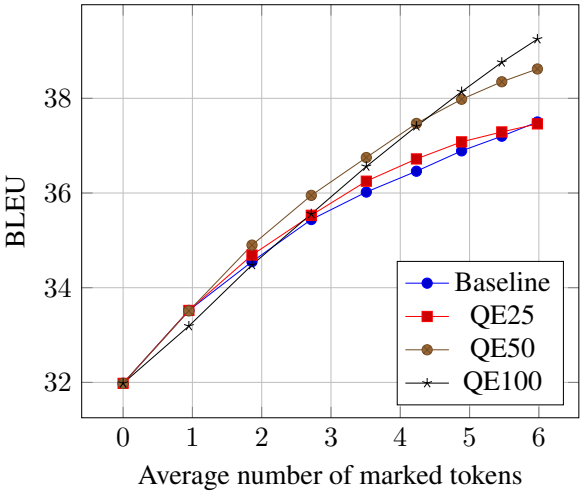


Figure 2: Post-editing results as a function of the average number of marked tokens per sentence on WMT'14 de-en validation set (45k sentences). QE25, QE50, QE100 refer to QuickEdit models trained with data where respectively 25, 50 or 100% of the guess tokens not present in the reference were marked to be changed.

from (Mallinson et al., 2017). Table 4 reports the results of this experiment. Accuracy measures the fraction of sentences considered valid paraphrases. Fluency measures the number of cases the paraphrase was considered more fluent or as fluent as the source sentence. Boldness measure the fraction of paraphrase tokens that were not in the source.

The results highlight the advantages of QuickEdit. The paraphrases from QuickEdit are accurate for 72% of the sentences versus 56% for ParaNet. The fluency of the generation from QuickEdit ranks equally or higher than the human source sentence for 53% of the examples, which compares to 37% for ParaNet. Table 5 shows a few paraphrases from both models. These examples highlight that the boldness operating point chosen by the authors of ParaNet is rather conservative, with few edits per sentence. Nevertheless, QuickEdit advantage is clear, showing that ParaNet often forgets part of the source sentence while QuickEdit does not, e.g. *could futher develop* in the first example is not expressed by ParaNet but QuickEdit proposes *would continue*. This tendency to shorten the input can yield an opposite meaning, e.g. in the second example, ParaNet rephrases *cause minimum threat* as *endanger* while QuickEdit proposes correctly *pose a minimum threat*. Examples with less conservative paraphrasing are shown in Table 3.

5 Conclusions

This work proposes QuickEdit, a neural sequence to sequence model that allows one to edit text by simply requesting few initial tokens to be changed. From a marked sentence, the model can generate an edited sentence both in the context of machine translation post-editing (a source sentence is also provided), or in a monolingual setting. In both cases, we assess the impact of the change requests. We show that marking words not present in a hidden reference sentence allow the model to generate text closer to this reference. In the context of post-editing, we conduct simulated post-edits, i.e. we mark words absent from the reference as rejected. We show that crossing out a few words per sentence can drastically improve BLEU, even on top of a strong MT system, e.g. BLEU on WMT'14-en-fr moves from 40.2 to 53.4 with QuickEdit post-editing as opposed to 47.8 for the post-editing baseline. In the context of monolingual editing, we show that our system both allow text editing and paraphrasing. For paraphrasing, we outperform a strong model (Mallinson et al., 2017) in a human evaluation on the MTC dataset, both in terms of accuracy (72% vs 53%) and fluency of the generation (53% vs 37%).

Our work opens several future directions of research. First, we want to extend our evaluation from simulated post-edits to a genuine interactive editing scenario. QuickEdit currently allows only to reject word forms for a whole sentence, not reject them in a specific context. We plan to explore this possibility. Also, QuickEdit could be a good basis for an automatic post-editing system (Chatterjee et al., 2015). QuickEdit can be applied for multi-step editing, letting the user refine their sentence multiple time. In that case, attending to all previous versions of the sentence would be relevant. Finally, we could also consider offering a richer set of simple edit actions. For instance, we could propose span substitutions to the user, which requires a decoding stage proposing a short list of promising spans and candidate replacements.

Acknowledgments

We thank Marc'Aurelio Ranzato, Sumit Chopra, Roman Novak for helpful discussions. We thank Sergey Edunov, Sam Gross, Myle Ott for writing the fairseq-py toolkit used in our experiments. We thank Jonathan Mallinson, Rico Sennrich, Mirella Lapata, for sharing ParaNet data.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 597–604.

Sergio Barrachina, Oliver Bender, Francisco Casacuberta, Jorge Civera, Elsa Cubel, Shahram Khadivi, Antonio Lagarda, Hermann Ney, Jesús Tomás, Enrique Vidal, and Juan-Miguel Vilar. 2009. Statistical approaches to computer-assisted translation. *Computational Linguistics* 35(1):3–28.

Steffen Bickel, Peter Haider, and Tobias Scheffer. 2005. Predicting sentences using n-gram language models. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 193–200.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*.

Léon Bottou. 1991. Stochastic gradient learning in neural networks. In *Proceedings of Neuro-Nîmes 91*. EC2, Nimes, France.

Eric Brill and Robert C Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 286–293.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th IWSLT evaluation campaign. In *Proceedings of the International Workshop on Spoken Language Translation, Hanoi, Vietnam*.

Rajen Chatterjee, Marion Weller, Matteo Negri, and Marco Turchi. 2015. Exploring the planet of the APEs: a comparative study of state-of-the-art methods for MT automatic post-editing. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. pages 156–161.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

Katja Filippova, Enrique Alfonseca, Carlos Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP'15)*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. 2017. Convolutional sequence to sequence learning .

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

Spence Green, Sida I Wang, Jason Chuang, Jeffrey Heer, Sebastian Schuster, and Christopher D Manning. 2014. Human effort and machine learnability in computer aided translation. In *EMNLP*. pages 1225–1236.

Kelvin Guu, Tatsunori B Hashimoto, Yonatan Oren, and Percy Liang. 2017. Generating sentences by editing prototypes. *arXiv preprint arXiv:1709.08878* .

Shudong Huang, David Graff, and George Doddington. 2002. *Multiple-translation Chinese corpus*. Linguistic Data Consortium, University of Pennsylvania.

Edgar T. Irons and Frans M. Djorup. 1972. A crt editing system. *Communications of the ACM* 15(1):16–20.

Philipp Koehn. 2009a. *Statistical machine translation*. Cambridge University Press.

Philipp Koehn. 2009b. A web-based interactive computer aided translation tool. In *Proceedings of the ACL-IJCNLP 2009 Software Demonstrations*. Association for Computational Linguistics, pages 17–20.

A-L Lagarda, Vicente Alabau, Francisco Casacuberta, Roberto Silva, and Enrique Diaz-de Liano. 2009. Statistical post-editing of a rule-based machine translation system. In *North American Chapter of the Association for Computational Linguistics (NAACL)*. Association for Computational Linguistics, pages 217–220.

Philippe Langlais, George Foster, and Guy Lapalme. 2000. Transtype: a computer-aided translation typing system. In *Proceedings of the 2000 NAACL-ANLP Workshop on Embedded machine translation systems-Volume 5*. Association for Computational Linguistics, pages 46–51.

Jindřich Libovickỳ and Jindřich Helcl. 2017. Attention strategies for multi-source sequence-to-sequence learning. *arXiv preprint arXiv:1704.06567* .

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* .

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2017. Paraphrasing revisited with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. volume 1, pages 881–893.

Benjamin Marie and Aurélien Max. 2015. Touch-based pre-post-editing of machine translation output. In *EMNLP*. pages 1040–1045.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*. volume 2, page 3.

Yurii Nesterov. 1983. A method of solving a convex programming problem with convergence rate o (1/k2). *Soviet Mathematics Doklady* 27(2).

Sergiu Nisioi, Sanja Štajner, Simone Paolo Ponzetto, and Liviu P Dinu. 2017. Exploring neural text simplification models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. volume 2, pages 85–91.

Roman Novak, Michael Auli, and David Grangier. 2016. Iterative refinement for machine translation .

Álvaro Peris, Miguel Domingo, and Francisco Casacuberta. 2017. Interactive neural machine translation. *Computer Speech & Language* 45:201–220.

Chris Quirk, Chris Brockett, and William Dolan. 2004. Monolingual machine translation for paraphrase generation. In *Proceedings of the 2004 conference on empirical methods in natural language processing*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685* .

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Edinburgh Neural Machine Translation Systems for WMT 16. In *Proc. of WMT*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural Machine Translation of Rare Words with Subword Units. In *Proc. of ACL*.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*. pages 1139–1147.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* .

John Wieting and Kevin Gimpel. 2017. Pushing the Limits of Paraphrastic Sentence Embeddings with Millions of Machine Translations. *arXiv:1711.05732* .

Tempo-Lexical Context driven Word Embedding for Cross-Session Search Task Extraction

Procheta Sen
ADAPT Centre
School of Computing
Dublin City University
Dublin 9, Ireland
procheta.sen2@mail.dcu.ie

Debasis Ganguly
IBM Research Labs
Dublin, Ireland
debasis.ganguly1@ie.ibm.com

Gareth J.F. Jones
School of Computing
ADAPT Centre
Dublin City University
Dublin 9, Ireland
Gareth.Jones@dcu.ie

Abstract

Search task extraction in information retrieval is the process of identifying search intents over a set of queries relating to the same topical information need. Search tasks may potentially span across multiple search sessions. Most existing research on search task extraction has focused on identifying tasks within a single session, where the notion of a session is defined by a fixed length time window. By contrast, in this work we seek to identify tasks that span across multiple sessions. To identify tasks, we conduct a global analysis of a query log in its entirety without restricting analysis to individual temporal windows. To capture inherent task semantics, we represent queries as vectors in an abstract space. We learn the embedding of query words in this space by leveraging the temporal and lexical contexts of queries. To evaluate the effectiveness of the proposed query embedding, we conduct experiments of clustering queries into tasks with a particular interest of measuring the cross-session search task recall. Results of our experiments demonstrate that task extraction effectiveness, including cross-session recall, is improved significantly with the help of our proposed method of embedding the query terms by leveraging the temporal and temp-lexical contexts of queries.

1 Introduction

A complex search task is defined as a "a multi-aspect or a multi-step information need consisting of a set of related subtasks, each of which might recursively be complex" (Hassan Awadallah et al., 2014). For example, a task of making arrangements for travel to a conference qualifies as a complex search task because there are several choices that a user needs to make in order to plan his entire trip, e.g. selecting flight, hotel, making arrangements for local transport, finding the conference venue, finding good places to eat around, finding local sight-seeing options after the conference etc. All these sub-tasks are likely to take place within their own search sessions, where a session is defined as a set comprised of queries executed during a time period of a specific length, usually about half-an-hour (Lucchese et al., 2013).

In this paper, we address the problem of automatically predicting whether search sessions, focused on specific activities, are a part of a broader complex search task, which we refer to as the *cross-session search task extraction problem*. Cross-session search task extraction can potentially find applications in designing more proactive search engines, which may suggest relevant information about specific subsequent subtasks along a timeline, e.g. suggesting places to eat around a conference venue without the user needing to execute these queries.

To see why cross-session search task extraction is a challenging problem, firstly, note that it is likely that a query session for flight booking and one for local sightseeing around a conference venue may be far apart in time, as a result of which simple approaches of grouping queries by their timestamps, e.g. (Lucchese et al., 2013), are not likely to yield satisfactory outcomes. Secondly, the term overlap between the queries of these two sessions is also likely to be low, indicating that using lexical similarity for clustering cross-session queries into a single group, e.g. (Lucchese et al., 2013; Wang et al., 2013), is unlikely to be effective.

As an illustrative example of term mismatch, consider the two queries 'Eric Harris', 'Reb Vodka' from the AOL query log[1]. Although these two queries do not share any common terms between them, they refer to the task of finding infor-

[1] https://archive.org/download/AOL_search_data_leak_2006

Proceedings of NAACL-HLT 2018, pages 283–292
New Orleans, Louisiana, June 1 - 6, 2018.

mation on the Columbine high school massacre, the first query referring to the name of the first murderer while the second one refers to their nickname.

Our Contributions. To alleviate the identified problems with attempting to group queries by their timestamps or lexical similarities, we propose to embed queries in a task-based semantic space in a manner that will give two similar queries in this space a high likelihood of pertaining to the same underlying task. Word embedding algorithms, such as 'word2vec' (Mikolov et al., 2013), make use of the lexical context in learning vector representations of words. We propose to *transform* these word vectors into a task-oriented semantic space with the objective of making two words that are likely to be a part of the same search task closer to each other.

To learn the transformation function, we make use of average session duration and lexical similarities between within-session queries. Our method thus provides a unifying framework for addressing tempo-lexical similarity, in contrast to previous approaches that treat these two separately. Another important contribution of our proposed method is that we are able to empirically demonstrate that our proposed method is more effective than existing algorithms (Lucchese et al., 2013; Mehrotra and Yilmaz, 2017) in extracting cross-session search tasks without the application of any external information for estimating task relatedness. For instance, the work in (Lucchese et al., 2013) relies on Wikipedia to contextualize queries, while the one in (Verma and Yilmaz, 2014) uses Wikipedia-based entity recognition to estimate task relatedness.

The rest of the paper is organized as follows. In Section 2 we overview previous work in task extraction and query embedding. In Section 3, we introduce our semantic context driven transformation-based word vector embedding algorithm to enhance cross-session query similarity matching. Section 4 then describes how the transformed query vectors are clustered into search tasks. Section 5 describes our experimental setup. Section 6 presents the results of our experiments. Section 7 concludes the paper with suggestions for future work.

2 Related Work

In this section we review existing work in search task extraction and query embedding and contrast this with the method introduced in this paper. We look first at work on unsupervised task extraction and then consider work on supervised methods, and final briefly consider a study introducing a query embedding method.

2.1 Unsupervised Task Extraction

A method for extracting tasks within each search session is proposed in (Lucchese et al., 2013). A session is defined with fixed length time windows. After investigating a wide range of time length values, the optimum is reported to be 26 minutes, which is what we also use in our work. The study reported in (Lucchese et al., 2013) also investigated a number of clustering techniques to group together related queries from each session into tasks. A wide range of features were investigated to define the similarity between a pair of queries, e.g. edit distance, cosine-similarity and Jaccard coefficient of character level trigrams. In contrast to (Lucchese et al., 2013; Wang et al., 2013), we investigate the use of embedded query vectors to compute similarity, rather than depending on character and word level lexical similarity features, e.g. edit distance, term overlap, trigram character overlap etc. Another difference of our method from (Lucchese et al., 2013) is that instead of restricting clustering to each session, we cluster the entire dataset globally, which implies that our method is not limited by variations in session duration. We also evaluate the effectiveness of clustering the entire dataset rather than on aggregating clustering effectiveness separately for each session as in (Lucchese et al., 2013).

Extraction of task hierarchies was investigated in (Mehrotra et al., 2016). Given a set of task related queries, they composed query vectors as weighted combinations of the constituent query term vectors, the weights being the maximum likelihood estimates from query-task relationships. A Chinese Restaurant Process (CRP) based posterior inference process was then used to extract the tasks from individual queries. In an extension of this work (Mehrotra and Yilmaz, 2017), the authors proposed a Bayesian non-parametric approach for extracting task hierarchies. The main difference between our approach and (Mehrotra et al., 2016; Mehrotra and Yilmaz, 2017) is that

our focus is on finding cross-session tasks from a query log, rather than finding hierarchies of tasks. Further, instead of using similarities between embedded query vectors as one of the features to estimate the relatedness between two queries, we propose a task semantics driven embedding technique to transform a query in close proximity to its task-related counterpart.

An entity extraction method was applied in (Verma and Yilmaz, 2014) to estimate similarities between queries for the purpose of task extraction. In contrast to this, our method does not rely on an entity extractor to extract cross-session tasks.

2.2 Supervised Task Extraction

A supervised approach for automatically segmenting queries into task hierarchies was proposed in (Jones and Klinkner, 2008). They trained logistic regression models to determine whether two queries belong to same task or not. According to (Wang et al., 2013), the disadvantage of using a classifier based approach for extracting tasks is that with the binary predictions of the classifier it is difficult to model the transitive task dependence between the queries, e.g. if query pairs (q_1, q_2) and (q_2, q_3) are predicted to be part of the same task, the classifier may not predict that q_1 and q_3 are also a part of the same task. Graph-based clustering on the binary adjacency matrix between query pairs (obtained from logistic regression output) is also likely to introduce noise during clustering (Wang et al., 2013). The limitations of (Jones and Klinkner, 2008) were alleviated in the work reported in (Wang et al., 2013), which employs a structural SVM framework for estimating the weights of different lexical features to measure the similarity between two queries.

The difference between the studies reported in (Jones and Klinkner, 2008; Wang et al., 2013) and our work is that we propose a completely unsupervised approach for clustering queries. This implies that our method does not rely on the availability of training data, the construction of which requires considerable manual effort.

2.3 Query Embedding

A relevance-based word embedding technique was developed in (Zamani and Croft, 2017). This method uses the top documents retrieved for each query to learn the association between the query terms and those occurring in the retrieved documents. In contrast to retrieving ranked lists for every query as in (Zamani and Croft, 2017), we capture the semantic context of query words with the help of other useful cues for task-relatedness, e.g. the time-gap between queries.

3 Embedding Query Words

'Word2vec' is a standard approach to obtain embedded words vectors (Mikolov et al., 2013). The word2vec approach aims to create similar vector representations of words that have similar context, and are thus assumed to be significantly semantically related. In this section, we explain why the standard word2vec method may not be suitable for embedding queries in an abstract space of task-semantics for the purpose of using these vectors to extract cross-session search tasks. To address this problem, we propose a method of word embedding that is able to capture larger semantic contexts for better estimation of the word vectors.

3.1 Problems with Short Documents

Let $\mathbf{w} \in \mathbb{R}^d$ denote the vector representation of a word $w \in V$, V and d being the vocabulary and the dimension of the embedded vectors, respectively. Let W be a $d \times V$ matrix, where each d dimensional column vector represents a word vector. Let D be an indicator random variable denoting semantic relatedness of a word with its context. Given a pair of words, (w, c), the probability that the word c is observed in the context of word w is given by $\sigma(\exp(-(\mathbf{w} \cdot \mathbf{c})))$. Word embedding for a given corpus is obtained by sliding a window along with its context through each word position in the corpus maximizing the objective function shown in Equation 1.

$$J(\theta) = \sum_{w_t, c_t \in D^+} \sum_{c \in c_t} log(P(D = 1 | \mathbf{w_t}, \mathbf{c_t})) - \sum_{w_t, c'_t \in D^-} \sum_{c \in c'_t} log(P(D = 1 | \mathbf{w_t}, \mathbf{c'_t})) \quad (1)$$

In Equation 1, w_t is the word in the t^{th} position in a training document corpus, c_t is the set of observed context words of word w_t within a word window, c'_t is the set of randomly sampled words from outside the context of w_t. D^+ denotes the set of all observed word-context pairs (w_t, c_t), whereas D^- consists of pairs (w_t, c'_t).

The word2vec algorithm respects document boundaries by not extending the context vector

across them. In the context of our empirical study, we aim to learn word vector embedding from a query log, where each document in the 'word2vec' terminology refers to a single query. In the case of keyword-based search queries, often comprising 2-3 words, the average number of context vectors is much lower than the number of contexts available for the standard word embedding scenario of full length documents, e.g. news articles and web pages. Consequently, this may result in ineffective estimation of the word-context semantic relations for the queries.

3.2 Word Vector Transformation with Semantic Contexts

To alleviate the problems of short contexts when embedding queries, we propose to learn a transformation matrix to transform a set of word vectors to, generally speaking, another abstract space. The aim is to transform a word vector $\mathbf{w}$ so that it is close to a set of other words that *respects* the characteristics of this abstract space. In the context of our problem, the abstract space refers to the embedding space of task-relatedness with the characteristics that queries that are a part of the same search task should be embedded close to each other.

We adopt a general terminology of referring to the desired similarity in the abstract space as *semantic similarity*, which in the context of our problem refers to task-relatedness and is not to be confused with linguistic semantics. Formally, the set of words similar to the word w is represented by the set $\Phi(w)$ shown in Equation 2,

$$\Phi(w) = \{v : (w, v) \in S\}, \tag{2}$$

where S denotes the semantic relation between a pair of words. In particular, the set $\Phi(w)$ depends on the definition of the semantic relation S between two words, which we will describe in Section 3.3.

Assuming the existence of a pre-defined semantic relation S between word pairs, we define the loss function for a word vector $\mathbf{w}$ as shown in Equation 3.

$$l(\mathbf{w};\theta) = \sum_{\mathbf{v}:v\in\Phi(w)} \sum_{\mathbf{u}:u\notin\Phi(w)} \max\big(0, m - ((\theta\mathbf{w})^T\mathbf{v} - (\theta\mathbf{w})^T\mathbf{u})\big) \tag{3}$$

Equation 3 defines a hinge loss function with margin m (set to 1 in our experiments). The loss function is parameterized by the transformation matrix $\theta \in \mathbb{R}^{d\times d}$, and is learned by iterating with stochastic gradient descent. The word vectors used in learning the parameter matrix θ are obtained by the word2vec skip-gram algorithm. After training, each word vector $\mathbf{w}$ is transformed to $\mathbf{w}'$ in Equation 4.

$$\mathbf{w}' = \theta \cdot \mathbf{w} \tag{4}$$

Informally speaking, the objective function aims to maximize the similarity between two word vectors $\mathbf{w}$ and $\mathbf{v}$ that are members of the same semantic context. On the other hand, it minimizes the similarity between the word vector $\mathbf{w}$ and a word vector $\mathbf{u}$ randomly sampled from outside its context, as defined by the semantic relation S of Equation 2. In principle, the objective function of Equation 3 is similar to the word2vec objective function of Equation 1, the difference being in the definition of the context vector. While the word2vec algorithm relies on an adjacent sequence of words to define a context, in our proposed approach, we rely on a pre-defined set of binary relations between words.

Another analogy of Equation 3 can be drawn with the multi-modal embedding loss function proposed in (Frome et al., 2013), where the words from the caption of an image constitute the notion of the 'semantic context' of the image vector used to transform it. For our problem, we make use of this context to associate the task-specific relationship between query words.

3.3 Temporal Semantic Context

In the particular context of query logs, temporal similarity is likely to play an important role in topically grouping queries. This is because queries in the same search session are usually related to the same topic, as observed in previous studies (Lucchese et al., 2013; Wang et al., 2013). For example, it can be observed from the AOL query log that the words 'reb' and 'vodka' belong to the same search session as the words 'eric' and 'harris' (see example in Section 1). In this case, the semantic relationship S, as described in Section 3.2, considers terms u (e.g. 'vodka') and v (e.g. 'harris') from the same query session to be semantically related. To define the semantic relation S, we take into account a temporal context specified by a time window of 26 minutes as reported in (Lucchese et al., 2013). Specifically, if two queries belong to the same search session, as defined by a fixed length time window, then each

constituent word pair within them is considered to be members of the set S.

3.4 Tempo-Lexical Semantic Context

In real-life settings, even within a session of a specified time length, users often multi-task their activities (possibly by using multiple browser tabs or windows) (Lucchese et al., 2013; Wang et al., 2013; Mehrotra and Yilmaz, 2017). To address this issue, we further cluster the queries of each search session into mutually disjoint groups. Our hypothesis is that this grouping of the queries of a single search session into multiple clusters may improve word embedding of the query terms further by restricting the semantic relationship S (Equation 2) to consider terms from related queries within each cluster separately.

Our clustering approach is based on a weighted graph of query similarities computed by a linear combination of content-based similarity (Sim_r) and retrieved document list based similarity (Sim_r) as described in Section 4 and Section 5.3. The clusters then provide the tempo-lexical contexts which are subsequently used to improve the quality of embedding of the query words.

4 Clustering of Embedded Queries

In this section, we describe our unsupervised approach to identifying cross-session tasks by clustering the query vectors, where the constituent query word vectors are obtained using the word embedding approaches described in Section 3.

4.1 Query Vector Embedding

We hypothesize that the modified word vector embedding approach of Section 3.2 will be more effective in capturing the session specific semantics of query terms since it takes into account the temporal context of query session information from query logs. We adopt a standard word vector combination method to form embedded query vectors. Because of the compositionality property of word vectors (Mikolov et al., 2013), the simple method of averaging over the constituent word vectors has been reported to work well for various tasks such as term re-weighting and query reformulation (Zheng and Callan, 2015; Grbovic et al., 2015).

4.2 Clustering of Query Vectors

Unlike previous approaches of grouping together queries according to fixed time windows, and then clustering the queries within each time window separately (Lucchese et al., 2013; Wang et al., 2013), we take a more general approach of clustering the overall set of query vectors.

Since the number of query clusters cannot be known a priori, the number of clusters is estimated by adopting a clustering approach that does not require the number of clusters to be specified. We adopt the best performing clustering method identified in (Lucchese et al., 2013) referred to as QC-$_{WCC}$. This is a graph based clustering algorithm that extracts the weighted connected components of a graph after constructing a complete graph and then pruning off the edges that are below a predefined threshold, η.

In QC-$_{WCC}$, the weights between the graph edges are defined by a linear combination of two types of similarities: i) content-based (Sim_c), and ii) retrieval based (Sim_r), as shown in Equation 5, in which the overall similarity is controlled by the linear combination parameter α.

$$Sim(q_i, q_j) = \alpha Sim_c(q_i, q_j) + (1 - \alpha) Sim_r(q_i, q_j) \quad (5)$$

- *Content-based* similarity: Measured with the help of character trigrams and normalized Levenshtein similarity between query pairs.
- *Retrieval-based* similarity: Each query is contextualized with a Wikipedia collection. More specifically, two queries are considered similar if the top 1000 documents retrieved by them are also similar.

In contrast to the experimental setup of (Lucchese et al., 2013), a) we conduct clustering globally instead of clustering each individual query session separately; and b) the edge weights of the graph-based clustering in our case refers to the cosine-similarity values computed between the embedded query vectors and cosine similarity values between the vectors obtained from top 1000 documents retrieved from Clueweb12B, a publicly available web collection[2].

5 Experimental Setup

In this section we describe the setup for our experimental study. We begin with an overview of our datasets, then introduce the experimental baselines used and the objectives of our experiments, finally

[2] http://boston.lti.cs.cmu.edu/clueweb12/

we set out the parameter settings used in our experiments.

5.1 Dataset

Similar to previous reported studies (Lucchese et al., 2013; Mehrotra and Yilmaz, 2017; Mehrotra et al., 2016; Verma and Yilmaz, 2014), we use the AOL query log for our experiments. In order to compare our results with these studies, we use the same subset of 1424 queries from the AOL query log for the evaluation of task extraction effectiveness as used in these earlier studies. However, since the purpose of these studies was only to extract tasks from a single session, in its annotation scheme two queries only qualified as part of the same task if they appeared within the same session.

In contrast, since we investigate cross-session task extraction, the time length threshold is not applicable to our annotation scheme, as a result of which, we re-annotated the task labelled dataset of (Lucchese et al., 2013). In particular, our annotation scheme was solely based on the underlying search intent of the query.

While re-annotating the dataset of (Lucchese et al., 2013), the annotators were instructed not to change the task labels within each session. Instead, the annotators were asked to re-label task identifiers spanning across different query sessions. For example, the annotation of (Lucchese et al., 2013) considered 'robert f kennedy jr' and 'robert francis kennedy' to belong to two different tasks since these queries were executed during different sessions. However, our annotation scheme considers them to be a part of the same task.

Two persons were employed to carry out our annotation step of the set of 1424 queries in two different batches. They were asked to come to a consensus when trying to merge the task labels across their individual batches. The annotators were instructed to use a commercial search engine (e.g. Google), if required, to determine if two queries from different search sessions could potentially relate to the same underlying task. Table 1 provides an overview of our annotated task labels; this shows that there are a considerable number of sessions that contain queries spanning across session boundaries. It can be seen from Table 1 that after post-processing the single session task labels, the total number of distinct tasks is reduced. This is indicative of the fact that the modified dataset is able to consider queries from different search sessions as a part of the same search task (there are 36,768 of them as shown in Table 1). The post-processed dataset with cross-session task labels that we use for our experiments is publicly available[3].

	Task label granularity	
Item	Within-session	Cross-session
#Queries	1424	1424
#Tasks annotated	554	224
#Sessions	307	307
#Sessions with cross-session tasks	0	239
#Query pairs across sessions judged in the same task	0	36768

Table 1: Dataset statistics of task annotated queries from the AOL query log. Cross-session task labels are post-processed annotations of the dataset prepared by (Lucchese et al., 2013).

5.2 Baselines and Experiment Objectives

Since our proposed task extraction method is unsupervised, for a fair comparison we only employ unsupervised approaches as baselines. More specifically, we did not consider the supervised approaches reported in (Jones and Klinkner, 2008; Wang et al., 2013) as our baselines.

As our first baseline, we re-implemented QC-WCC, the best performing approach (Lucchese et al., 2013) (briefly described in Section 4.2). This study investigated a wide range of features, clustering methods and parameter settings. We adopt the same linear combination of similarities in our study as shown in Equation 5.

Our re-implementation of this work involves a slight change to the original one. Instead of using a Wikipedia document collection, we employ a much larger collection of crawled web documents, namely the ClueWeb12B collection, comprising of nearly $52M$ documents[4]. Our reasons for using the ClueWeb collection are as follows. Firstly, our study is carried out using queries from a Web search log and hence it is reasonable to expect that a web collection will provide better estimates of semantic similarities between the queries. Secondly, a number of our queries in our dataset are not of expository type, and hence the num-

[3] https://github.com/procheta/AOLTaskExtraction/blob/master/Task.csv

[4] http://boston.lti.cs.cmu.edu/clueweb12/

ber of matching Wikipedia articles is expected to be low for them due to vocabulary mismatch. On the other hand, the web collection, being diverse, is expected to retrieve more matching articles for these types of queries. To compare the performance of our implementation of QC-$_{WCC}$ with ClueWeb12B with the original one, we adopted an experimental and evaluation setup identical to that of (Lucchese et al., 2013), the only difference being in the collection used for deriving the semantic similarities. For the retrieval model we used the LM-JM (Language Model with Jelineck-Mercer smoothing) with the smoothing parameter set to 0.6 as suggested in (Lavrenko and Croft, 2001).

To demonstrate the potential benefits of our proposed tempo and tempo-lexical context-driven word embedding based approaches for the query terms, we employed the following three baselines.

1. **Qry vec skip-gram**: In this approach, query vectors were obtained by summing over the constituent word vectors obtained using the standard skip-gram (Mikolov et al., 2013).
2. **Qry vec (All-in-one Session Context)**: We hypothesized that additional context is likely to capture task-specific semantics of the query terms. A boundary condition arises when the entire query log is assumed to belong to one session. To show that the temporal context needs to be focused, in this approach, we investigate the effect of setting the context set S to the entire vocabulary of query terms.
3. **Qry vec (Pre-trained Google news vectors)** We hypothesized that additional context is likely to be useful to learn the vector representations of constituent words of short documents (in this case, queries). To see if pre-trained word vectors from an external generic corpus can be useful to alleviate the problem of short documents, we employ pre-trained word vectors from the Google news corpus to obtain the vector representation of the queries.

The objective of the experiments is to show that our proposed query term embedding method can outperform the above mentioned baselines, thus indicating that within-session adjacency information can be useful to learn task specific semantics.

5.3 Parameters and Evaluation Metrics

Parameters. In our method, we use the cosine similarity between the embedded query vectors instead of using character 3-grams and Levenshtein similarity, as used in (Lucchese et al., 2013)), to compute $Sim_c(q_i, q_j)$ between any two query pairs q_i and q_j. We employ three different embedding strategies for our experiments: i) standard word2vec, ii) transformed vectors with temporal context, and iii) transformed vectors with tempo-lexical contexts. In all our experiments, we tune α from 0 to 1 in steps of 0.1. The second parameter common to all the methods, the threshold η, which is used in QC-$_{WCC}$ clustering to prune off edges from the weighted similarity graph between query pairs. We tuned η in the range 0.1 to 1 in steps of 0.1 for each method separately.

For our word vector based experiments, we used the skip-gram model to train the word vectors using the entire AOL query log comprising over 6M queries. The dimensionality of the word vectors was set to 200. The initially obtained word vectors were used as starting inputs to learn the temporal and tempo-lexical transformations. For the tempo-lexical based transformation method, we used the optimal value of η as obtained from the QC-$_{WCC}$ baseline, to cluster the queries in each temporal window of 26 minutes.

Evaluation Metrics. Since we use weighted clustering to extract cross-session search tasks, we used standard clustering evaluation metrics to evaluate the effectiveness of the task extraction. Clustering is typically evaluated with the effectiveness of the pair-wise decisions of assigning data points to the same or different clusters. In our case, the number of true positives was given by the number of query pairs in the ground-truth that were judged to belong to the same task and were also predicted by the system to be a part of the same task. Similarly, we computed the false positives and the true negatives. Based on these counts, we computed the standard metrics of precision, recall, and F-score (similar to (Lucchese et al., 2013)).

Additionally, to measure how many of the total number of cross-session queries that were part of the same search tasks were discovered by these approaches, we computed the cross-session recall (denoted as 'CS-Recall'). This metric was computed as the ratio of the number of correctly identified cross-session similar-task query pairs against

Query Similarity	Parameters		Metrics			
	α	η	F-score	Prec	Recall	CS-Recall
QC-$_{WCC}$(3gram+ Levenestine) (Lucchese et al., 2013)	0.8	0.4	0.471	0.387	0.603	0.1930
Qry vec skip-gram	0.7	0.8	0.524*	**0.465***	0.602	0.7161*
Qry vec (All-in-one Session Context)	0.7	0.5	0.499	0.430	0.595	0.6400
Qry vec (Pre-trained Google news vectors)	0.6	0.5	0.473	0.410	0.558	0.6400
Qry vec with temporal context	1.0	0.7	0.536*†	0.461*	0.643*†	0.7393*†
Qry vec with tempo-lexical context	0.6	0.7	**0.538***†	0.441*	**0.691***†‡	**0.7395***†‡

Table 2: Comparison between the best results obtained after parameter tuning on different unsupervised approaches of task extraction. For all methods, $1-\alpha$ represents the weight of the semantic similarity estimated from ClueWeb12B. *†‡ indicates statistical significance (paired t-test with 95% confidence) with respect to (Lucchese et al., 2013), 'Qry vec skip-gram' and 'Qry vec with temporal context' respectively.

Task extraction method	Session-F-socre
QC-$_{WCC}$ on trigram+Levenshtein with Wikipedia (Lucchese et al., 2013)	0.812*
QC-$_{WCC}$ on trigram+Levenshtein with ClueWeb12B	0.834
Non-parametric clustering on average query term vectors (Mehrotra et al., 2016)	0.845†
QC-$_{WCC}$ on average of baseline skip-gram query word vectors	0.837
QC-$_{WCC}$ on transformed word vectors with temporal context	**0.847**
QC-$_{WCC}$ on transformed word vectors with tempo-lexical context	0.840

Table 3: Within-session Task Extraction effectiveness.

the total number of them (36,768 as reported in Table 1).

In order to extend evaluation of our proposed approach to within-session task extraction, for comparison with existing studies, we computed the clustering metrics for each individual session and then computed the weighted average of these values over each session as reported in (Lucchese et al., 2013; Mehrotra and Yilmaz, 2017). Although these earlier studies refer to this weighted measure as F-score, we refer to this version of F-score as 'Session-F-score'.

6 Results

In this section, we report the results of our investigations of our proposed query vector based cross-session search task extraction. We first investigate the effectiveness of our proposed approach on the cross-session search task extraction and then report and compare results with existing approaches for within-session task extraction.

6.1 Cross-Session Task Extraction

Table 2 shows the results of weighted clustering QC-$_{WCC}$ with optimal α and η settings for each individual method. It can be seen that the first baseline approach QC-$_{WCC}$ performs poorly because trigram and Levenshtein similarities lack the semantic information required to effectively cluster task-related queries into the same cluster. Clustering effectiveness improves considerably when weighted clustering is conducted using the cosine similarities between the query vectors, i.e. the 'Qry vec skip-gram' approach. This suggests that the word vectors are better able to capture the semantic relatedness between the task-related query terms.

It can be seen that using the entire query log as one context, i.e. the approach 'Qry vec (All-in-one Sesion Context)' yields worse results than the baseline skip-gram approach, which shows that a focused context is required for effectively embedding the query terms. Results with pre-trained word vectors on a large news corpora, i.e. the approach 'Qry vec (Pre-trained Google news vectors)', show that additional out-of-domain and generic context is not helpful for improving the quality of the embedded query term vectors.

Transformation of the word vectors leveraging the semantic contexts (i.e. our proposed method in Section 3) outperforms the clustering effectiveness obtained with the baseline approaches. The most important observation is that the use of temporal context in learning word vectors results in best performance for $\alpha = 1$, i.e. when no retrieval-based similarity is used (see Equation 5). This suggests that optimally trained word vectors can produce effective task clusters without the use of external collections in contextualizing the queries. The use of tempo-lexical contexts, i.e. when the semantic context used to learn the transformation matrix for the word vectors is restricted to sim-

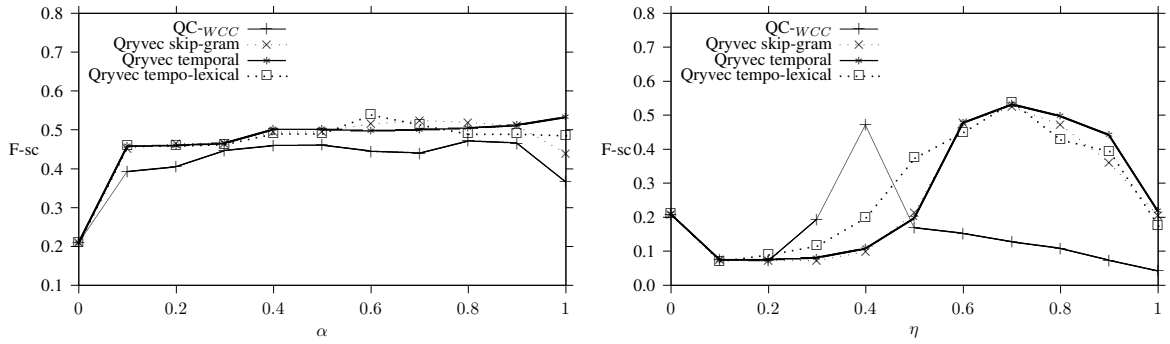


Figure 1: Sensitivity of task clustering with variations in α (left) and η (right).

ilar queries within search sessions, the effectiveness improves further. In particular, Table 2 shows that both tempo and tempo-lexical transformations are able to improve recall significantly suggesting that the transformation helps to group more truly task-related queries into the same cluster.

Next, we show the effect of varying the parameters α and η separately in Figure 1. The values of η for each corresponding method in the left graph of Figure 1 are those reported in Table 2. Similarly, for the plot on the right of Figure 1, the α values correspond to those reported in Table 2. A value of $\alpha = 1$ considers only the content based similarity (see Equation 5). It can be observed from Figure 1 (left) that at $\alpha = 1$, the F-score values for all the query embedding based approaches are higher than the baseline method of QC-$_{WCC}$. This indicates that the query embedding based approaches perform well without relying on similarity-based retrieval using an external collection. In general, it can be observed that over a wide range of α and η settings, the F-score values of the embedding based methods outperform the QC-$_{WCC}$ method.

6.2 Within-session task extraction

In this experimental setup, we make use of the session duration span of 26 minutes, similar to (Lucchese et al., 2013), to restrict query clustering to each individual session. Similar to (Lucchese et al., 2013; Mehrotra and Yilmaz, 2017), we employ the session averaged clustering metrics for measuring the effectiveness of the different approaches (see Section 5.3). We use the within-session ground-truth of (Lucchese et al., 2013) to evaluate the task extraction effectiveness.

Table 3 reports the results for various within-session task clustering approaches. The results with * and † are taken from the results reported in (Lucchese et al., 2013) and (Mehrotra et al., 2016). The following observations can be made with regard to Table 3. Firstly, the use of ClueWeb12B contributed to an improvement in task extraction effectiveness, thus demonstrating that our re-implementation of (Lucchese et al., 2013) is comparable with that of the original. Secondly, an important observation is that the use of average query term vectors along with contextual information from ClueWeb12B outperforms the approach of trigram and Levenshtein based similarity computation of (Lucchese et al., 2013). Thirdly, it can be observed that results improve with the application of transformation based word vector embedding of the query terms. The temporal context proves more effective than the tempo-lexical one.

7 Conclusions and Future Work

In this paper, we studied the problem of cross-session task extraction. We proposed a transformation based word embedding approach that takes into account the temporal and tempo-lexical contexts of queries to learn task-specific semantics. Our experiments on the AOL query log indicate that the proposed temporal and tempo-lexical query embedding method significantly outperform the baseline word2vec embedding. As future work, we would like to investigate supervised methods for cross-session task extraction.

Acknowledgement

This work was supported by Science Foundation Ireland as part of the ADAPT Centre (Grant No. 13/RC/2106) (www.adaptcentre.ie).

References

Andrea Frome, Gregory S. Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. 2013. Devise: A deep visual-semantic embedding model. In *Proc. of NIPS'13*. pages 2121–2129.

Mihajlo Grbovic, Nemanja Djuric, Vladan Radosavljevic, Fabrizio Silvestri, and Narayan Bhamidipati. 2015. Context- and content-aware embeddings for query rewriting in sponsored search. In *Proc. of SIGIR '15*. pages 383–392.

Ahmed Hassan Awadallah, Ryen W. White, Patrick Pantel, Susan T. Dumais, and Yi-Min Wang. 2014. Supporting complex search tasks. In *Proc. of CIKM'14*. pages 829–838.

Rosie Jones and Kristina Lisa Klinkner. 2008. Beyond the session timeout: Automatic hierarchical segmentation of search topics in query logs. In *Proc. of CIKM '08*. pages 699–708.

Victor Lavrenko and W. Bruce Croft. 2001. Relevance based language models. In *Proc. of SIGIR '01*. pages 120–127.

Claudio Lucchese, Salvatore Orlando, Raffaele Perego, Fabrizio Silvestri, and Gabriele Tolomei. 2013. Discovering tasks from search engine query logs. *ACM Trans. Inf. Syst.* 31(3):14:1–14:43.

Rishabh Mehrotra, Prasanta Bhattacharya, and Emine Yilmaz. 2016. Deconstructing complex search tasks: a bayesian nonparametric approach for extracting sub-tasks. In *Proc. of NAACL HLT '16*. pages 599–605.

Rishabh Mehrotra and Emine Yilmaz. 2017. Extracting hierarchies of search tasks and subtasks via a bayesian nonparametric approach. In *Proc. of SIGIR'17*. pages 285–294.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proc. NIPS '13*. pages 3111–3119.

Manisha Verma and Emine Yilmaz. 2014. Entity oriented task extraction from query logs. In *Proc. of CIKM'14*. pages 1975–1978.

Hongning Wang, Yang Song, Ming-Wei Chang, Xiaodong He, Ryen W. White, and Wei Chu. 2013. Learning to extract cross-session search tasks. In *Proc. of WWW '13*. pages 1353–1364.

Hamed Zamani and W. Bruce Croft. 2017. Relevance-based word embedding. In *Proc. of SIGIR'17*. pages 505–514.

Guoqing Zheng and Jamie Callan. 2015. Learning to reweight terms with distributed representations. In *Proc. of SIGIR '15*. pages 575–584.

Zero-shot Sequence Labeling: Transferring Knowledge from Sentences to Tokens

Marek Rei
The ALTA Institute
Computer Laboratory
University of Cambridge
United Kingdom
marek.rei@cl.cam.ac.uk

Anders Søgaard
CoAStaL DIKU
Department of Computer Science
University of Copenhagen
Denmark
soegaard@di.ku.dk

Abstract

Can attention- or gradient-based visualization techniques be used to infer token-level labels for binary sequence tagging problems, using networks trained only on sentence-level labels? We construct a neural network architecture based on soft attention, train it as a binary sentence classifier and evaluate against token-level annotation on four different datasets. Inferring token labels from a network provides a method for quantitatively evaluating what the model is learning, along with generating useful feedback in assistance systems. Our results indicate that attention-based methods are able to predict token-level labels more accurately, compared to gradient-based methods, sometimes even rivaling the supervised oracle network.

1 Introduction

Sequence labeling is a structured prediction task where systems need to assign the correct label to every token in the input sequence. Many NLP tasks, including part-of-speech tagging, named entity recognition, chunking, and error detection, are often formulated as variations of sequence labeling. Recent state-of-the-art models make use of bidirectional LSTM architectures (Irsoy and Cardie, 2014), character-based representations (Lample et al., 2016), and additional external features (Peters et al., 2017). Optimization of these models requires appropriate training data where individual tokens are manually labeled, which can be time-consuming and expensive to obtain for each different task, domain and target language.

In this paper, we investigate the task of performing sequence labeling without having access to any training data with token-level annotation. Instead of training the model directly to predict the label for each token, the model is optimized using a sentence-level objective and a modified version of the attention mechanism is then used to infer labels for individual words.

While this approach is not expected to outperform a fully supervised sequence labeling method, it opens possibilities for making use of text classification datasets where collecting token-level annotation is not possible or cost-effective.

Inferring token-level labels from a text classification network also provides a method for analyzing and interpreting the model. Previous work has used attention weights to visualize the focus of neural models in the input data. However, these analyses have largely been qualitative examinations, looking at only a few examples from the datasets. By formulating the task as a zero-shot labeling problem, we can provide quantitative evaluations of what the model is learning and where it is focusing. This will allow us to measure whether the features that the model is learning actually match our intuition, provide informative feedback to end-users, and guide our development of future model architectures.

2 Network Architecture

The main system takes as input a sentence, separated into tokens, and outputs a binary prediction as the label of the sentence. We use a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) architecture for sentence classification, with dynamic attention over words for constructing the sentence representations. Related architectures have been successful for machine translation (Bahdanau et al., 2015), sentence summarization (Rush and Weston, 2015), entailment detection (Rocktäschel et al., 2016), and error correction (Ji et al., 2017). In this work, we modify the attention mechanism and training objective in order to make the resulting network suitable for

Proceedings of NAACL-HLT 2018, pages 293–302
New Orleans, Louisiana, June 1 - 6, 2018.

also inferring binary token labels, while still performing well as a sentence classifier.

Figure 1 contains a diagram of the network architecture. The tokens are first mapped to a sequence of word representations $[w_1, w_2, w_3, ..., w_N]$, which are constructed as a combination of regular word embeddings and character-based representations, following Lample et al. (2016). These word representations are given as input to a bidirectional LSTM which iteratively passes through the sentence in both directions. Hidden representations from each direction are concatenated at every token position, resulting in vectors h_i that are focused on a specific word but take into account the context on both sides of that word. We also include a transformation with $tanh$ activation, which helps map the information from both directions into a joint feature-space:

$$\overrightarrow{h_i} = LSTM(w_i, \overrightarrow{h_{i-1}}) \quad (1)$$

$$\overleftarrow{h_i} = LSTM(w_i, \overleftarrow{h_{i+1}}) \quad (2)$$

$$\widetilde{h_i} = [\overrightarrow{h_i}; \overleftarrow{h_i}] \qquad h_i = tanh(W_h \widetilde{h_i} + b_h) \quad (3)$$

where W_h is a parameter matrix and b_h is a parameter vector, optimized during training.

Next, we include an attention mechanism that allows the network to dynamically control how much each word position contributes to the combined representation. In most attention-based systems, the attention amount is calculated in reference to some external information. For example, in machine translation the attention values are found based on a representation of the output that has already been generated (Bahdanau et al., 2015); in question answering, the attention weights are calculated in reference to the input question (Hermann et al., 2015). In our task there is no external information to be used, therefore we predict the attention values directly based on h_i, by passing it through a separate feedforward layer:

$$e_i = tanh(W_e h_i + b_e) \quad (4)$$

$$\widetilde{e_i} = W_{\widetilde{e}} e_i + b_{\widetilde{e}} \quad (5)$$

where $W_{\widetilde{e}}$, $b_{\widetilde{e}}$, W_e and b_e are trainable parameters and $\widetilde{e_i}$ results in a single scalar value. This method is equivalent to calculating the attention weights in reference to a fixed weight vector, which is optimized during training. Shen and Lee (2016) proposed an architecture for dialogue act detection where the attention values are found based on a separate set of word embeddings. We found that the method described above was consistently equivalent or better in development experiments, while requiring a smaller number of parameters.

The values of $\widetilde{e_i}$ are unrestricted and should be normalized before using them for attention, to avoid sentences of different length having representations of different magnitude. The common approach is to use an exponential function to transform the value, and then normalize by the sum of all values in the sentence:

$$a_i = \frac{exp(\widetilde{e_i})}{\sum_{k=1}^{N} exp(\widetilde{e_k})} \quad (6)$$

The value a_i is now in a range $0 \leq a_i \leq 1$ and higher values indicate that the word at position i is more important for predicting the sentence class. The network learns to predict informative values for a_i based only on the sentence objective, without receiving token-level supervision. Therefore, we can use these attention values at each token in order to infer an unsupervised sequence labeling output.

The method in Equation 6 is well-suited for applications such as machine translation – the exponential function encourages the attention to prioritize only one word in the sentence, resulting in a word-word alignment. However, the same function is less suitable for our task of unsupervised sequence labeling, as there is no reason to assume that exactly one word has a positive label. An input sentence can contain more than one tagged token, or it can contain no tokens of interest, and this should be reflected in the predictions.

Instead of the exponential function, we make use of the logistic function σ for calculating soft attention weights:

$$\widetilde{a_i} = \sigma(\widetilde{e_i}) \qquad a_i = \frac{\widetilde{a_i}}{\sum_{k=1}^{N} \widetilde{a_k}} \quad (7)$$

where each $\widetilde{a_i}$ has an individual value in the range $0 \leq \widetilde{a_i} \leq 1$ and a_i is normalized to sum up to 1 over all values in the sentence. The normalized weights a_i are used for combining the context-conditioned hidden representations from Equation

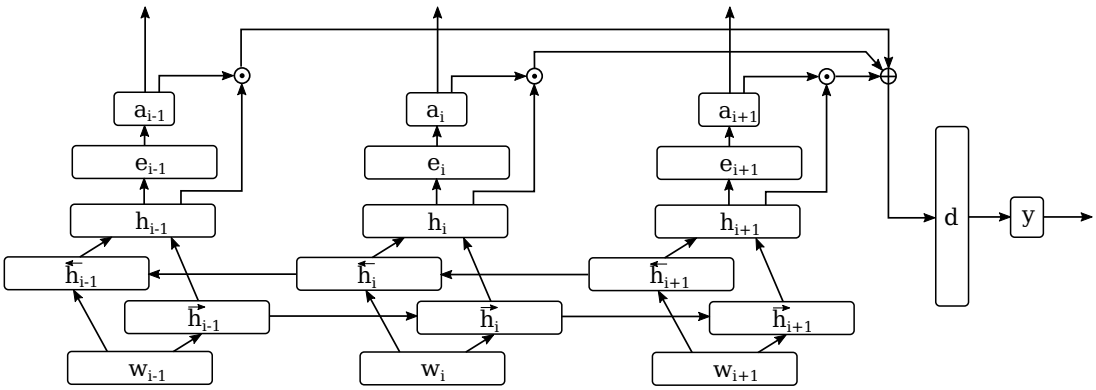


Figure 1: The neural network architecture for zero-shot sequence labeling. The soft attention values a_i are used for weighting hidden representations h_i as well as providing a binary label for each token. The network is only optimized through the sentence classification objective, predicting the sentence-level label y.

3 into a single sentence representation:

$$c = \sum_{i=1}^{N} a_i h_i \tag{8}$$

In addition, we can use the pre-normalization value $\widetilde{a_i}$ as a score for sequence labeling, with a natural decision boundary of 0.5 – higher values indicate that the token at position i is important and should be labeled positive, whereas lower values suggest the token is largely ignored for sentence classification and can receive a negative label. Attention weights with sigmoid activation have been shown to also improve performance on classification tasks (Shen and Lee, 2016), which indicates that this architecture has the benefit of being both accurate and interpretable on the token level.

Finally, we pass the sentence representation c through a feedforward layer and predict a binary label for the overall sentence:

$$d = tanh(W_d c + b_d) \tag{9}$$

$$y = \sigma(W_y d + b_y) \tag{10}$$

where d is a sentence vector and y is a single value between $0 \leq y \leq 1$, with values higher than 0.5 indicating a positive class and lower values indicating a negative prediction.

In order to optimize the model, we use several different loss functions. The first is the squared loss which optimizes the sentence-level score prediction to match the gold label in the annotation:

$$L_1 = \sum_j (y^{(j)} - \widetilde{y}^{(j)})^2 \tag{11}$$

where $y^{(j)}$ is the predicted score for the j-th sentence, and $\widetilde{y}^{(j)}$ is the true binary label $(0, 1)$ for the j-th sentence.

In addition, we want to encourage the model to learn high-quality token-level labels as part of the attention weights. While the model does not have access to token-level annotation during training, there are two constraints that we can take advantage of:

1. Only some, but not all, tokens in the sentence can have a positive label.

2. There are positive tokens in a sentence only if the overall sentence is positive.

We can then construct loss functions that encourage the model to optimize for these constraints:

$$L_2 = \sum_j (min_i(\widetilde{a_i}) - 0)^2 \tag{12}$$

$$L_3 = \sum_j (max_i(\widetilde{a_i}) - \widetilde{y}^{(j)})^2 \tag{13}$$

where $min_i(\widetilde{a_i})$ is the minimum value of all the attention weights in the sentence and $max_i(\widetilde{a_i})$ is the corresponding maximum value. Equation 12 optimizes the minimum unnormalized attention weight in a sentence to be 0, satisfying the constraint that all tokens in a sentence should not have a positive token-level label. Equation 13 then optimizes for the maximum unnormalized attention weight in a sentence to be equal to the gold label for that sentence, which is either 0 or 1, incentivizing the network to only assign large attention

weights to tokens in positive sentences. These objectives do not provide the model with additional information, but serve to push the attention scores to a range that is suitable for binary classification.

We combine all of these loss objectives together for the main optimization function:

$$L = L_1 + \gamma(L_2 + L_3) \tag{14}$$

where γ is used to control the importance of the auxiliary objectives.

3 Alternative Methods

We compare the attention-based system for inferring sequence labeling with 3 alternative methods.

3.1 Labeling Through Backpropagation

We experiment with an alternative method for inducing token-level labels, based on visualization methods using gradient analysis. Research in computer vision has shown that interpretable visualizations of convolutional networks can be obtained by analyzing the gradient after a single backpropagation pass through the network (Zeiler and Fergus, 2014). Denil et al. (2014) extended this approach to natural language processing, in order to find and visualize the most important sentences in a text. Recent work has also used the gradient-based approach for visualizing the decisions of text classification models on the token level (Li et al., 2016; Alikaniotis et al., 2016). In this section we propose an adaptation that can be used for sequence labeling tasks.

We first perform a forward pass through the network and calculate the predicted sentence-level score y. Next, we define a pseudo-label $y^* = 0$, regardless of the true label of the sentence. We then calculate the gradient of the word representation w_i with respect to the loss function using this pseudo-label:

$$g_i = \frac{\partial L_1}{\partial w_i}\Big|_{(y^*,y)} \tag{15}$$

where L_1 is the squared loss function from Equation 11. The magnitude of g_i, $|g_i|$ can now be used as an indicator of how important that word is for the positive class. The intuition behind this approach is that the magnitude of the gradient indicates which individual words need to be changed the most in order to make the overall label of the sentence negative. These are the words that are contributing most towards the positive class and should be labeled as such individually.

An obstacle in using this score for sequence labeling comes from the fact that there is no natural decision boundary between the two classes. The magnitude of the gradient is not constrained to a specific range and can vary quite a bit depending on the sentence length and the predicted sentence-level score. In order to map this magnitude to a decision, we analyze the distribution of magnitudes in a sentence. Intuitively, we want to detect outliers – scores that are larger than expected. Therefore, we map all the magnitudes in a sentence to a Gaussian distribution and set the decision boundary at 1.5 standard deviations. Any word that has a gradient magnitude higher than that will be tagged with a positive class for sequence labeling. If all the magnitudes in a sentence are very similar, none of them will cross this threshold and therefore all words will be labeled as negative.

We calculate the gradient magnitude using the same network architecture as described in Section 2, at word representation w_i after the character-based features have been included. The attention-based architecture is not necessary for this method, therefore we also report results using a more traditional bidirectional LSTM, concatenating the last hidden states from both directions and using the result as a sentence representation for the main objective.

3.2 Relative Frequency Baseline

The system for producing token-level predictions based on sentence-level training data does not necessarily need to be a neural network. As the initial experiment, we trained a Naive Bayes classifier with n-gram features on the annotated sentences and then used it to predict a label only based on a window around the target word. However, this did not produce reliable results – since the classifier is trained on full sentences, the distribution of features is very different and does not apply to a window of only a few words.

Instead, we calculate the relative frequency of a feature occurring in a positive sentence, normalized by the overall frequency of the feature, and calculate the geometric average over all features that contain a specific word:

$$r_k = \frac{c(X_k = 1, Y = 1)}{\sum_{z \in (0,1)} c(X_k = 1, Y = z)} \tag{16}$$

$$score_i = \sqrt[|F_i|]{\prod_{k \in F_i} r_k} \quad (17)$$

where $c(X_k = 1, Y = 1)$ is the number of times feature k is present in a sentence with a positive label, F_i is the set of n-gram features present in the sentence that involve the i-th word in the sentence, and $score_i$ is the token-level score for the i-th token in the sentence. We used unigram, bigram and trigram features, with extra special tokens to mark the beginning and end of a sentence.

This method will assign a high score to tokens or token sequences that appear more often in sentences which receive a positive label. While it is not able to capture long-distance context, it can memorize important keywords from the training data, such as modal verbs for uncertainty detection or common spelling errors for grammatical error detection.

3.3 Supervised Sequence Labeling

Finally, we also report the performance of a supervised sequence labeling model on the same tasks. This serves as an indicator of an upper bound for a given dataset – how well the system is able to detect relevant tokens when directly optimized for sequence labeling and provided with token-level annotation.

We construct a bidirectional LSTM tagger, following the architectures from Irsoy and Cardie (2014), Lample et al. (2016) and Rei (2017). Character-based representations are concatenated with word embeddings, passed through a bidirectional LSTM, and the hidden states from both direction are concatenated. Based on this, a probability distribution over the possible labels is predicted and the most probable label is chosen for each word. While Lample et al. (2016) used a CRF on top of the network, we exclude it here as the token-level scores coming from that network do not necessarily reflect the individual labels, since the best label sequence is chosen globally based on the combined sentence-level score. The supervised model is optimized by minimizing cross-entropy, training directly on the token-level annotation.

4 Datasets

We evaluate the performance of zero-shot sequence labeling on 3 different datasets. In each experiment, the models are trained using only sentence-level annotation and then evaluated based on token-level annotation.

4.1 CoNLL 2010 Uncertainty Detection

The CoNLL 2010 shared task (Farkas et al., 2010) investigated the detection of uncertainty in natural language texts. The use of uncertain language (also known as hedging) is a common tool in scientific writing, allowing scientists to guide research beyond the evidence without overstating what follows from their work. Vincze et al. (2008) showed that 19.44% of sentences in the biomedical papers of the BioScope corpus contain hedge cues. Automatic detection of these cues is important for downstream tasks such as information extraction and literature curation, as typically only definite information should be extracted and curated.

The dataset is annotated for both hedge cues (keywords indicating uncertainty) and scopes (the area of the sentence where the uncertainty applies). The cues are not limited to single tokens, and can also consist of several disjoint tokens (for example, *"either ... or ..."*). An example sentence from the dataset, with bold font indicating the hedge cue and curly brackets marking the scope of uncertainty:

> Although IL-1 has been reported to contribute to Th17 differentiation in mouse and man, it remains to be determined {**whether** therapeutic targeting of IL-1 will substantially affect IL-17 in RA}.

The first subtask in CoNLL 2010 was to detect any uncertainty in a sentence by predicting a binary label. The second subtask required the detection of all the individual cue tokens and the resolution of their scope. In our experiments, we train the system to detect sentence-level uncertainty, use the architecture to infer the token-level labeling and evaluate the latter on the task of detecting uncertainty cues. Since the cues are defined as keywords that indicate uncertainty, we would expect the network to detect and prioritize attention on these tokens. We use the train/test data from the second task, which contains the token-level annotation needed for evaluation, and randomly separate 10% of the training data for development.

4.2 FCE Error Detection

Error detection is the task of identifying tokens which need to be edited in order to produce a

	CoNLL 2010					FCE				
	Sent F_1	MAP	P	R	F_1	Sent F_1	MAP	P	R	F_1
Supervised	-	96.54	78.92	79.41	79.08	-	59.13	49.15	26.96	34.76
Relative freq	-	81.78	15.94	**79.98**	26.59	-	37.75	14.37	**86.36**	24.63
LSTM-LAST-BP	84.42	77.90	7.16	66.64	12.92	85.10	46.12	**29.49**	16.07	20.80
LSTM-ATTN-BP	**84.94**	80.38	9.13	71.42	16.18	**85.14**	44.52	27.62	17.81	21.65
LSTM-ATTN-SW	**84.94**	**87.86**	**77.48**	69.54	**73.26**	**85.14**	**47.79**	28.04	29.91	**28.27**

Table 1: Results for different system configurations on the CoNLL 2010 and FCE datasets. Reporting sentence-level F_1, token-level Mean Average Precision (MAP), and token-level precision/recall/F_1.

grammatically correct sentence. The task has numerous applications for writing improvement and assessment, and recent work has focused on error detection as a supervised sequence labeling task (Rei and Yannakoudakis, 2016; Kaneko et al., 2017; Rei, 2017).

Error detection can also be performed on the sentence level – detecting whether the sentence needs to be edited or not. Andersen et al. (2013) described a practical tutoring system that provides sentence-level feedback to language learners. The 2016 shared task on Automated Evaluation of Scientific Writing (Daudaravicius et al., 2016) also required participants to return binary predictions on whether the input sentence needs to be corrected.

We evaluate our system on the First Certificate in English (FCE, Yannakoudakis et al. (2011)) dataset, containing error-annotated short essays written by language learners. While the original corpus is focused on aligned corrections, Rei and Yannakoudakis (2016) converted the dataset to a sequence labeling format, which we make use of here. An example from the dataset, with bold font indicating tokens that have been annotated as incorrect given the context:

> When the show started the person who was acting **it** was not Danny Brook and **he seemed not** to be an actor.

We train the network as a sentence-level error detection system, returning a binary label and a confidence score, and also evaluate how accurately it is able to recover the locations of individual errors on the token level.

4.3 SemEval Sentiment Detection in Twitter

SemEval has been running a series of popular shared tasks on sentiment analysis in text from social media (Nakov et al., 2013; Rosenthal et al., 2014, 2015). The competitions have included various subtasks, of which we are interested in two: Task A required the polarity detection of individual phrases in a tweet, and Task B required sentiment detection of the tweet as a whole. A single tweet could contain both positive and negative phrases, regardless of its overall polarity, and was therefore separately annotated on the tweet level.

In the following example from the dataset, negative phrases are indicated with a bold font and positive phrases are marked with italics, whereas the overall sentiment of the tweet is annotated as negative:

> They may *have* a *SuperBowl* in Dallas, but Dallas **ain't winning** a SuperBowl. **Not with that** quarterback and owner. @S4NYC @RasmussenPoll

Sentiment analysis is a three-way task, as the system needs to differentiate between positive, negative and neutral sentences. Our system relies on a binary signal, therefore we convert this dataset into two binary tasks – one aims to detect positive sentiment, the other focuses on negative sentiment. We train the system as a sentiment classifier, using the tweet-level annotation, and then evaluate the system on recovering the individual positive or negative tokens. We use the train/dev/test splits of the original SemEval 2013 Twitter dataset, which contains phrase-level sentiment annotation.

5 Implementation Details

During pre-processing, tokens are lowercased while the character-level component still retains access to the capitalization information. Word embeddings were set to size 300, pre-loaded from publicly available Glove (Pennington et al., 2014)

	SemEval Negative					SemEval Positive				
	Sent F_1	MAP	P	R	F_1	Sent F_1	MAP	P	R	F_1
Supervised	-	67.70	31.79	44.66	37.02	-	67.41	36.27	50.71	42.24
Relative freq	-	44.15	17.39	15.67	16.48	-	47.64	13.39	**54.69**	21.51
LSTM-LAST-BP	53.65	43.02	8.33	28.41	12.88	70.83	49.06	17.66	35.06	23.48
LSTM-ATTN-BP	**55.83**	50.96	11.55	**31.54**	16.90	**71.26**	53.89	23.45	34.53	27.92
LSTM-ATTN-SW	**55.83**	**54.37**	**29.41**	14.40	**19.23**	**71.26**	**56.45**	**37.19**	25.96	**30.45**

Table 2: Results for different system configurations on the SemEval Twitter sentiment dataset, separated into positive and negative sentiment detection. Reporting sentence-level F_1, token-level Mean Average Precision (MAP), and token-level precision/recall/F_1.

embeddings and fine-tuned during training. Character embeddings were set to size 100. The recurrent layers in the character-level component have hidden layers of size 100; the hidden layers $\overrightarrow{h_i}$ and $\overleftarrow{h_i}$ are size 300. The hidden combined representation h_i was set to size 200, and the attention weight layer e_i was set to size 100. Parameter γ was set to 0.01 based on development experiments.

The model was implemented using Tensorflow (Abadi et al., 2016). The network weights were randomly initialized using the uniform Glorot initialization method (Glorot and Bengio, 2010) and optimization was performed using AdaDelta (Zeiler, 2012) with learning rate 1.0. Dropout (Srivastava et al., 2014) with probability 0.5 was applied to word representations w_i and the composed representations h_i after the LSTMs. The training was performed in batches of 32 sentences. Sentence-level performance was observed on the development data and the training was stopped if performance did not improve for 7 epochs. The best overall model on the development set was then used to report performance on the test data, both for sentence classification and sequence labeling. In order to avoid random outliers, we performed each experiment with 5 random seeds and report here the averaged results.

The code used for performing these experiments is made available online.[1]

6 Evaluation

Results for the experiments are presented in Tables 1 and 2. We first report the sentence-level F-measure in order to evaluate the performance on the general text classification objective. Next, we report the Mean Average Precision (MAP) at returning the active/positive tokens. This measure rewards systems that assign higher scores to positive tokens as opposed to negative ones, evaluating this as a ranking problem. It disregards a specific classification threshold and therefore provides a more fair evaluation towards systems that could be improved simply by choosing a different decision boundary. Finally, we also report token-level precision, recall and F-measure for evaluating the accuracy of this model as a sequence labeler.[2]

We report five different system configurations: **Relative freq** is the n-gram based approach described in Section 3.2. **Supervised** is the fully supervised sequence labeling system described in Section 3.3. **LSTM-LAST-BP** is using the last hidden states from the word-level LSTMs for constructing a sentence representation, and the backpropagation-based method from Section 3.1 for inducing token labels. **LSTM-ATTN-BP** is using the attention-based network architecture together with the backpropagation-based labeling method. **LSTM-ATTN-SW** is the method described in Section 2, using soft attention weights for sequence labeling and additional objectives for optimizing the network.

The method using attention weights achieves the best performance on all datasets, compared to other methods not using token-level supervision. On the CoNLL 2010 uncertainty detection dataset the system reaches 73.26% F-score, which is 93% of the supervised upper bound. The alternative methods using backpropagation and relative fre-

[1] http://www.marekrei.com/projects/mltagger

[2] The CoNLL 2010 shared task on uncertainty detection comes with an official scorer which requires additional steps and the detection of both cues and scopes, whereas the binary labels from the zero-shot systems are not directly applicable to this format. Similarly, error detection is commonly evaluated using $F_{0.5}$, which is motivated by end-user experience, but in this case we wish to specifically measure the tagging accuracy. Therefore we use the regular F_1 score as the main evaluation metric for both of these tasks.

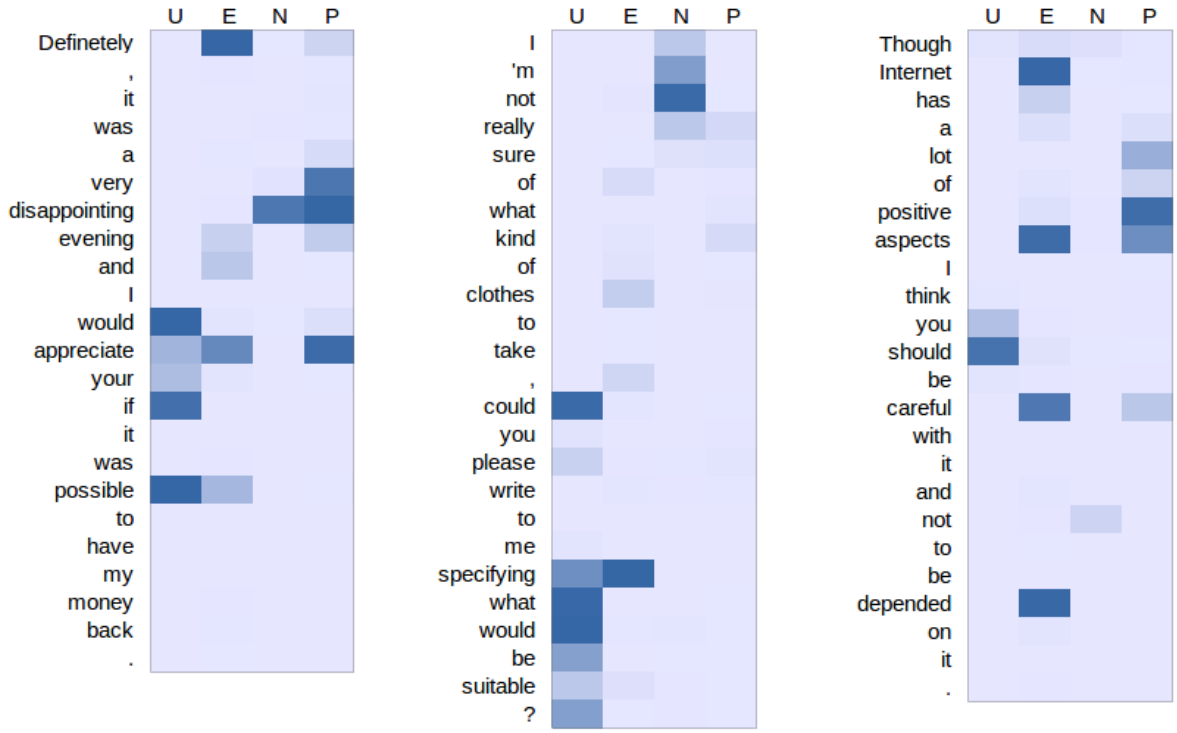


Figure 2: Example output from each of the zero-shot sequence labeling models, trained on 4 different tasks. **U**: uncertainty detection, **E**: error detection, **N**: negative sentiment detection, **P**: positive sentiment detection. Darker blue indicates higher predicted values.

quency achieve high recall values, but comparatively lower precision. On the FCE dataset, the F-score is considerably lower at 28.27% – this is due to the difficulty of the task and the supervised system also achieves only 34.76%. The attention-based system outperforms the alternatives on both of the SemEval evaluations. The task of detecting sentiment on the token level is quite difficult overall as many annotations are context-specific and require prior knowledge. For example, in order to correctly label the phrase *"have Superbowl"* as positive, the system will need to understand that organizing the Superbowl is a positive event for the city.

Performance on the sentence-level classification task is similar for the different architectures on the CoNLL 2010 and FCE datasets, whereas the composition method based on attention obtains an advantage on the SemEval datasets. Since the latter architecture achieves competitive performance and also allows for attention-based token labeling, it appears to be the better choice. Analysis of the token-level MAP scores shows that the attention-based sequence labeling model achieves the best performance even when ignoring classification thresholds and evaluating the task through ranking.

Figure 2 contains example outputs from the attention-based models, trained on each of the four datasets. In the first example, the uncertainty detector correctly picks up *"would appreciate if"* and *"possible"*, and the error detection model focuses most on the misspelling *"Definetely"*. Both the positive and negative sentiment models have assigned a high weight to the word *"disappointing"*, which is something we observed in other examples as well. The system will learn to focus on phrases that help it detect positive sentiment, but the presence of negative sentiment provides implicit evidence that the overall label is likely not positive. This is a by-product of the 3-way classification task and future work could investigate methods for extending zero-shot classification to better match this requirement.

In the second example, the system correctly labels the phrase *"what would be suitable?"* as uncertain, and part of the phrase *"I'm not really sure"* as negative. It also labels *"specifying"* as an error, possibly expecting a comma before it. In the third example, the error detection model labels *"Internet"* for the missing determiner, but also captures a more difficult error in *"depended"*,

which is an incorrect form of the word given the context.

7 Conclusion

We investigated the task of performing sequence labeling without having access to any training data with token-level annotation. The proposed model is optimized as a sentence classifier and an attention mechanism is used for both composing the sentence representations and inferring individual token labels. Several alternative models were compared on three tasks – uncertainty detection, error detection and sentiment detection.

Experiments showed that the zero-shot labeling system based on attention weights achieved the best performance on all tasks. The model is able to automatically focus on the most salient areas of the sentence, and additional objective functions along with the soft attention mechanism encourage it to also perform well as a sequence labeler. The zero-shot labeling task can provide a quantitative evaluation of what the model is learning, along with offering a low-cost method for creating sequence labelers for new tasks, domains and languages.

Acknowledgments

We would like to thank the NVIDIA Corporation for the donation of the Titan GPU that was used for this research. Anders Søgaard was partially funded by the ERC Starting Grant LOWLANDS No. 313695.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *Arxiv preprint arXiv:1603.04467* https://doi.org/10.1038/nn.3331.

Dimitrios Alikaniotis, Helen Yannakoudakis, and Marek Rei. 2016. Automatic Text Scoring Using Neural Networks. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* .

Øistein E. Andersen, Helen Yannakoudakis, Fiona Barker, and Tim Parish. 2013. Developing and testing a self-assessment and tutoring system. *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications* http://www.aclweb.org/anthology/W13-1704.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Learning Representations*. https://doi.org/10.1146/annurev.neuro.26.041002.131047.

Vidas Daudaravicius, Rafael E Banchs, Elena Volodina, and Courtney Napoles. 2016. A Report on the Automatic Evaluation of Scientific Writing Shared Task. *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications* pages 53–62.

Misha Denil, Alban Demiraj, Nal Kalchbrenner, Phil Blunsom, and Nando de Freitas. 2014. Modelling, Visualising and Summarising Documents with a Single Convolutional Neural Network. In *Arxiv preprint arXiv:1406.3830*. http://arxiv.org/abs/1406.3830.

Richárd Farkas, Veronika Vincze, György Móra, János Csirik, and György Szarvas. 2010. The CoNLL-2010 shared task: learning to detect hedges and their scope in natural language text. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, July, pages 1–12. http://dl.acm.org/citation.cfm?id=1870535.1870536.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)* 9:249–256. https://doi.org/10.1.1.207.2059.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching Machines to Read and Comprehend. In *Advances in Neural Information Processing Systems (NIPS 2015)*. pages 1–14. https://doi.org/10.1109/72.410363.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural Computation* 9. https://doi.org/10.1.1.56.7752.

Ozan Irsoy and Claire Cardie. 2014. Opinion Mining with Deep Recurrent Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jianshu Ji, Qinlong Wang, Kristina Toutanova, Yongen Gong, Steven Truong, and Jianfeng Gao. 2017. A Nested Attention Neural Hybrid Model for Grammatical Error Correction. In *ACL 2017*. pages 753–762. https://doi.org/10.18653/v1/P17-1070.

Masahiro Kaneko, Yuya Sakaizawa, and Mamoru Komachi. 2017. Grammatical Error Detection Using Error- and Grammaticality-Specific Word Embeddings. In *Proceedings of the The 8th International Joint Conference on Natural Language Processing*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. In *Proceedings of NAACL-HLT 2016*.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and Understanding Neural Models in NLP. *Proceedings of NAACL-HLT 2016* https://doi.org/10.18653/v1/N16-1082.

Preslav Nakov, Sara Rosenthal, Zornitsa Kozareva, Veselin Stoyanov, Alan Ritter, and Theresa Wilson. 2013. SemEval-2013 task 2: Sentiment analysis in twitter. *Proceedings of the International Workshop on Semantic Evaluation, SemEval* .

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe : Global Vectors for Word Representation. In *Proceedings of the Empiricial Meth- ods in Natural Language Processing (EMNLP 2014)*. https://doi.org/10.3115/v1/D14-1162.

Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *ACL 2017*. https://doi.org/10.18653/v1/P17-1161.

Marek Rei. 2017. Semi-supervised Multitask Learning for Sequence Labeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL-2017)*.

Marek Rei and Helen Yannakoudakis. 2016. Compositional Sequence Labeling Models for Error Detection in Learner Writing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. https://aclweb.org/anthology/P/P16/P16-1112.pdf.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, and Phil Blunsom. 2016. Reasoning about Entailment with Neural Attention. *International Conference on Learning Representations* .

Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif M. Mohammad, Alan Ritter, and Veselin Stoyanov. 2015. SemEval-2015 Task 10: Sentiment Analysis in Twitter. *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)* http://alt.qcri.org/semeval2015/cdrom/pdf/SemEval078.pdf.

Sara Rosenthal, Alan Ritter, Preslav Nakov, and Veselin Stoyanov. 2014. SemEval-2014 Task 9: Sentiment Analysis in Twitter. *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)* http://alt.qcri.org/semeval2014/cdrom/pdf/SemEval009.pdf.

Alexander M Rush and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. In *Proceedings of EMNLP 2015*.

Sheng Syun Shen and Hung Yi Lee. 2016. Neural attention models for sequence classification: Analysis and application to key term extraction and dialogue act detection. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH* 08-12-September-2016:2716–2720. https://doi.org/10.21437/Interspeech.2016-1359.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout : A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research (JMLR)* 15. https://doi.org/10.1214/12-AOS1000.

Veronika Vincze, György Szarvas, Richárd Farkas, György Móra, and János Csirik. 2008. The BioScope corpus: biomedical texts annotated for uncertainty, negation and their scopes. *BMC Bioinformatics* 9(Suppl 11):S9. https://doi.org/10.1186/1471-2105-9-S11-S9.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A New Dataset and Method for Automatically Grading ESOL Texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. http://www.aclweb.org/anthology/P11-1019.

Matthew D. Zeiler. 2012. ADADELTA: An Adaptive Learning Rate Method. *arXiv preprint arXiv:1212.5701* http://arxiv.org/abs/1212.5701.

Matthew D. Zeiler and Rob Fergus. 2014. Visualizing and Understanding Convolutional Networks. *Computer VisionECCV 2014* 8689. https://doi.org/10.1007/978-3-319-10590-1_53.

Variable Typing: Assigning Meaning to Variables in Mathematical Text

Yiannos A. Stathopoulos[♠] **Simon Baker**[♠♣] **Marek Rei**[♠♢] **Simone Teufel**[♠]
[♠]Computer Laboratory, University of Cambridge, United Kingdom
[♣]Language Technology Lab, University of Cambridge, United Kingdom
[♢]The ALTA Institute, University of Cambridge, United Kingdom
{yiannos.stathopoulos,simon.baker,marek.rei,simone.teufel}@cl.cam.ac.uk

Abstract

Information about the meaning of mathematical variables in text is useful in NLP/IR tasks such as symbol disambiguation, topic modeling and mathematical information retrieval (MIR). We introduce *variable typing*, the task of assigning one *mathematical type* (multi-word technical terms referring to mathematical concepts) to each variable in a sentence of mathematical text. As part of this work, we also introduce a new annotated data set composed of 33,524 data points extracted from scientific documents published on arXiv. Our intrinsic evaluation demonstrates that our data set is sufficient to successfully train and evaluate current classifiers from three different model architectures. The best performing model is evaluated on an extrinsic task: MIR, by producing a *typed formula index*. Our results show that the best performing MIR models make use of our typed index, compared to a formula index only containing raw symbols, thereby demonstrating the usefulness of variable typing.

1 Introduction

Scientific documents, such as those from Physics and Computer Science, rely on mathematics to communicate ideas and results. Written mathematics, unlike general text, follows strong domain-specific conventions governing how content is presented. According to Ganesalingam (2008), the sense of mathematical text is conveyed through the interaction of two contexts: the textual context (flowing text) and the mathematical (or symbolic) context (mathematical formulae).

In this work, we introduce a new task that focuses on one particular interaction: the assignment of meaning to variables by surrounding text in the same sentence[1]. For example, in the sentence

> *Let P be a parabolic subgroup of $GL(n)$ with Levi decomposition $P = MN$, where N is the unipotent radical.*

the variables P and N in the symbolic context are assigned the meaning "parabolic subgroup" and "unipotent radical" by the textual context surrounding them respectively.

We will refer to the task of assigning one *mathematical type* to each variable in a sentence as *variable typing*. We use mathematical types (Stathopoulos and Teufel, 2016) as variable denotation labels. Types are multi-word phrases drawn from the technical terminology of the mathematical discourse that label mathematical objects (e.g., "set"), algebraic structures (e.g., "monoid") and instantiable notions (e.g., "cardinality of a set"). In the sentence presented earlier, the phrases "parabolic subgroup", "Levi decomposition" and "unipotent radical" are examples of types.

Typing variables may be beneficial to other natural language processing (NLP) tasks, such as topic modeling, to group documents that assign meaning to variables consistently (e.g., "E" is "energy" consistently in some branches of Physics). In mathematical information retrieval (MIR), for instance, enriching formulae with types may improve precision. For example, the formulae $x + y$ and $a + b$ can be considered α-equivalent matches. However, if a and b are matrices while x and y are vectors, the match is likely to be a false positive. Typing information may be helpful in reducing such instances and improving retrieval precision.

Variable typing differs from similar tasks in three fundamental ways. First, meaning – in the form of mathematical types – is explicitly assigned to variables, rather than arbitrary mathematical expressions. Second, variable typing is carried out at the sentential level, with valid type assignments for variables drawn from the sentences in which

[1]Data for the task is available at https://www.cst.cam.ac.uk/~yas23/

Proceedings of NAACL-HLT 2018, pages 303–312
New Orleans, Louisiana, June 1 - 6, 2018.

they occur, rather than from larger contexts, such as documents. Third, denotations are drawn from a pre-determined list of types, rather than from free-form text in the surrounding context of each variable.

As part of our work, we have constructed a new data set for variable typing that is suitable for machine learning (Section 4) and is distributed under the Open Data Commons license. We propose and evaluate three models for typing variables in mathematical documents based on current machine learning architectures (Section 5). Our intrinsic evaluation (Section 6) suggests that our models significantly outperform the state-of-the-art SVM model by Kristianto et al. (2012, 2014) (originally developed for description extraction) on our data set. More importantly, our intrinsic evaluation demonstrates that our data set is sufficient to successfully train and evaluate classifiers from three different architectures. We also demonstrate that our variable typing task and data are useful in MIR in our extrinsic evaluation (Section 7).

2 Related Work

The task of extracting semantics for variables from the linguistic context was first proposed by Grigore et al. (2009) with the intention of disambiguating symbols in mathematical expressions. Grigore et al. took operators listed in OpenMath content dictionaries (CDs) as concepts and used term clusters to model their semantics. A bag of nouns is extracted from the operator description in the dictionary and enriched manually using terms taken from online lexical resources. The cluster that maximises the similarity (based on Pointwise Mutual Information (PMI) and DICE) between nouns in the cluster and the local context of a target formula is taken to represent its meaning.

Wolska et al. (2011) used the Cambridge dictionary of mathematics and the mathematics subject classification hierarchy to manually construct taxonomies used to assign meaning to *simple expressions*. Simple expressions are defined by the authors to be mathematical formulae taking the form of an identifier, which may have super/subscripted expressions of arbitrary complexity. Lexical features surrounding simple expressions are used to match the context of candidate expressions to suitable taxonomies using a combination of PMI and DICE (Wolska et al., 2011). Wolska et al. report a precision of 66%.

Quoc et al. (2010) used a rule-based approach to extract descriptions for formulae (phrases or sentences) from surrounding context. In a similar approach, Kristianto et al. (2012) applied pattern matching on sentence parse trees and a "nearest noun" approach to extract descriptions. These rule-based methods have been shown to perform well for recall but poorly for precision (Kristianto et al., 2012). However, Kristianto et al. (2012) note that domain-agnostic parsers are confused by mathematical expressions making rule-based methods sensitive to parse tree errors. Both rule-based extraction methods were outperformed by Support Vector Machines (SVMs) (Kristianto et al., 2012, 2014).

Schubotz et al. (2016) use hierarchical named topic clusters, referred to as *namespaces*, to model the semantics of mathematical identifiers. Namespaces are derived from a document collection of 22,515 Wikipedia articles. A vector-space approach is used to cluster documents into namespaces using mini-batch K-means clustering. Clusters beyond a certain purity threshold are selected and converted into namespaces by extracting phrases that assign meaning to identifiers in the selected clusters. Schubotz et al. (2016) take a ranked approach at determining the phrase that best assigns meaning to a particular identifier. The authors report F_1 scores of 23.9% and 56.6% for their definition extraction methods.

In contrast, we assign meaning exclusively to variables, using denotations from a pre-computed dictionary of mathematical types, rather than free-form text. Types as pre-identified, compositionally constructed denotational labels enable efficient determination of relatedness between mathematical concepts. In our extrinsic MIR experiment (Section 7), the mathematical concept that two or more types are derived from is identified by locating their common parent type – the supertype – on a suffix trie. Topically related types that do not share a common supertype can be identified using an automatically constructed type embedding space (Stathopoulos and Teufel (2016), Section 5.1), rather than manually curated namespaces or fuzzy term clusters.

3 The Variable Typing Task

We define the task of variable typing as follows. Given a sentence containing a pre-identified set of

variables V and types T, variable typing is the task of classifying all edges $V \times T$ as either existent (positive) or non-existent (negative).

However, not all elements of $V \times T$ are valid edges. Invalid edges are usually instances of *type parameterisation*, where some type is parameterised by what appears to be a variable. For example, the set of candidate edges for the sentence

> *We now consider the* $\boldsymbol{q}$*-exterior algebras of* $\boldsymbol{V}$ *and* $\boldsymbol{V^*}$*, cf. [21].*

would include $(\boldsymbol{V}, \text{exterior algebra})$ and $(\boldsymbol{V^*}, \text{exterior algebra})$ but *not* $(\boldsymbol{q}, \text{exterior algebra})$. Such edges are identified using pattern matching (Java regular expressions) and are not presented to annotators or recorded in the data set.

Our definition of "variable" mirrors that of "simple expression" proposed by Grigore et al. (2009): instances of formulae in the discourse are considered to be "typeable variables" if they are only composed of a single, potentially scripted base identifier.

Variable typing, as defined in this work, is based on four assumptions: (1) typings occur at the sentential level and variables in a sentence can only be assigned a type phrase occurring in that sentence, (2) variables and types in the sentence are known a priori, (3) edges in each sentence are independent of one another, and (4) edges in one sentence are independent of those in other sentences – given a variable v in sentence s, type assignment for v is agnostic of other typings involving v from other sentences.

The decision to constrain variable typing at the sentential level is motivated by empirical studies (Grigore et al., 2009; Gödert, 2012). Grigore et al. (2009) have shown that the majority of variables are introduced and declared in the same sentence. In addition, mathematical text tends to be composed of local contexts, such as theorems, lemmas and proofs (Ganesalingam, 2008).

The assumptions introduced above simplify the task of variable typing without sacrificing the generalisability of the task. For example, cases where the same variable is assigned multiple conflicting types from different sentences within a document can be collected and resolved using a *type disambiguation* algorithm.

4 Variable Typing Data Set

We have constructed an annotated data set of sentences for building variable typing classifiers. The sentences in our corpus are sourced from the Mathematical REtrieval Corpus (MREC) (Líška et al., 2011), a subset of arXiv (over 439,000 papers) with all LaTeX formulae converted to MathML.

	Train	Dev	Test	Total
Sentences	5,273	841	1,689	**7,803**
Positive edges	1,995	457	1,049	**3,501**
Negative edges	15,164	4,386	10,473	**30,023**
Total edges	17,159	4,843	11,522	**33,524**

Table 1: Data set statistics.

The data set is split into a standard training/development/test machine learning partitioning scheme as outlined in Table 1. The idea behind this scheme is to train and evaluate new models on standardised data partitions so that results can be directly comparable.

4.1 Sentence Sampling

The structure and role of sentences in mathematical papers may vary according to their location in the discourse. For example, sentences in the "Introduction" – intended to introduce the subject matter – can be expected to differ in structure from those in a proof, which tend to be short, formal statements. Our sampling strategy is designed to control for this diversity in sentence structure. First, we sentence-tokenised and transformed each document in the MREC into a graph that encodes its section structure. Document graphs also take into account blocks of text unique to the mathematical discourse such as theorems, proofs and definitions. Then, we sampled sentences for our data set by distribution according to their location in the source arXiv document.

Variables in each MREC document are identified via a parser that recognises the variable description given in Section 3. Our variable parser is designed to operate on *symbol layout trees* (SLTs) (Schellenberg et al., 2012) – trees representing the 2-dimensional presentation layout of mathematical formulae. We identified 28.6 million sentences that contain variables.

The distribution of sentences according to (a) the type of discourse/math block of origin and (b) the number of unique types in the sentence is reconstructed by putting sentences into bins based

on the value of these features. Sentences are selected from the bins at random in proportion to their size. The training, development and test samples have been produced via repeated application of this sample-by-distribution strategy over the set of all sentences that contain variables.

4.2 Extended Type Dictionary

The type dictionary distributed by Stathopoulos and Teufel (2016) contains 10,601 automatically detected types from the MREC. However, the MREC contains 2.9 million distinct technical terms, many of which might also be types. Therefore, the seed dictionary is too small to be used with variable typing at scale since types from the seed dictionary will be sparsely present in sampled sentences. To overcome this problem, we used the *double suffix trie algorithm* (DSTA) to automatically expand the type dictionary. The algorithm makes use of the fact that most types are compositional (Stathopoulos and Teufel, 2016): longer subtypes can be constructed out of shorter supertypes by attaching pre-modifiers (e.g., a "Riemannian manifold" can be considered a subtype of "manifold").

The DSTA takes two lists of technical terms as input – the seed dictionary of types and the MREC master list (2.9 million technical terms). First, technical terms on both lists are word-tokenised. Then, all technical terms in the seed dictionary (the known types) are placed onto the *known types suffix trie* (KTST). Additional types are generated from single word types on the KTST by expanding them with one of 40 prefixes observed in the corpus. For example, the type "algebra" might generate the supertype "coalgebra". These are also added on the KTST as known types.

Technical terms in the KTST are copied onto the *candidate type suffix trie* (CTST) and are labeled as types. Next, the technical terms on the master list are inserted into the CTST. Technical terms in the master list that have known types from the seed dictionary as their suffix on the CTST are also marked as types. A new dictionary of types (in the form of a list of technical terms) is produced by traversing the CTST and recording all phrases that have a known type as their suffix. This way, we have expanded the type dictionary from 10,601 types to approximately 1.23 million technical terms, from which an updated KTST can be produced.

4.3 Human Annotation and Agreement

Two of the authors jointly developed the annotation scheme and guidelines using sentences sampled by distribution as discussed in Section 4.1. Sentences sampled for this purpose are excluded from subsequent sampling. The labeling scheme, presented in Table 2, implements the assumptions of the variable typing task – each variable in a sentence is assigned exactly one label: either one type from the sentence or one of six fixed labels for special situations.

An annotation experiment was carried out using two authors as annotators to investigate (a) how intuitive the task of typing is to humans and (b) the reliability of the annotation scheme. For this purpose, a further 1,000 sentences were sampled (and removed) from the pool and organised into two subsamples each with 554 sentences. The subsamples have an overlap of 108 sentences with a total of 182 edges, which are used to measure inter-annotator agreement.

We report annotator agreement for three separate cases. The first case reflects whether annotators agree that a variable can be typed or not by its context. A variable falls into the first category if it is assigned a type from the sentential context and in the latter category if it is assigned one of the six fixed labels from Table 2. In this case, agreement is substantial (Cohen's $K = 0.80$, $N = 182$, $k = 2$, $n = 2$). The second case is for instances where both annotators believe a variable can be typed by its sentential context – the variable is assigned a type by both annotators. In this case, Cohen's Kappa is not applicable because the number of labels varies: there are as many labels as there are types in the sentence. Instead, we report accuracy as the proportion of decisions where annotators agree over all decisions: 90.9%. In the last case where both annotators agree that a variable is not a type (i.e., is assigned one of the six fixed labels), agreement has been found to be moderate (Fleiss' $K = 0.61$, $N = 123$, $k = 2$, $n = 6$).

The bulk of the annotation was carried out by one of the author-annotators and was produced by repeated sampling by distribution (as described in Section 4.1). Sentences in the bulk sample are combined with the 554 sentences annotated by the author during the annotation experiment to produce a final data set composed of 7,803 sentences. The training, test and development sets have been produced using the established 70% for training,

Label	Description
One label per type instance	One label per instance of any type in the sentence.
Type Unknown	The type of the variable is not in the scope of the sentence.
Type Present but Undetected	The type of the variable is in the scope of the sentence but is not in the dictionary.
Parameterisation	Variable is part of an instance of parameterisation.
Index	Variable is an instance of indexing (numeric or non-numeric).
Number	Variable is implied to be a number by the textual context (e.g., "the n-th element...").
Formula is not a variable	Label used to mark data errors. For example, in some instances end-of-proof symbols are encoded as identifiers in the corpus and are mistaken for variables.

Table 2: Labels for special typing situations.

20% for test and 10% for development data set partitioning strategy. Each partition is sampled by distribution in order to model training and predicting typings over complete discourse units, such as documents.

5 Experiments

We compare three models for variable typing to two baselines: the "nearest type" baseline and the SVM proposed by Kristianto et al. (2014). One of our models is an extension of the latter baseline with both type and variable-centric features. The other two models are based on deep neural networks: a convolutional neural network and a bidirectional LSTM.

We treat the task of typing as binary classification: every possible typing in a sentence is presented to a classifier which, in turn, is expected to make a "type" or "not-type" decision. We say that an edge is *positive* if it connects a variable to a type in the sentence and *negative* otherwise.

5.1 Computing a Type Embedding Space

We use the extended dictionary of types (Section 4.2) to pre-train a *type embedding space*. Computed over the MREC, a type embedding space includes embeddings for both words and types (as atomic lexical tokens). These vectors are used by our deep neural networks to model the distributional meaning of words and types. The type embedding space is constructed using the process described by Stathopoulos and Teufel (2016): occurrences of extended dictionary type phases in the MREC are substituted with unique atomic lexical units before the text is passed on to `word2vec`.

5.2 Models for Variable Typing

Nearest Type baseline (NT) Given a variable v, the nearest type baseline takes the edge that minimises the word distance between v and some type in the sentence to be the positive edge. This baseline is intended to approximate the "nearest noun" baseline (Kristianto et al., 2012, 2014) which we cannot directly compute due to the fact that noun phrases in the text become parts of types.

Support Vector Machine (Kristianto et al.) (SVM) This is an implementation of the features and linear SVM described by Kristianto et al. (2012). Furthermore, we use the same value for hyperparameter C (the soft margin cost parameter) used by Kristianto et al. (2012). Due to the class imbalance in our data set we have used inversely proportional class weighting (as implemented in scikit-learn). L2-normalisation is also applied.

Extended Support Vector Machine (SVM+) We have extended the SVM proposed by Kristianto et al. (2012) with the features that are type and variable-centric, such as the 'base symbol of a candidate variable' and 'first letter in the candidate type'. A description of these extended features are listed in Table 4. We applied automatic class weighting and L2-normalisation. We have found that $C = 2$ is optimal for this model by fine-tuning over the development set.

Convolutional Neural Network (Convnet) We use a Convnet to classify each of the $V \times T$ assignment edges as either positive or negative, where V are the variables in the input text and T are the types. Unlike the SVM models, we do not use any hand-crafted features, but only the inputs (Table 3), and the pre-trained embeddings (Section 5.1).

The input is a tensor that encodes the input described in Table 3. We use the embeddings to represent the input tokens. In addition, we concatenate two dimensions to the input for each token : one dimension to denote (using 1 or 0) whether a given token is a type and another dimension to denote if a token is a variable.

The model has a set of different sized filters, and each filter size has an associated number of filters to be applied (all are hyperparameters to

Name	Description
Token	A word in the sentence. If the token is a formula (including a variable), the token is '@@@'. Types are represented by the key of their embedding vector.
Token class	An integer – 0 for normal word, 1 for type, 2 for variable and 3 to indicate that a variable token is part of the edge being considered.
Type of Interest	If the token is a type and it is part of the edge being considered, this field takes the value 'TYPE' or '-' otherwise.

Table 3: Input and features to neural network typing models.

Orientation	Description
Type	Number of words in the candidate type.
Type	The base type of each candidate type.
Type and Variable	The first letter in the type and base symbol of the candidate variable.
Type	The grammatical number of the type as it appears in the sentence.
Variable	The variables and symbols in the candidate variable layout graph (one string per symbol).
Variable	The number of distinct symbols in the candidate variable layout graph.
Variable	The base symbol of the candidate variable layout graph.
Variable	The directions (Above, Below,Up-left, Up-right, Down-left, Down-right, Next) in which a candidate symbol has neigbouring symbols.
Variable	Operators in the mathematical context of the candidate variable layout graph.
Sentence	Prefix sequence: tokens from start of sentence to a (exclusive)
Sentence	Middle sequence: tokens between a and b (exclusive)
Sentence	Suffix sequence: tokens between b (exclusive) and end of sentence.

Table 4: SVM+ features. For each edge e, let a be the position of its left-most component (variable or type) and b the position of its rightmost component (variable or symbol).

the model). The filters are applied to the input text (i.e. convolutions), and then max-pooled, flattened, concatenated, and a dropout layer ($p = 0.5$) is then applied before being fed into a multilayer perceptron (MLP), with the number of hidden layers and their hidden units as hyperparameters. Finally, a softmax layer is used to output a binary decision.

The model is implemented using the Keras library using binary cross-entropy as loss function, and the ADAM optimizer (Kingma and Ba, 2014). We tune the aforementioned hyperparameters on the development data and we use balanced oversampling with replacement in order to adjust for the class imbalance in the data.

Our tuned hyperparameters are as follows: filter window sizes (2 to 12, then 14,16,18,20) with an associated number of filters (300 for the first five, 200 for the next four, 100 for the next three, then 75,70,50). One hidden layer of the MLP with 512 units is used with batch size 50.

Bidirectional LSTM (BiLSTM) The architecture takes as input a sequence of words, which are then mapped to word embeddings. For each token in the input sentence, we also include the inputs described in Table 3. In addition, the model uses one string feature we refer to as "supertype". If the token is a type, then this feature is the string key of the embedding vector of its supertype or "NONE" otherwise.

These features are mapped to a separate embedding space and then concatenated with the word embedding to form a single task-specific word representation. This allows us to capture useful information about each word, and also designate which words to focus on when processing the sentence.

We use a neural sequence labeling architecture, based on the work of Lample et al. (2016) and Rei and Yannakoudakis (2016). The constructed word representations are given as input to a bidirectional LSTM (Hochreiter and Schmidhuber, 1997), and a context-specific representation of each word is created by concatenating the hidden representations from both directions.

A hidden layer is added on top to combine the features from both directions. Finally, we use a softmax output layer that predicts a probability distribution over positive or negative assignment for a given edge.

We also make use of an extension of neural sequence labeling that combines character-based word representations with word embeddings using a predictive gating operation (Rei et al., 2016). This allows our model to capture character-level patterns and estimate representations for previously unseen words.

In this framework, an alternative word repre-

sentation is constructed from individual characters, by mapping characters to an embedding space and processing them with a bidirectional LSTM. This representation is then combined with a regular word embedding by dynamically predicting element-wise weights for a weighted sum, allowing the model to choose for each feature whether to take the value from the word-level or character-level representation.

The LSTM layer size was set to 200 in each direction for both word- and character-level components; the hidden layer d was set to size 50. During training, sentences were grouped into batches of size 64. Performance on the development set was measured at every epoch and training was stopped when performance had not improved for 10 epochs; the best-performing model on the development set was then used for evaluation on the test set.

6 Intrinsic Evaluation

Evaluation is performed over edges, rather than sentences, in the test set. We measure performance using precision, recall and F_1-score. We use the non-parametric *paired randomisation test* to detect significant differences in performance across classifiers.

The convnet and BiLSTM models are trained and evaluated with as many sentences as there are edges: the source sentence is copied for each input edge, with inputs modified to reflect the relation of interest. We employed early stopping and dropout to avoid overfitting with these models.

Table 5 shows the performance results of all classifiers considered. All three proposed models have significantly outperformed the NT baseline and Kristianto et al.'s (Kristianto et al., 2014) state-of-the-art SVM. The best performing model is the bidirectional LSTM ($F_1 = 78.98\%$) which has significantly outperformed all other models ($\alpha = 0.01$).

According to the results in Table 5, both deep neural network models have significantly outperformed classifiers based on other paradigms. This is consistent with the intuition that the language of mathematics is formulaic: we expect deep neural networks to effectively recognise patterns and identify correlations between tokens.

The neural models outperform SVM+ despite the fact that the latter is a product of laborious manual feature engineering. In contrast, no manual feature engineering has been performed on the Convnet model (or indeed on any of the deep neural network models).

	Precision (%)	Recall (%)	F_1-score (%)
NT	30.30	82.94	44.39
SVM	55.39	76.36	64.21
SVM+	71.11	72.74	71.91
Convnet	80.11	70.26	74.86
BiLSTM	83.11	74.77	78.98

Table 5: Model performance summary. All figures are statistically significant ($p < 0.01$) according to the randomisation test.

The nearest type (NT) baseline demonstrates high recall but low precision. This is not surprising since the NT baseline is not capable of making a negative decision: it always assigns some type to all variables in a given sentence.

7 Extrinsic Evaluation

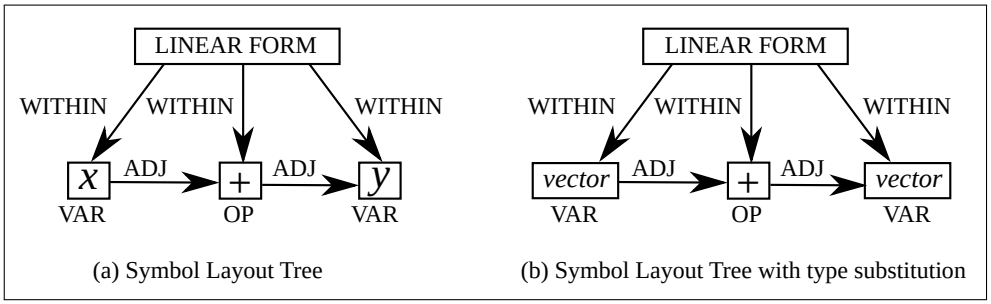
We demonstrate that our data set and variable typing task are useful using a mathematical information retrieval (MIR) experiment. The hypothesis for our MIR experiment is two-fold: (a) types identified in the textual context for the variable typing task are also useful for text-based mathematical retrieval and (b) substituting raw symbols with types in mathematical expressions will have an observable effect to MIR.

In order to motivate the second hypothesis, consider the following natural language query:

> *Let x be a vector. Is there another vector y such that $x + y$ will produce the zero element?*

In the context of MIR, mathematical expressions are represented using SLTs (Pattaniyil and Zanibbi, 2014) that are constructed by parsing presentation MathML. The expression "$x + y$" is represented by the SLT in figure 1(a). The variable typing classifier and the type disambiguation algorithm determine the types of the variables x and y as "vector". Thus, the variable nodes in figure 1(a) will be substituted with their type, producing the SLT in figure 1(b).

The example query can be satisfied by identifying a vector y such that when added to x will produce the zero vector. This operation is abstract in mathematics and extends to objects beyond vectors, including integers. In an untyped formula index, there is no distinction between instances of $x + y$ where the variables are integers or vectors. As a result, documents where both variables are integers might also be returned. In contrast,



(a) Symbol Layout Tree (b) Symbol Layout Tree with type substitution

Figure 1: (a) SLT representation of the expression $x + y$, (b) typed SLT for the expression $x + y$.

a typed formula index will return instances of the typed SLT in figure 1(b) where the variables are vectors, as opposed to integers. Therefore, a typed index can reduce the number of false positives and increase precision.

Four MIR retrieval models are introduced in Section 7.3 designed to control for text indexing/retrieval so that the effects of type-aware vs type-agnostic formula indexing and scoring can be isolated. These models make use of the Tangent formula indexing and scoring functions (Pattaniyil and Zanibbi, 2014), which we have implemented.

We use the Cambridge University Math IR Test Collection (CUMTC) (Stathopoulos and Teufel, 2015) which is composed of 120 research-level mathematical information needs and 160 queries. The CUMTC is ideal for our evaluation for two reasons. First, topics in the CUMTC are expressed in natural language and are rich in mathematical types. This allows us to directly apply our best performing variable typing model (BiLSTM) in our retrieval experiment in order to extract variable typings for documents and queries. Second, the CUMTC uses the MREC as its underlying document collection, which enables downstream evaluation in an optimal setting for variable typing.

7.1 Tangent Formula Indexing and Scoring

Given a mathematical formula, the Tangent indexing algorithm starts from the root node of an SLT and generates symbol pair tuples in a depth-first manner. Symbol pair tuples record parent/child relationships between SLT nodes, the distance (number of edges) and vertical offset between them. At each step in the traversal, the index is updated to record one tuple representing the relationship between the current node and every node in the path to the SLT root. We have also implemented Tangent's method of indexing matrices, but we refer the reader to Pattaniyil and Zanibbi (2014) for further details.

Tangent scoring proceeds as follows. For each query formula, the symbol pair tuples are generated and matched exactly to those in the document index. Let C denote the set of matched index formulae and $|s|$ the number of symbol pairs in any given expression s in C. For each s in C, recall (R) is said to be $\frac{|C|}{|Q|}$, where $|C|$ and $|Q|$ are the numbers of tuples in C and the query formula Q respectively, and precision (P) is $\frac{|C|}{|s|}$. Candidate s is assigned the F score of these precision and recall values. The mathematical context score for a given document d and query with formulae $e_1, \ldots, e_n$ is

$$m(d, e_1, \ldots, e_n) = \sum_{j=1}^{n} \frac{|e_j| \cdot t1(d, e_j)}{\sum_{i=1}^{n} |e_i|}$$

where $|e_j|$ represents the number of tuples in expression e_j and $t1(d, e_j)$ represents the top F-score for expression e_i in document d. The final score for document d is a linear combination of the math context score above and its Lucene text score (L(d)):

$$\lambda \times L(d) + (1 - \lambda) \times m(d, e_1, \ldots, e_n)$$

7.2 Typed Tangent Indexing and Scoring

We have applied the BiLSTM variable typing model to obtain variable typings for all symbols in the documents in the MREC. For each document in the collection our adapted Tangent formula indexer first groups the variable typing edges for that document according to the variable identifier involved. Subsequently, our typed indexing process applies a type disambiguation algorithm to determine which of the candidate types associated with the variable will be designated as its type.

For a variable v in document d, our type disambiguation algorithm first looks at the known types suffix trie (KTST) containing all 1.23 million types in order to find a common parent be-

tween the candidate types. If a common supertype T is discovered, then v is said to be of type T. Otherwise, the type disambiguation algorithm uses simple majority vote amongst the candidates to determine the final type for variable v.

The type disambiguation algorithm is applied to every typing group until all variable typings have been processed. Variable groups with no type candidates (e.g., no variable typings have been extracted for a variable) are assigned a missing type symbol ("*"). Subsequently, variables in the SLT of each formula in d are replaced with their type or the missing type symbol. An index, referred to as the typed index, is generated by applying the tangent indexing process on the modified SLTs.

The same process is applied to query formulae during query time in order to facilitate typed matching and scoring.

7.3 Results

We have replicated runs of the Lucene vector-space model (VSM) and BM25 models presented by Stathopoulos and Teufel (2016) on the CUMTC. Furthermore, we introduce four models based on Tangent indexing and scoring that represent different strategies in handling types in text and formulae. We refer to a model as *typed* if it uses the type-substituted version of the Tangent index and *untyped* otherwise.

Text with types removed (RT): The Lucene score $L(d)$ is computed over a text index with type phrases completely removed. This model is intended to isolate the performance of retrieval on the formula index alone. We consider both typed and untyped instances of this model.

Text with types(TY): The Lucene score is computed over a text index that treats type phrases as atomic lexical tokens. This model is intended to simulate type-aware text that enables the application of variable typing. Both typed and untyped instances of this model are considered.

Optimal values for the linear combination parameter λ are obtained using 13 queries in the "development set" of the CUMTC. We report mean average precision (MAP) for our models computed over all 160 queries in the main CUMTC. MAPs obtained over the CUMTC are low due to the difficulty of the queries rather than an unstable evaluation (Stathopoulos and Teufel, 2016). The paired randomisation test is used to test for significance in retrieval performance gains between the models.

	VSM	**BM25**		
MAP	.076	.079		
	RT typed	**RT untyped**	**TY typed**	**TY untyped**
MAP	.046	.052	.139	.083
λ_{opt}	.9	.9	.9	.4

Table 6: MIR model performance summary.

The results of our MIR experiments are presented in Table 6. The best performing model is TY/typed which significantly outperforms all other baselines ($p - value < 0.05$ for comparison with BM25 and $p - value < 0.01$ with all other models). The TY/typed model yields almost double the MAP performance of its untyped counterpart (TY/untyped, .083 MAP). In contrast, the RT/typed and RT/untyped models perform comparably (no significant difference) but poorly. This drop in MAP performance suggests that type phrases are beneficial for text-based retrieval of mathematics. Retrieval models employing formula indexing seem to be affected by both the presence of types in the text as well as in the formula index. The TY/typed model outperforms the TY/untyped model, which in turn outperforms RT/untyped. This suggests that gains in retrieval performance are strongest when types are used in both text and formula retrieval – models using either approach alone do not perform as well. These results demonstrate that variable typing is a valuable task in MIR.

8 Conclusions

This work introduces the new task of variable typing and an associated data set containing 33,524 labeled edges in 7,803 sentences. We have constructed three variable typing models and have shown that they outperform the current state-of-the-art methods developed for similar tasks. The BiLSTM model is the top performing model achieving 79% F_1-score. This model is then evaluated in an extrinsic downstream task–MIR, where we augmented Tangent formula indexing with variable typing. A retrieval model employing the typed Tangent index outperforms all considered retrieval models demonstrating that our variable typing task, data and trained model are useful in downstream applications. We make our variable typing data set available through the Open Data Commons license.

References

Mohan Ganesalingam. 2008. *The Language of Mathematics*. Ph.D. thesis, Cambridge University Computer Laboratory.

Winfried Gödert. 2012. Detecting multiword phrases in mathematical text corpora. *CoRR*, abs/1210.0852.

Mihai Grigore, Magdalena Wolska, and Michael Kohlhase. 2009. Towards context-based disambiguation of mathematical expressions. In *The joint conference of ASCM 2009 and MACIS 2009. 9th international conference on Asian symposium on computer mathematics and 3rd international conference on mathematical aspects of computer and information sciences, Fukuoka, Japan, December 14–17, 2009. Selected papers.*, pages 262–271. Fukuoka: Kyushu University, Faculty of Mathematics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural Computation*, 9.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Giovanni Yoko Kristianto, Minh quoc Nghiem, Yuichiroh Matsubayashi, and Akiko Aizawa. 2012. Extracting definitions of mathematical expressions in scientific papers. In *In JSAI*.

Giovanni Yoko Kristianto, Goran Topic, and Akiko Aizawa. 2014. Exploiting textual descriptions and dependency graph for searching mathematical expressions in scientific papers.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. In *Proceedings of NAACL-HLT 2016*.

Martin Líška, Petr Sojka, Michal Růžička, and Petr Mravec. 2011. Web interface and collection for mathematical retrieval: Webmias and mrec. In *Towards a Digital Mathematics Library.*, pages 77–84, Bertinoro, Italy. Masaryk University.

Nidhin Pattaniyil and Richard Zanibbi. 2014. Combining TF-IDF text retrieval with an inverted index over symbol pairs in math expressions: The tangent math search engine at NTCIR 2014. In *Proceedings of the 11th NTCIR Conference on Evaluation of Information Access Technologies, NTCIR-11, National Center of Sciences, Tokyo, Japan, December 9-12, 2014*.

Minh Nghiem Quoc, Keisuke Yokoi, Yuichiroh Matsubayashi, and Akiko Aizawa. 2010. Mining coreference relations between formulas and text using wikipedia.

Marek Rei, Gamal K. O. Crichton, and Sampo Pyysalo. 2016. Attending to Characters in Neural Sequence Labeling Models. In *Coling 2016*.

Marek Rei and Helen Yannakoudakis. 2016. Compositional Sequence Labeling Models for Error Detection in Learner Writing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Thomas Schellenberg, Bo Yuan, and Richard Zanibbi. 2012. Layout-based substitution tree indexing and retrieval for mathematical expressions. pages 82970I–82970I–8.

Moritz Schubotz, Alexey Grigorev, Marcus Leich, Howard S. Cohl, Norman Meuschke, Bela Gipp, Abdou S. Youssef, and Volker Markl. 2016. Semantification of identifiers in mathematics for better math information retrieval. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 135–144, New York, NY, USA. ACM.

Yiannos Stathopoulos and Simone Teufel. 2015. Retrieval of research-level mathematical information needs: A test collection and technical terminology experiment. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 334–340.

Yiannos Stathopoulos and Simone Teufel. 2016. Mathematical information retrieval based on type embeddings and query expansion. In *Proceedings of the 26th International Conference on Computational Linguistics, Coling 2016, December 11-16, 2016, Osaka, Japan*, pages 334–340.

Magdalena Wolska, Mihai Grigore, and Michael Kohlhase. 2011. Using discourse context to interpret object-denoting mathematical expressions. In *Towards a Digital Mathematics Library. Bertinoro, Italy, July 20-21st, 2011*, pages 85–101. Masaryk University Press.

Learning beyond datasets: Knowledge Graph Augmented Neural Networks for Natural language Processing

Annervaz K M[*] **Somnath Basu Roy Chowdhury**[*†] **Ambedkar Dukkipati**
Indian Institute of Science, Accenture Technology Labs annervaz@iisc.ac.in
IIT Kharagpur brcsomnath@ee.iitkgp.ernet.in
Indian Institute of Science ambedkar@iisc.ac.in

Abstract

Machine Learning has been the quintessential solution for many AI problems, but learning models are heavily dependent on specific training data. Some learning models can be incorporated with prior knowledge using a Bayesian setup, but these learning models do not have the ability to access any organized world knowledge on demand. In this work, we propose to enhance learning models with *world knowledge* in the form of Knowledge Graph (KG) fact triples for Natural Language Processing (NLP) tasks. Our aim is to develop a deep learning model that can extract relevant prior support facts from *knowledge graphs* depending on the task using attention mechanism. We introduce a *convolution-based model* for learning representations of knowledge graph entity and relation clusters in order to reduce the attention space. We show that the proposed method is highly scalable to the amount of prior information that has to be processed and can be applied to any generic NLP task. Using this method we show significant improvement in performance for text classification with 20Newsgroups (News20) & DBPedia datasets, and natural language inference with Stanford Natural Language Inference (SNLI) dataset. We also demonstrate that a deep learning model can be trained with substantially less amount of labeled training data, when it has access to organized world knowledge in the form of a knowledge base.

1 Introduction

Today, machine learning is centered around algorithms that can be trained on available task-specific labeled and unlabeled training samples. Although learning paradigms like Transfer Learning (Pan and Yang, 2010) attempt to incorporate knowledge from one task into another, these techniques are limited in scalability and are specific to the task at hand. On the other hand, humans have the intrinsic ability to elicit required past knowledge from the world on demand and infuse it with newly learned concepts to solve problems.

The question that we address in this paper is the following: Is it possible to develop learning models that can be trained in a way that it is able to infuse a general body of world knowledge for prediction apart from learning based on training data?

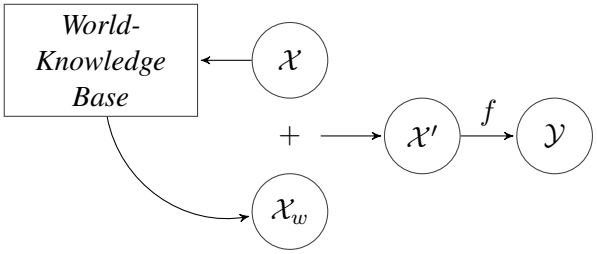


Figure 1: The Basic Idea: $\mathcal{X}$ is the feature input and $\mathcal{Y}$ is the prediction. The relevant world knowledge for the task $\mathcal{X}_w$, is retrieved and augmented with the feature input before making the final prediction

By world knowledge, we mean structured general purpose knowledge that need not be domain specific. Knowledge Graphs (Nickel et al., 2016a) are a popular source of such structured world knowledge. Knowledge Graphs represent information in the form of fact triplets, consisting of a subject entity, relation and object entity (example: *<Italy, capital, Rome>*). The entities represent the nodes of the graph and their relations act as edges. A fact triple (subject entity, relation, object relation) is represented as (h, r, t). Practical knowledge bases congregate information from secondary databases or extract facts from unstructured text using various statistical learning mechanisms, examples of such systems are NELL (Mitchell et al., 2015) and DeepDive (Niu

[*] equal contribution
[†] Main work done during internship at Accenture Technology Labs

Proceedings of NAACL-HLT 2018, pages 313–322
New Orleans, Louisiana, June 1 - 6, 2018.

et al., 2012). There are human created knowledge bases as well, like Freebase (FB15k) (Bollacker et al., 2008) and WordNet (Miller et al., 1990). The knowledge present in these knowledge bases includes common knowledge and partially covers common-sense knowledge and domain knowledge (Song and Roth, 2017). Knowledge Graphs and Knowledge Bases are conceptually equivalent for our purpose and we will use the name interchangeably in this paper.

We illustrate the significance of world knowledge using a few examples. For the example of a Natural Language Inference (NLI) problem (MacCartney, 2009), consider the two following statements, A: `The couple is walking on the sea shore` and B: `The man and woman are wide awake`. Here, for a learning model to infer B from A, it should have access to the common knowledge that "*The man and woman* and *The couple* means the same" since this information may not be specific for a particular inference. Further, it is not possible for a model to learn all such correlations from just the labeled training data available for the task.

Consider another example of classifying the news snippet, `Donald Trump offered his condolences towards the hurricane victims and their families in Texas`. We cannot classify it as a political news unless we know the facts <*Donald Trump, president, United States*> and <*Texas, state, United States*>. We posit that machine learning models, apart from training them on data with the ground-truth can also be trained to fetch relevant information from structured knowledge bases in order to enhance their performance.

In this work, we propose a deep learning model that can extract relevant support facts on demand from a knowledge base (Mitchell et al., 2015) and incorporate it in the feature space along with the features learned from the training data (shown in Figure 1). This is a challenging task, as knowledge bases typically have millions of fact triples. Our proposed model involves a deep learning mechanism to jointly model this look-up scheme along with the task specific training of the model. The look-up mechanism and model is generic enough so that it can be augmented to any task specific learning model to boost the learning performance. In this paper, we have established superior performance of the proposed KG-augmented models over vanilla model on text classification and natural language inference.

Although there is a plethora of work on knowledge graph representation (Nickel et al., 2016a) (Mitchell et al., 2015) (Niu et al., 2012) from natural language text, no attempt to augment learning models with knowledge graph information have been done. To the best of our knowledge this is the first attempt to incorporate world knowledge from a knowledge base for learning models.

2 Knowledge Graph Representations

Knowledge Graph entities/relations need to be encoded into a numerical representation for processing. Before describing the model, we provide a brief overview of graph encoding techniques. Various KG embedding techniques can be classified at a high level into: *Structure-based embeddings* and *Semantically-enriched embeddings*.

Structure-based embeddings: TransE (Bordes et al., 2013) is the introductory work on knowledge graph representation, which translated subject entity to object entity using one-dimensional relation vector $(h + r = t)$. Variants of the TransE (Bordes et al., 2013) model uses translation of the entity vectors over relation specific subspaces. TransH (Wang et al., 2014b) introduced the relation-specific hyperplane to translate the entities. Similar work utilizing only the structure of the graph include ManifoldE (Xiao et al., 2015b), TransG (Xiao et al., 2015a), TransD (Ji et al., 2015), TransM (Fan et al., 2014), HolE (Nickel et al., 2016b) and ProjE (Shi and Weninger, 2017).

Semantically-enriched embeddings: These embedding techniques learn to represent entities/relations of the KG along with its semantic information. Neural Tensor Network(NTN) (Socher et al., 2013) was the pioneering work in this field which initialized entity vectors with the average word embeddings followed by tensor-based operations. Recent works involving this idea are "Joint Alignment" (Zhong et al., 2015) and SSP (Xiao et al., 2017). DKRL (Xie et al., 2016) is a KG representation technique which also takes into account the descriptive nature of text keeping the simple structure of TransE model. Pretrained word2vec (Mikolov et al., 2013) is used to form the entity representation by passing through a Convolutional Neural Network (CNN) (Kim, 2014) architecture constraining the relationships to hold.

In our experiments we have used the DKRL (Xie et al., 2016) encoding scheme as it emphasizes on the semantic description of the text. Moreover, DKRL fundamentally uses TransE (Bordes et al., 2013) method for encoding structural information. Therefore, we can retrieve relevant entities & relation and obtain the complete the fact using $t = h + r$. This reduces the complexity of fact retrieval as the number of entities/relations is much less compared to the number of facts, thus making the retrieval process faster.

3 The Proposed Model

Conventional supervised learning models with parameters Θ, given training data x and label y, tries to maximize the following function

$$\max_{\Theta} P(y|x, \Theta)$$

The optimized parameters Θ are given as,

$$\Theta = \underset{\Theta}{\operatorname{argmax}} \log P(y|x, \Theta)$$

In this work, we propose to augment the supervised learning process by incorporation of world knowledge features x_w. The world knowledge features are retrieved using the data x, using a separate model where, $x_w = F(x, \Theta^{(2)})$. Thus, our modified objective function can be expressed as

$$\max_{\Theta} P(y|x, x_w, \Theta^{(1)})$$

where, $\Theta = \{\Theta^{(1)}, \Theta^{(2)}\}$. The optimized parameters can be obtained using the equation

$$\Theta = \underset{\Theta}{\operatorname{argmax}} \log P(y|x, F(x, \Theta^{(2)}), \Theta^{(1)})$$

The subsequent sections focus on the formulation of the function F which is responsible for fact triple retrieval using the data sample x. Here it is important to note that, we are not assuming any structural form for P based on F. So the method is generic and applicable to augment any supervised learning setting with any form for P, only constraint being P should be such that the error gradient can be computed with respect to F. In the experiments we have used softmax using the LSTM (Greff et al., 2015) encodings of the input as the form for P. As for F, we use soft attention (Luong et al., 2015; Bahdanau et al., 2014) using the LSTM encodings of the input and appropriate representations of the fact(s). Based on the representation used for the facts, we propose two models (a) Vanilla Model (b) Convolution-based entity/relation cluster representation, for fact retrieval in the subsequent sections.

3.1 Vanilla Model

The entities and relationships of KG are encoded using DKRL, explained earlier. Let $e_i \in \mathbb{R}^m$ stand for the encoding of the entity i and $r_j \in \mathbb{R}^m$ stands for j^{th} relationship in the KG. The input text in the form of concatenated word vectors, $\boldsymbol{x} = (x_1, x_2, \ldots, x_T)$ is first encoded using an LSTM (Greff et al., 2015) module as follows,

$$h_t = f(x_t, h_{t-1})$$

and

$$o = \frac{1}{T}\sum_{t=1}^{T} h_t,$$

$h_t \in \mathbb{R}^n$ is the hidden state of the LSTM at time t, f is a non-linear function and T is the sequence length. Then a context vector is formed from o as follows,

$$C = \text{ReLU}(o^T W),$$

where, $W \in \mathbb{R}^{n \times m}$ represent the weight parameters. The same procedure is duplicated with separate LSTMs to form two seperate context vectors, one for entity retrieval (C_E) and one for relationship retrieval (C_R).

As the number of fact triples in a KG is in the order of millions in the vanilla model, we resort to generating attention over the entity and relation space separately. The fact is then formed using the retrieved entity and relation. The attention for the entity, e_i using entity context vector is given by

$$\alpha_{e_i} = \frac{\exp(C_E^T e_i)}{\sum_{j=0}^{|E|} \exp(C_E^T e_j)}$$

where $|E|$ is the number of entities in the KG.

Similarly the attention for a relation vector r_i is computed as

$$\alpha_{r_i} = \frac{\exp(C_R^T r_i)}{\sum_{j=0}^{|R|} \exp(C_R^T r_j)}$$

where $|R|$ is the number of relations in the KG. The final entity and relation vector retrieval is computed by the weighted sum with the attention

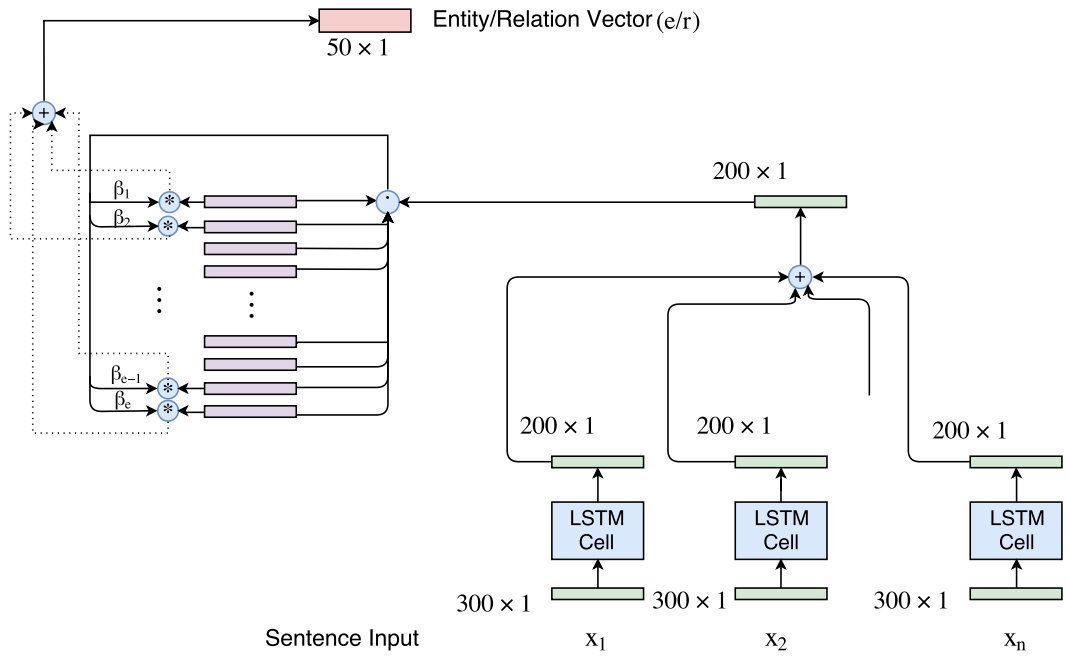


Figure 2: Vanilla Entity/Relationship Retrieval Block Diagram

values of individual retrieved entity/relation vectors.

$$e = \sum_{i=0}^{|E|} \alpha_{e_i} e_i \qquad r = \sum_{i=0}^{|R|} \alpha_{r_i} r_i$$

Figure 2 shows the schematic diagram for entity/relation retrieval. After the final entity and relation vectors are computed, we look forward to completion of the fact triple. The KG embedding technique used for the experiment is DKRL which inherently uses the TransE model assumption ($h+r \approx t$). Therefore, using the subject entity and relation we form the object entity as $t = e+r$. Thus the fact triplet retrieved is $\mathcal{F} = [e, r, e+r]$, where $\mathcal{F} \in \mathbb{R}^{3m}$. This retrieved fact information is concatenated along with the context vector (C) of input x obtained using LSTM module. The final classification label $\mathbf{y}$ is computed as follows,

$$\mathcal{F}^{'} = \mathrm{ReLU}(\mathcal{F}^{T} V)$$

$$\mathbf{y} = \mathrm{softmax}([\mathcal{F}^{'} : C]^{T} U)$$

where, $V \in \mathbb{R}^{3m \times u}$ and $U \in \mathbb{R}^{2u \times u}$ are model parameters to be learned. $\mathbf{y}$ is used to compute the cross entropy loss. We minimize this loss averaged across the training samples, to learn the various model parameters using stochastic gradient descent (Bottou, 2012). The final prediction $\mathbf{y}$, now includes information from both dataset specific samples and world knowledge to aid in enhanced performance. While jointly training the attention mechanism tunes itself to retrieve relevant facts that are required to do the final classification.

3.2 Pre-training KG Retrieval

The vanilla model attends over the entire entity/relation space which is not a good approach as we observe that the gradient for each attention value gets saturated easily. While training the classification and retrieval module together, the model tends to ignore the KG part and gradient propagates only through the classification module. This is expected to an extent as the most pertinent information for the task at hand comes from the training samples, only background aiding information comes from KG. After few epochs of training, the KG retrieved fact always converged to a fixed vector. To overcome this problem, we attempted pre-training KG retrieval part separately. A pre-trained KG model is used to retrieve the facts and then concatenate with the classification module, while we allow error to be propagate through the pre-trained model, at the time of joint training. We infer that KG doesn't return noise and has essential information for the task as the separate KG part alone shows significant performance (59% for News20 & 66% for SNLI). Figure 3 depicts the entire training scheme. This procedure solved the issue of gradient saturation in the KG retrieval part

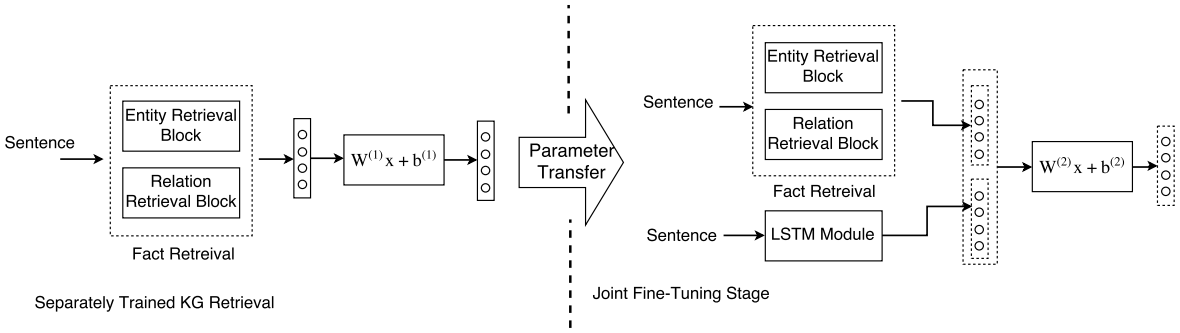


Figure 3: Separately Training Knowledge Graph Retrieval and Jointly Training the Full Model

at the time of joint training. However, the key problem of attention mechanism having to cover a large span of entities/relation, remained.

3.3 Convolution-based Entity and Relation Cluster Representation

In this section, we propose a mechanism to reduce the large number of entities/relationships over which attention has to be generated in the knowledge graph. We propose to reduce the attention space by learning the representation of similar entity/relation vectors and attending over them.

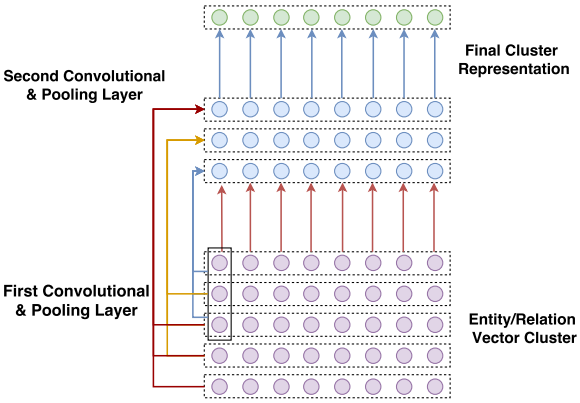


Figure 4: Convolution model cluster representation

In order to cluster similar entity/relation vectors, we used k-means clustering (Bishop, 2006) and formed l clusters with equal number of entity/relation vectors in each cluster. Each of the clusters were then encoded using convolutional filters. The output of the k-means clustering is a sequence of entity/relation vectors $\{e_1^T, e_2^T, \cdots, e_q^T\}$, where $e_i \in \mathbb{R}^m$. For each cluster these vectors were stacked to form $\mathcal{E}$ as the 2-D input to the CNN encoder, where $\mathcal{E} \in \mathbb{R}^{m \times q}$. During experimentation for finding a suitable filter shape, it was observed that using 2-D filters the model failed to converge at all. Therefore, we inferred that the latent representation of two different indices in the vector e_i, should not be tampered using convolution. We then resorted to use 1-D convolution filters which slide along only the columns of $\mathcal{E}$, as shown Figure 4. The stride length along y-axis is the window length k. The output of the convolution layer is expressed as,

$$\mathcal{E}'(i,j) = W^T[e_{i,j}, e_{i+1,j}, \ldots, e_{i+k-1,j}]^T$$

where, $\mathcal{E}'(i,j)$ is the $(i,j)^{th}$ element of the output matrix $\mathcal{E}'$ and $W \in \mathbb{R}^k$ is the convolution weight filter. A pooling layer followed the convolution layer in order to reduce the parameter space, we used 1-D window only along the y-axis similar to the convolutional kernel mentioned above. We used a two layered convolution network with the stride length k & max-pool windows n is adjusted to obtain output $E_i \in \mathbb{R}^m$, where i is the cluster index. Similar procedure of clustering followed by the encoding of the cluster entities is done for relations as well. Thus both the entity and relation space were reduced to contain fewer elements, one each for each cluster. After the compact entity space E and relation space R is formed, we followed the same steps as earlier for forming the attention, but now the training was more effective as the gradient was propagating effectively and was not choked by the large space. As the convolution architecture is also simultaneously trained, attention mechanism was not burdened as before, to learn over the large space of entities and relations.

Another point that needs to be mentioned here is regarding ranking/ordering items in the clusters, we have done experiments to verify the ordering does not affect the final result. We have verified this by randomly shuffling the entities/relationships in every clusters and the ac-

curacy output remained within an error bound of $\pm 0.5\%$. In various permutations, the representations learned by the convolution operator for clusters varies, but it does not affect the overall results. Regarding the interpretation of what convolution operator learns, the operator is applied along each dimension of the entity/relationship vector, to learn a representation of the clusters. This representation includes information from relevant entities in the cluster, as the relevant entities varies across tasks, the representation learned using convolution also adapts accordingly. It is analogous to learning relevant features from an image, in our case the convolution layer learns the features focusing on relevant entities/relations in a cluster pertaining to the task.

4 Experiments and Evaluations

Our experiments were designed to analyze whether a deep learning model is being improved when it has access to KG facts from a relevant source. The selection of knowledge graph has to be pertinent to the task at hand, as currently there is no single knowledge base that contains multiple kinds of information and can cater to all tasks. We illustrate with results that the performance of a deep learning model improves when it has access to relevant facts. We also illustrate that as the model learns faster with access to knowledge bases, we can train deep learning models with substantially less training data, without compromising on the accuracy. In the subsequent section we briefly describe the datasets and associated Knowledge Bases used.

Datasets and Relevant Knowledge Graphs

In our experiments, we have mainly used the popular text classification dataset 20Newsgroups (Lichman, 2013) and the Natural Language Inference dataset, Stanford Natural Language Inference (SNLI) corpus (Bowman et al., 2015). We have also done experiments on DBPedia ontology classification dataset[1], with a very strong baseline. These datasets are chosen as they share domain knowledge with two most popular knowledge bases, Freebase (FB15k) (Bollacker et al., 2008) and WordNet (WN18) (Bordes et al., 2013). The training and test size of the datasets are mentioned in Table 1.

[1] http://wiki.dbpedia.org/services-resources/dbpedia-data-set-2014

Dataset	Train Size	Test Size	# Classes
News20	16000	2000	20
SNLI	549367	9824	3
DBPedia	553,000	70,000	14

Table 1: Dataset Specifications

Freebase (FB15k) (Bollacker et al., 2008) contains facts about people, places and things (contains 14904 entities, 1345 relations & 4.9M fact triples), which is useful for text classification in 20Newsgroups (Lichman, 2013) dataset. On the other hand, WordNet (WN18) (Bordes et al., 2013) (has 40943 entities, 18 relations & 1.5M fact triples) contains facts about common day-to-day things (example: furniture includes bed), which can help in inference tasks like SNLI. Both the knowledge bases are directed graphs, due to fewer number of relations WN18 the entities are more likely to be connected using the same type of relations. For experiments relating to both the datasets 20Newsgroups & SNLI we used the standard LSTM as the classification module. As iterated earlier, our KG based fact retrieval is independent of the base model used. We show improvement in performance using the proposed models by KG fact retrieval. We use classification accuracy of the test set as our evaluation metric.

4.1 Experimental Setup

All experiments were carried on a Dell Precision Tower 7910 server with Quadro M5000 GPU with 8 GB of memory. The models were trained using the Adam's Optimizer (Kingma and Ba, 2014) in a stochastic gradient descent (Bottou, 2012) fashion. The models were implemented using TensorFlow (Abadi et al., 2015). The relevant hyperparameters are listed in Table 2. The word embeddings for the experiments were obtained using the pre-trained GloVe (Pennington et al., 2014)[2] vectors. For words missing in the pre-trained vectors, the local GloVe vectors which was trained on the corresponding dataset was used.

4.2 Results & Discussion

Table 3 shows the results of test accuracy of the various methods proposed on the datasets News20 & SNLI. We observe that incorporation of KG facts using the basic vanilla model improves the performance slightly, as the retrieval model was

[2] http://nlp.stanford.edu/data/glove.840B.300d.zip

Hyper-parameter	News20	SNLI
Batch size	256	1024
Learning rate	0.05	0.05
Word Vector Dimension	300	300
Sequence Length	300	85
LSTM hidden-state Dimension	200	200
KG Embedding Dimension	50	50
# Clusters	20	20
# Epochs	20	20

Table 2: Hyper-parameters which were used in experiments for News20 & SNLI datasets

not getting trained effectively. The convolution-based model shows significant improvement over the normal LSTM classification. While tuning the parameters of the convolution for clustered entities/relations it was observed that smaller stride length and longer max-pool window improved performance. For News20 dataset we show an improvement of almost 3% and for SNLI an improvement of almost 5%.

The work is motivated more from the perspective of whether incorporation of world knowledge will improve any deep learning model rather than beating the state-of-the-art performance. Although LSTM is used to encode the input for the model as well as the retrieval vector, as mentioned earlier, these two modules need not be same. For encoding the input any complex state-of-the-art model can be used. LSTM has also been used to generate the retrieval vector. For DBPedia ontology classification dataset, we have used a strong baseline of 98.6%, and after augmenting it with KG (Freebase) using convolution based model we saw an improvement of ∼0.2%. As the baseline is stronger, the improvement quantum has decreased. This is quite intuitive as complex models are self-sufficient in learning from the data by itself and therefore the room available for further improvement is relatively less. The improvement as observed in the experiments is significant in weaker learning models, however it is also capable of improving stronger baselines as is evident from the results of DBPedia dataset.

4.3 Reducing Dataset Size Requirements for Training Deep Learning Models

We hypothesized that as Knowledge Graph is feeding more information to the model, we can achieve better performance with less training data.

Model	Accuracy	
	News20	SNLI
Plain LSTM	66.75%	68.73%
Vanilla KG Retrieval	67.30%	69.20%
Convolution-based KG	**69.34%**	**73.10%**

Table 3: Test accuracy of approaches in News20 using FB15K & SNLI datasets using WN18

To verify this we have performed experiments on varying dataset fractions for 20Newsgroups dataset as shown in Figure 5. From the plot, we observe that KG augmented LSTM with 70% data outperforms the baseline model with full dataset support, thereby reducing the dependency on labeled data by 30%.

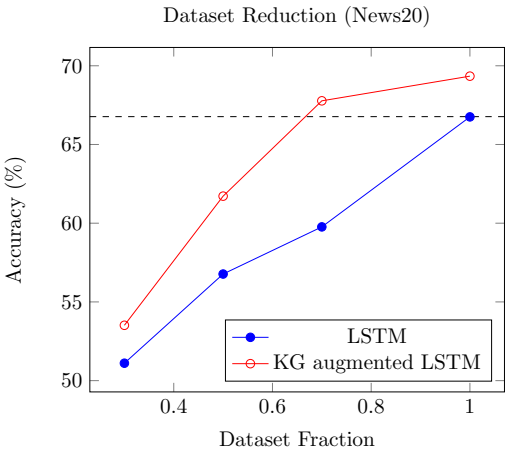


Figure 5: Accuracy Plot over dataset fractions for baseline and KG augmented model for News20

We also designed an experiment to compare the accuracy of the baseline model trained on full training data and compared it with the accuracy of the KG augmented model trained with just 70% of the training data for 20Newsgroups and SNLI datasets. The accuracy and training loss plots across training epochs is given in Figure 6. Even with just 70% of the data, KG augmented model is able to achieve better accuracy compared to the vanilla LSTM model trained on the full data. This clearly indicates that relevant information pertaining to the task is retrieved from the knowledge graph and the training loss reduction is not due to lesser data only. Also note that training loss is substantially less for KG LSTM compared to normal LSTM when the dataset size is reduced. This result is very promising, to reduce the large labeled training data requirement of large deep learning

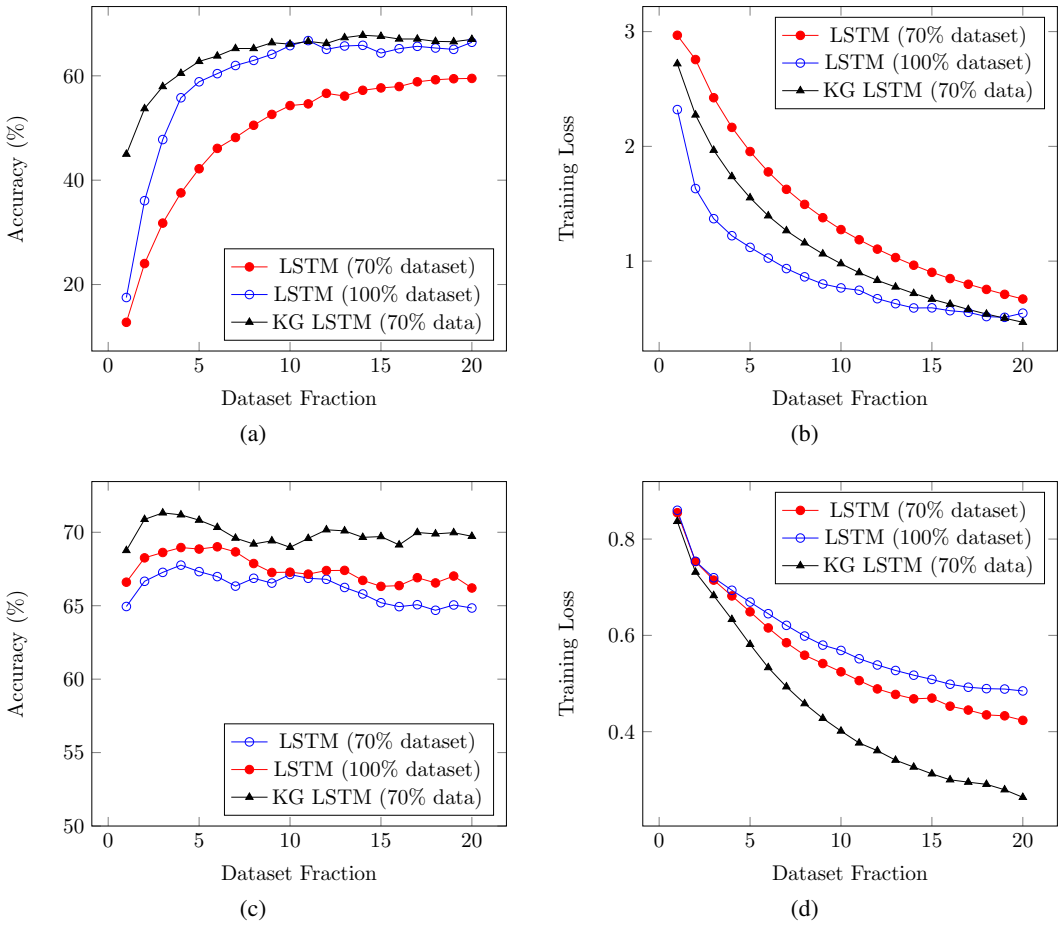


Figure 6: (a) Accuracy Plot over training epochs for LSTM (using full & 70% dataset) and KG augmented LSTM (using 70% dataset) for News20 task (b) Corresponding Training Loss plots for the aforementioned methods using News20 dataset (c) Accuracy Plot over training epochs for LSTM (using full & 70% dataset) and KG augmented LSTM (using 70% dataset) for SNLI task (d) Corresponding Training Loss plots for the aforementioned methods using SNLI dataset

models, which is hard to come by.

5 Relevant Previous Work

The basic idea of infusing general world knowledge for learning tasks, especially for natural language processing, has not been attempted before. For multi-label image classification, the use of KGs has been pursued recently by (Marino et al., 2016). In this work, they first obtain labels of the input data (using a different model), use these labels to populate features from the KG and in turn use these features back for the final classification. For NLP tasks the information needed may not necessarily depend on the final class, and we are directly using all the information available in the input for populating the relevant information from the knowledge graphs. Our attempt is very different from Transfer Learning (Pan and Yang, 2010). In Transfer Learning the focus is on training the model for one task and tuning the trained model to use it for another task. This is heavily dependent on the alignment between source task and destination task and transferred information is in the model. In our case, general world knowledge is being infused into the learning model for any given task. By the same logic, our work is different from domain adaptation (Glorot et al., 2011) as well. There has been attempts to use world knowledge (Song and Roth, 2017) for creating more labeled training data and providing distant supervision etc. Incorporating Inductive Biases (Ridgeway, 2016) based on the known information about a domain onto the structure of the learned models, is an active area of research. However our motivation and approach is fundamentally different from these works.

6 Conclusion & Future Work

In this work we have illustrated the need for incorporating world knowledge in training task specific models. We presented a novel convolution-based architecture to reduce the attention space over entities and relations that outperformed other models. With significant improvements over the vanilla baselines for two well known datasets, we have illustrated the efficacy of our proposed methods in enhancing the performance of deep learning models. We showcased that the proposed method can be used to reduce labeled training data requirements of deep learning models. Although in this work we focused only on NLP tasks and using LSTM as the baseline model, the proposed approach is applicable for other domain tasks as well, with more complicated deep learning models as baseline. To the best of our knowledge this is the first attempt at infusing general world knowledge for task specific training of deep learning models.

Being the first work of its kind, there is a lot of scope for improvement. A more sophisticated model which is able to retrieve facts more efficiently from millions of entries can be formulated. Currently we have focused only on a flat attention structure, a hierarchical attention mechanism would be more suitable. The model uses soft attention to enable training by simple stochastic gradient descent. Hard attention over facts using reinforcement learning can be pursued further. This will further help in selection of multi-facts, that are not of similar type but relevant to the task. The convolution based model, helped to reduce the space over entities and relationships over which attention had to be generated. However more sophisticated techniques using similarity based search (Wang et al., 2014a; Mu and Liu, 2017) can be pursued towards this purpose. The results from the initial experiments illustrates the effectiveness of our proposed approach, advocating further investigations in these directions.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, and Craig Citro et al. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. `http://tensorflow.org/`.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Christopher M Bishop. 2006. *Pattern recognition and machine learning*. springer.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. AcM, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*. pages 2787–2795.

Léon Bottou. 2012. *Stochastic Gradient Tricks*, Springer, volume 7700, page 430445. `https://www.microsoft.com/en-us/research/publication/stochastic-gradient-tricks/`.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Miao Fan, Qiang Zhou, Emily Chang, and Thomas Fang Zheng. 2014. Transition-based knowledge graph embedding with relational mapping properties. In *PACLIC*. pages 328–337.

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

Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. 2015. Lstm: A search space odyssey. *arXiv preprint arXiv:1503.04069* .

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *ACL (1)*. pages 687–696.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* .

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

M. Lichman. 2013. UCI machine learning repository. `http://archive.ics.uci.edu/ml`.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* .

Bill MacCartney. 2009. *Natural language inference*. Ph.D. thesis, Citeseer.

Kenneth Marino, Ruslan Salakhutdinov, and Abhinav Gupta. 2016. The more you know: Using knowledge graphs for image classification. *arXiv preprint arXiv:1612.04844* .

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. 1990. Introduction to wordnet: An on-line lexical database. *International journal of lexicography* 3(4):235–244.

Tom M Mitchell, William W Cohen, Estevam R Hruschka Jr, Partha Pratim Talukdar, Justin Betteridge, Andrew Carlson, Bhavana Dalvi Mishra, Matthew Gardner, Bryan Kisiel, Jayant Krishnamurthy, et al. 2015. Never ending learning. In *AAAI*. pages 2302–2310.

Yadong Mu and Zhu Liu. 2017. Deep hashing: A joint approach for image signature learning. In *AAAI*. pages 2380–2386.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016a. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE* 104(1):11–33.

Maximilian Nickel, Lorenzo Rosasco, Tomaso A Poggio, et al. 2016b. Holographic embeddings of knowledge graphs. In *AAAI*. pages 1955–1961.

Feng Niu, Ce Zhang, Christopher Ré, and Jude W Shavlik. 2012. Deepdive: Web-scale knowledge-base construction using statistical learning and inference. *VLDS* 12:25–28.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22(10):1345–1359.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. volume 14, pages 1532–1543.

Karl Ridgeway. 2016. A survey of inductive biases for factorial representation-learning. *arXiv preprint arXiv:1612.05299* .

Baoxu Shi and Tim Weninger. 2017. Proje: Embedding projection for knowledge graph completion. In *AAAI*. pages 1236–1242.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*. pages 926–934.

Yangqiu Song and Dan Roth. 2017. Machine learning with world knowledge: The position and survey. *arXiv preprint arXiv:1705.02908* .

Jingdong Wang, Heng Tao Shen, Jingkuan Song, and Jianqiu Ji. 2014a. Hashing for similarity search: A survey. *arXiv preprint arXiv:1408.2927* .

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014b. Knowledge graph embedding by translating on hyperplanes. In *AAAI*. pages 1112–1119.

Han Xiao, Minlie Huang, Yu Hao, and Xiaoyan Zhu. 2015a. Transg: A generative mixture model for knowledge graph embedding. *arXiv preprint arXiv:1509.05488* .

Han Xiao, Minlie Huang, Lian Meng, and Xiaoyan Zhu. 2017. Ssp: Semantic space projection for knowledge graph embedding with text descriptions. In *AAAI*. pages 3104–3110.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2015b. From one point to a manifold: Knowledge graph embedding for precise link prediction. *arXiv preprint arXiv:1512.04792* .

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *AAAI*. pages 2659–2665.

Huaping Zhong, Jianwen Zhang, Zhen Wang, Hai Wan, and Zheng Chen. 2015. Aligning knowledge and text embeddings by entity descriptions. In *EMNLP*. pages 267–272.

Comparing Constraints for Taxonomic Organization

Anne Cocos[*], **Marianna Apidianaki**[*†], and **Chris Callison-Burch**[*]

[*] Department of Computer and Information Science, University of Pennsylvania
[†] LIMSI, CNRS, Université Paris-Saclay, 91403 Orsay
{acocos,marapi,ccb}@seas.upenn.edu

Abstract

Building a taxonomy from the ground up involves several sub-tasks: selecting terms to include, predicting semantic relations between terms, and selecting a subset of relational instances to keep, given constraints on the taxonomy graph. Methods for this final step – taxonomic organization – vary both in terms of the constraints they impose, and whether they enable discovery of synonymous terms. It is hard to isolate the impact of these factors on the quality of the resulting taxonomy because organization methods are rarely compared directly. In this paper, we present a head-to-head comparison of six taxonomic organization algorithms that vary with respect to their structural and transitivity constraints, and treatment of synonymy. We find that while transitive algorithms out-perform their non-transitive counterparts, the top-performing transitive algorithm is prohibitively slow for taxonomies with as few as 50 entities. We propose a simple modification to a non-transitive optimum branching algorithm to explicitly incorporate synonymy, resulting in a method that is substantially faster than the best transitive algorithm while giving complementary performance.

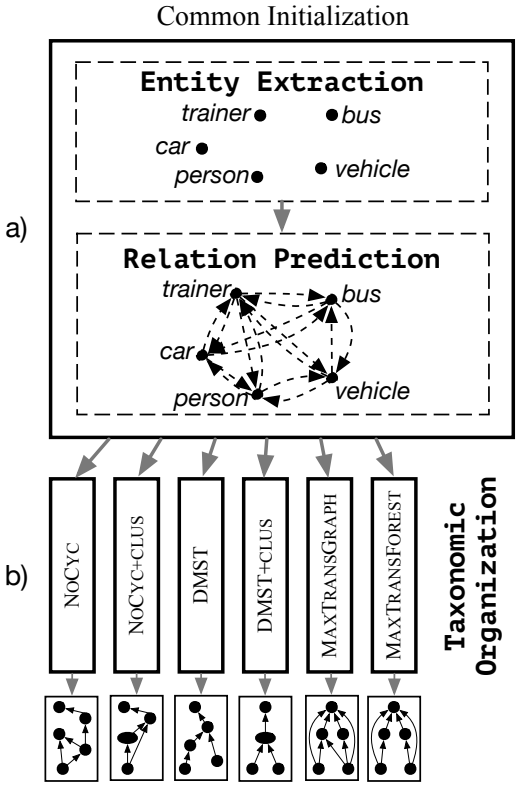


Figure 1: In this study we compare algorithms for taxonomic organization. We first (a) run entity extraction and pairwise relation prediction as a common initialization; we then (b) feed the resulting graphs as identical input to six taxonomic organization algorithms. We evaluate the impact of varied structural constraints between algorithms.

1 Introduction

Many words and phrases fit within a natural semantic hierarchy: a *mobile* is a type of *telephone*, which in turn is a *communications device* and an *object*. Taxonomies, which encode this knowledge, are important resources for natural language understanding systems.

There is ongoing interest in developing methods to build taxonomic resources automatically (Bordea et al., 2015, 2016). Although several widely-used general ontologies (e.g. WordNet (Miller, 1995)) and domain-specific ontologies (e.g. Unified Medical Language System (UMLS) (Bodenreider, 2004)) exist, these resources are handcrafted and therefore expensive to update or expand. Automatic taxonomy induction enables the construction of taxonomic resources at scale in new languages and domains. Further, there is evidence that it is useful to build dynamic or context-specific taxonomies extemporaneously for some applications (Do and Roth, 2010).

Proceedings of NAACL-HLT 2018, pages 323–333
New Orleans, Louisiana, June 1 - 6, 2018.

Taxonomy induction involves three sub-tasks: *entity extraction*, *relation prediction*, and *taxonomic organization*. In many cases these subtasks are undertaken sequentially to build a taxonomy from the ground up. While many works directly compare methods for relation prediction (e.g. Turney and Mohammad (2015), Shwartz et al. (2017) and others), none directly compare methods for the final taxonomic organization step with varying constraints. Each paper that proposes a taxonomic organization method starts with its own set of predicted relations, making it impossible to determine – even with benchmark datasets – the extent to which improvements in identifying ground-truth relations are due to (a) better relation prediction, or (b) better taxonomic organization.

In this work, we present an empirical apples-to-apples comparison of six algorithms for unsupervised taxonomic organization. The algorithms vary along three axes: whether they impose transitivity constraints on the taxonomic graph, whether they specify that the final graph structure be a directed acyclic graph (DAG) or tree/forest, and whether they identify 'clusters' of synonymous terms. In each case we begin with the same sets of terms and predicted relations (see Figure 1). This makes it possible to address several research questions. First, which combination of these factors produces a taxonomy that most closely mirrors a set of ground-truth taxonomic relations? Second, which algorithms are efficient enough in practice to run on large term sets? And third, how robust is each algorithm to noise in the predicted relations used as input?

We find that while transitive algorithms perform better than non-transitive algorithms given the same constraints on graph structure, the best-performing transitive algorithm is prohibitively slow to use on input with as few as 50 nodes. By modifying a commonly-used optimum branching algorithm to consolidate clusters of predicted synonyms into a single graph node, we show that it is possible to achieve complementary performance levels with an average runtime that is faster by orders of magnitude.

2 General Framework for Taxonomy Induction

The problem of taxonomy induction can be summarized via three core sub-tasks. While all systems that build taxonomies automatically must address each of these tasks, the sequence and manner in which they are addressed varies. In the most straightforward case, the core tasks are viewed as orthogonal and carried out sequentially. They are:

1. **Entity Extraction:** Identify a set of entities E (i.e. word types, synsets, etc) that will become nodes in the eventual taxonomy graph.

2. **Relation Prediction:** Predict the presence or absence of a directed semantic relation (hypernymy or entailment) between each pair of nodes, $(e_i, e_j) \in E \times E$. The outputs are (a) a set of potential edges $R \subseteq E \times E$, where we use the notation $r_{ij} \in R$ to signify the relational instance, or edge, (e_i, e_j), and (b) relation scores $s(r_{ij})$ for each edge derived from the classifier's predicted likelihood that the relational instance exists.

3. **Taxonomic Organization:** Select a subset of the predicted edges, $\hat{R} \subseteq R$, that produces a high sum of scores, $\sum_{r \in \hat{R}} s(r_{ij})$, subject to structural constraints. The final output is the graph $\hat{G}(E, \hat{R})$.

Structural constraints dictate what can be considered a *valid* or *invalid* combination of edges in a taxonomic graph (Do and Roth, 2010). Two structural constraints frequently imposed are that the final graph be a DAG, or that the final graph be a tree/forest.[1] Examples of algorithms that produce DAG structures are the longest-path algorithm of Kozareva and Hovy (2010), the *ContrastMedium* approach of Faralli et al. (2017), and the random cycle-breaking method used in (Panchenko et al., 2016) and Faralli et al. (2015). We experiment with a variation of the last one here, which we call NoCyc. To produce tree-structured taxonomies, most researchers (including us) use algorithms for finding the maximally-weighted rooted tree spanning a directed graph (DMST). Examples of prior work following this approach are Navigli et al. (2011) and Bansal et al. (2014).

Another dimension along which taxonomy organization approaches differ is whether they explicitly require the set of chosen relational instances $\hat{R}$ to be fully transitive. The transitivity constraint dictates that if (*beetle* IS-A *insect*) is selected as part of $\hat{R}$, and (*insect* IS-A *organism*) is

[1] WLOG, the tree and forest constraints are identical, as a dummy root node can be attached to the root of each component in a forest to produce a tree.

selected as part of $\hat{R}$, then (*beetle* IS-A *organism*) must also be selected. Two methods that impose such transitivity constraints are the MAXTRANSGRAPH and MAXTRANSFOREST methods of Berant et al. (2015), both of which we experiment with here.

A final consideration when choosing a taxonomy organization algorithm is whether the method should enable the consolidation of synonyms into a single taxonomic entity. Synonym sets, or *synsets*, are present as nodes in the WordNet graph (Miller, 1995). Potential advantages to using synonym sets, rather than individual terms, as nodes include the ability to model polysemy (*horse* means one thing when grouped with its synonym *cavalry* and another entirely when grouped with *sawhorse*), and the ability to be more precise in defining relations. A few early taxonomy induction approaches incorporated synonym clustering (e.g. Lin and Pantel (2002) and Pantel and Ravichandran (2004)). The two transitive algorithms that we analyze here, MAXTRANSGRAPH and MAXTRANSFOREST, also consolidate equivalent terms into a single node.

3 Taxonomic Organization Algorithms

The six algorithms that we compare differ along the three dimensions just described, namely, the structural constraints imposed (DAG or tree), whether transitivity is required, and whether synonyms are combined into a single taxonomy node (Figure 2). Here we provide a short description of each.

	No Transitivity	**Transitivity**
DAG	NOCYC NOCYC+CLUS	MAXTRANSGRAPH
Tree / Forest	DMST* DMST+CLUS	MAXTRANSFOREST

Figure 2: Classification of the algorithms compared in our study. The starred DMST algorithm is the only one that does not support consolidation of synonyms into clusters.

3.1 NOCYC: Organizing a DAG

The no-cycles method, which we abbreviate as NOCYC, is a simple method for constructing a **DAG** with high score from a set of predicted relational edges. It is **not transitive**.

The algorithm works as follows. From the set R of all predicted hypernym relations, we first filter out of the graph $G(E, R)$ any edges with score $s(r_{ij})$ less than a tunable threshold τ. Next, we break any cycles by finding strongly connected components (SCC) in the graph (i.e. a subset of nodes such that each node in the subset has a path to every other node in the subset), and iteratively removing the lowest-scoring edge from each SCC until all cycles are broken. This implementation is slightly different from that of Faralli et al. (2015) and Panchenko et al. (2016), where cycles were broken by removing cycle edges randomly. The search for SCCs in each iteration is linear using Tarjan's algorithm (Tarjan, 1972).

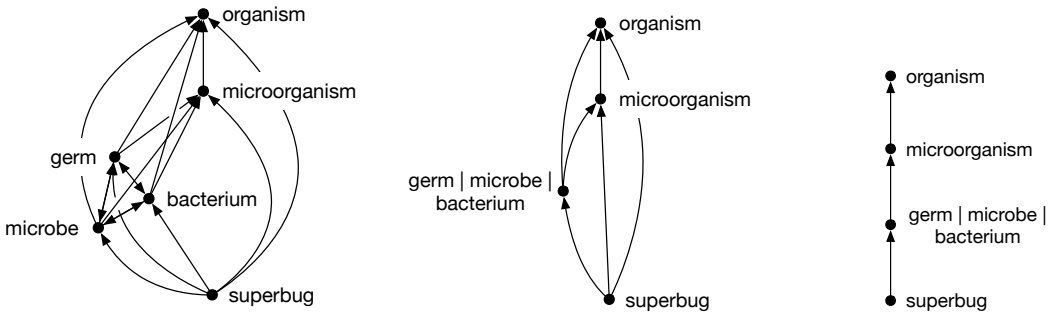
The NOCYC algorithm does not explicitly cluster synonyms, but we can find synonyms in the resulting graph implicitly as follows. If we assume all synonymous terms share the same direct hypernyms and direct hyponyms, we can find such pairs by taking the transitive reduction[2] of the resulting graph $\hat{G} = (E, \hat{R})$, and grouping all pairs of terms that have identical sets of direct hypernyms and hyponyms in the transitive reduction.

While NOCYC itself does support finding synonyms within the graph implicitly, we also experiment with an explicit synonym-clustering version, NOCYC+CLUS. We modify NOCYC by collapsing into a single node all subsets of nodes predicted to be synonym clusters, using a method described in Section 4.2.2, prior to executing the cycle breaking algorithm.

3.2 DMST: Organizing a Tree

Our second method selects hypernym edges for the taxonomy by using the Chu-Liu-Edmonds optimum branching algorithm (Chu and Liu, 1965; Edmonds, 1967) to solve the directed analog of the maximum spanning tree problem (DMST). It constrains the final graph to be a **tree** and is **not transitive**.

[2]In the *transitive closure* of a graph, each node e_i is directly connected by a single edge to every node e_j to which it has a path. The *transitive reduction* can be obtained for a graph G by removing all edges from G that do not change its transitive closure. The transitive reduction of a DAG is unique (Aho et al., 1972).



(a) Selected edges in $\hat{R}$. (b) SCC from 3a collapsed (c) Transitive reduction of 3b

Figure 3: An example transitive taxonomy graph as output by MAXTRANSGRAPH or MAXTRANSFOREST is given in 3a. Its strongly-connected component (SCC) is collapsed into a single node of synonyms to create the DAG in 3b. Finally, its transitive reduction is in 3c. Because flipping edges in the result produces a tree rooted at *organism*, the graph in 3a is called forest-reducible.

The algorithm works by adding a dummy root node e_{ROOT} to E, and an edge from e_{ROOT} to every other node e_i in the graph. We then use Chu-Liu-Edmonds to find the directed tree rooted at e_{ROOT} that spans all nodes in E and has the maximal sum of scores. Note that until now we have considered edges in taxonomy graphs to point from hyponyms to hypernyms; in this case we must switch the order, so that the spanning tree starts at the most general level of the hierarchy and reaches down to the leaves. Chu-Liu-Edmonds finds the DMST efficiently in polynomial time (Tarjan, 1977).

Because DMST requires the final graph to be a tree, there is no implicit way to find synonyms within the taxonomy graphs it generates. As with NOCYC, we test a modification called DMST+CLUS that collapses predicted synonym clusters into a single graph node prior to running the DMST algorithm (see Section 4.2.2).

3.3 MAXTRANSGRAPH: Organizing a Transitive DAG

The first **transitive** algorithm we evaluate is MAXTRANSGRAPH (Berant et al., 2012, 2015), which constrains the graph structure to be a **DAG**. MAXTRANSGRAPH was originally designed for building taxonomies of entailment relations (which can be subclassified as either synonyms or hypernyms) and is solved using integer linear programming (ILP). Rather than using classifier scores directly as input, MAXTRANSGRAPH first computes a weight between each term pair (e_i, e_j) that is equal to the classifier score minus a tunable parameter: $w_{ij} = s(r_{ij}) - \lambda$. The purpose of modifying scores this way is efficiency; MAXTRANSGRAPH solves its optimization on each connected component of the graph independently, where components are constructed by considering only positively-weighted edges in the graph. Increasing λ increases sparsity and decreases runtime.

The objective of the ILP is to maximize the weights of selected relations, while requiring that the graph respects transitivity. Berant et al. (2012) proved this problem is NP-hard and provided an ILP formulation for it as follows. Let x_{ij} be a binary variable that indicates whether edge (e_i, e_j) is in the subset of selected edges, $\hat{R}$.

$$\begin{aligned} \max_{x} \quad & \sum_{i \neq j} w_{ij} x_{ij} \\ \text{s.t.} \quad & \forall e_i, e_j, e_k \in E, \; x_{ij} + x_{jk} - x_{ik} \leq 1 \\ & \forall e_i, e_j \in E, \; x_{ij} \in \{0, 1\} \end{aligned} \tag{1}$$

The objective maximizes the sum of edge weights where the edge is 'turned on' (i.e. $x_{ij} = 1$). The first constraint enforces transitivity, i.e. for every triple of nodes (e_i, e_j, e_k), if edge $(e_i, e_j) \in \hat{R}$ and edge $(e_j, e_k) \in \hat{R}$, then edge $(e_i, e_k) \in \hat{R}$. The second constraint specifies that all x_{ij} are binary. The number of variables is $O(|E|^2)$ and number of constraints is $O(|E|^3)$.

MAXTRANSGRAPH assumes that cycles of entailment relations in the resulting graph $G(E, \hat{R})$ comprise cycles of synonyms, and that the remaining edges which are not part of a cycle are hypernym edges. Because the resulting graph must be transitive, all cycles of three or more nodes are

cliques, in which each node is directly connected to every other. Once every SCC is collapsed into a single synonym cluster node, the transitive reduction of the resulting graph is a DAG (Figure 3b).

3.4 MAXTRANSFOREST: Organizing a Transitive Forest

The final algorithm we evaluate is MAXTRANSFOREST (Berant et al., 2012, 2015), which like MAXTRANSGRAPH is **transitive**, but produces a **forest/tree** structure.

MAXTRANSFOREST is nearly identical to MAXTRANSGRAPH, with the addition of one constraint that imposes its forest structure. More specifically, the graphs produced by MAXTRANSFOREST are *forest reducible*. A forest reducible graph is one where, after collapsing every SCC into a single node, the transitive reduction of the result is a forest (see Figure 3).

In practice, the forest reducibility constraint is enforced by applying one additional constraint to the ILP in Equation 1:

$$\forall e_i, e_j, e_k \in E \;\; x_{ij} + x_{ik} - x_{jk} - x_{kj} \leq 1 \quad (2)$$

This constraint says that each node e_i can have only a single parent. If relations r_{ij} and r_{ik} are in $\hat{R}$, then either e_j is the parent of e_k or vice versa; e_i may not have two parents that are not related via a hypernym relationship. Like MAXTRANSGRAPH, the number of variables is $O(|E|^2)$ and number of constraints is $O(|E|^3)$. Also like MAXTRANSGRAPH, cycles in the resulting graph are assumed to constitute clusters of synonymous terms.

4 Experimental Setup

In order to directly compare the organization algorithms described, we organize our experiments as follows. We first run entity extraction (Section 4.1) and relation prediction (Section 4.2) as a common initialization for all algorithms. Then, we take the edge scores output by the relation prediction step and feed them to each taxonomic organization algorithm (Section 4.3). Finally, we compare the output from each algorithm. Here we describe the initialization steps in more detail.

4.1 Entity Extraction

We extract sets of entities from the Paraphrase Database (PPDB) (Ganitkevitch et al., 2013; Pavlick et al., 2015). Our goal is to construct *local taxonomies*, where each entity set E consists of terms sharing a common target paraphrase. For example, a local taxonomy centered around the target *coach* might contain entities *bus*, *vehicle*, *trainer*, *person*, *car*, and *railcar*. The local taxonomy for a target word does not contain the target word itself.

We build a dataset for constructing local taxonomies centered around 50 target nouns drawn from the 2010 SemEval word sense induction dataset (Manandhar et al., 2010). For each target noun, we extract as taxonomy terms the set of PPDB paraphrases having a PPDB2.0SCORE of at least 2.0 with the target.[3] The number of entities in each local taxonomy ranges from 13 to 126, with a median of 40 entities per set. We hold out 5 local taxonomies to tune parameters for NOCYC, MAXTRANSGRAPH, and MAXTRANSFOREST, and use the remaining 45 as our test set.

Because they consist of related terms centered around a common paraphrase, there are several semantic relations present among these entity sets in addition to hypernymy and synonymy. We analyze the overlap between all pairs of terms appearing in our local taxonomies and in WordNet, and find that the distribution of relation types among the overlapping pairs is 6.0% hypernym/hyponym, 1.3% synonym, 0.1% meronym/holonym, 3.1% coordinate terms (sharing a common direct hypernym), and 89.5% none of these.

4.2 Relation Prediction

Having extracted a set of entities, the next step in our initialization process is to make pairwise relation predictions for each pair of terms (e_i, e_j) that exist within an entity set E. The different organization algorithms we compare take predicted synonym and/or hypernym edge scores as input. Here we describe the methods we use to generate these scores.

4.2.1 Hypernym Prediction

For hypernym prediction, we adopt the state-of-the-art HypeNET method of Shwartz et al. (2016). HypeNET integrates distributional (Lin and Pantel, 2002; Roller et al., 2014; Levy et al., 2015; Benotto, 2015) and path-based (Hearst, 1992; Snow et al., 2004; Nakashole et al., 2012) ap-

[3] The PPDB2.0SCORE is a supervised metric designed to correlate with human judgements of paraphrase quality (Pavlick et al., 2015).

proaches to hypernym prediction. It uses a recurrent neural network to represent the set of observed dependency paths connecting an input word pair, and concatenates this representation with distributional word embeddings to produce a set of features for predicting hypernymy.

We create a dataset of noun pairs for training and evaluating the HypeNET model. It combines noun pairs from four benchmark relation prediction datasets (BLESS (Baroni and Lenci, 2011), ROOT09 (Santus et al., 2016), EVALution (Santus et al., 2015), and K&H+N (Necsulescu et al., 2015)) with a set of related and unrelated noun pairs extracted from PPDB. Since each of these is a multi-class dataset, we binarize the data by labeling noun pairs with a hypernym relation as positive instances, and all others as negative. The combined benchmark+PPDB training set contains 76,152 noun pairs with a 1:4 hypernym:non-hypernym ratio, and the evaluation set contains 29,051 pairs. We ensure lexical separation from our taxonomy induction dataset; no terms in the classifier training set appear in any of the local taxonomies. We train HypeNET using our 76K-pair test set, and provide the results of evaluation on the 29K-pair test set in Table 1. The trained model achieves an overall average F1-score of 0.93 on the entire benchmark+PPDB test set. The full details of our dataset creation and classifier training are provided in the supplementary material.

Finally, we use the trained model to predict hypernym likelihoods for each potential edge r_{ij} in one of our local taxonomies, corresponding to an ordered pair of terms (e_i, e_j) that appear together in one of the 50 entity sets. We assign a hypernym score $s_h(r_{ij})$ to each potential directed edge that equals the HypeNET predicted likelihood for that pair of terms.

4.2.2 Synonym Prediction and Cluster Formation

We predict synonymy between noun pairs using distributional similarity, operationalized as the cosine similarity of PARAGRAM (Wieting et al., 2015) word embeddings.[4] We use PARAGRAM vectors because they perform well in semantic similarity tasks, and because they were originally extracted from PPDB and thus have 100% coverage of our entity sets. The synonym score $s_s(r_{ij})$ for a potential edge r_{ij} between entities (e_i, e_j) is simply the cosine similarity of their PARAGRAM embeddings.

[4]We also tried using HypeNET to predict synonym relations, but results were significantly worse.

Dataset	# Test Inst.	% Hyp.	Avg F1 (hyp.)	% Syn.	Avg F1 (syn.)
PPDB	3,000	20.1	.777	24.7	.707
BLESS	6,637	5.3	.978	0	–
EVALution	1,846	24.5	.763	15.1	.797
K&H+N	14,377	7.3	.988	0	–
ROOT09	3,191	26.3	.808	0	–

Table 1: Evaluation of the HypeNET hypernym classifier and the PARAGRAM synonym classifier on the PPDB test set and four benchmark test sets. We report micro-averaged F1-scores for positive and negative instances in the test sets.

We also tune a synonymy threshold for the purpose of consolidating clusters of synonymous terms into a single node for DMST+CLUS and NOCYC+CLUS (see Section 4.3). We tune threshold $\tau = 0.76$ over the benchmark+PPDB training set (binarized for synonymy) such that we predict a term pair (e_i, e_j) to be synonymous if $s_s(r_{ij}) \geq \tau$. When evaluated over the test sets, this method achieves weighted average F1-scores of 0.707 and 0.797 for predicting synonyms in the PPDB and EVALution test sets respectively (Table 1).

Target word	Clustered Entities
field	[(topic, issue, subject matter), (respect, regard), (battlefield, battleground), (outside, exterior), (territory, land), (domain, purview, sphere, ambit, realm, area, fields)]
address	[(directorate, direction), (administration, management), (answer, response), (discourse, speech), (treat, handling), (domicile, residence)]
innovation	[(novelty, imagination, creativity, newness), (modernization, modernisation), (regeneration, renewal, renovation, rejuvenation)]

Table 2: Examples of clustered entities produced using the PARAGRAM vector cosine similarity threshold of 0.76.

4.3 Input to organization algorithms

Finally, we use the calculated hypernym and synonym scores $s_h(r_{ij})$ and $s_s(r_{ij})$ to initialize each organization algorithm as follows.

NOCYC and DMST: We use the hypernym scores as input, setting $s(r_{ij}) = s_h(r_{ij})$ for all

	Algorithm	Constraint	Hypernyms			Synonyms			Combined		
			P	R	F	P	R	F	P	R	F
Baseline Methods											
	RANDOM	(none)	.036	.235	.061	.013	.034	.018	.033	.173	.054
	MAXTRANSFOREST**	Tree/Forest	.214	.758	.325	.707	.585	.586	.255	.708	.366
	DMST**	Tree/Forest	.411	.661	.470	0.	0.	0.	.411	.469	.418
Basic Methods											
Transitive	MAXTRANSGRAPH	DAG	.123	.529	.193	.727	.028	.040	.126	.375	.182
	MAXTRANSFOREST	Tree/Forest	.147	.473	.217	.353	.091	.121	.155	.365	.210
Non-Transitive	NOCYC	DAG	.104	.596	.172	.119	.013	.014	.101	.415	.158
	DMST	Tree/Forest	.192	.195	.178	0.0	0.0	0.0	.192	.131	.147
Clustering Variations											
Non-Transitive	NOCYC+CLUS	DAG	.081	.562	.138	.232	.368	.234	.091	.520	.149
	DMST+CLUS	Tree/Forest	.165	.204	.168	.304	.364	.266	.199	.265	.201

Table 3: Precision, recall, and F1 of hypernym, synonym, and all (relation-specific) edges for each method. Metrics are weighted averages over the 45 local taxonomies in the test set, where each taxonomy's result is weighted by its number of nodes. Starred methods indicate an oracle, where the weight of edges appearing in WordNet is set to 1 at input.

potential edges.

NOCYC+CLUS and DMST+CLUS: Initialization for these algorithms requires two steps. First, we collapse clusters of likely synonyms into a single entity as follows. For each local taxonomy, we create a graph with the extracted terms as nodes, and add an edge between every pair of terms having $s_s(r_{ij}) \geq \tau$ (the threshold tuned on our training set). We take the resulting connected components as the final entity set E. See examples of synonyms clustered by this method in Table 2.

Next, we calculate scores $s(r_{ij})$ for each pair of entities. When e_i and e_j are single-term entities (i.e. not synonym clusters), we simply set $s(r_{ij}) = s_h(r_{ij})$. To obtain an edge score when one or both nodes is a cluster, we simply calculate the average hypernym score over every pair of terms (t_m, t_n) such that $t_m \in e_i$ and $t_n \in e_j$:

$$s(r_{ij}) = \frac{\sum_{t_m \in e_i; t_n \in e_j} s_h(r_{mn})}{|e_i| + |e_j|}$$

MAXTRANSGRAPH and MAXTRANSFOREST: Since these algorithms are designed to use entailment relation predictions as input, we set the score of each edge to be the maximum of the synonym and hypernym scores: $s(r_{ij}) = \max(s_h(r_{ij}), s_s(r_{ij}))$. Intuitively, this reflects the idea that entailment can be sub-classified as synonymy *or* hypernymy.

5 Experiments

We conduct experiments aimed at addressing three primary research questions: (1) How does each taxonomic organization algorithm perform? In particular, how do DAG algorithms compare to tree-constrained ones, and how do transitive algorithms compare to their non-transitive counterparts? (2) Are any algorithms, particularly the ILP methods, too slow to use on large sets of terms? (3) Given that hypernym relation prediction is far from perfect, how robust is each algorithm to noise in the predicted relations?

5.1 Head-to-head Algorithm Comparison

In our first experiment, we predict PPDB local taxonomies for the 45 target nouns in our test set using each of the six algorithms after the initialization described in Section 4. In keeping with current work on this topic (Bordea et al., 2015, 2016), we evaluate the taxonomy organization algorithms' performance by calculating precision, recall, and F1-score of WordNet 3.0 hypernym and synonym edges for the 93% of PPDB taxonomy terms that are in WordNet. When evaluating hypernym edges we consider both di-

rect and transitive hypernym edges. We report hypernym-specific scores – where the set of ground-truth edges considers just WordNet hypernyms – synonym-specific scores, and combined scores – where all WordNet hypernym and synonym edges are taken as ground truth, and a predicted edge must have the correct start node, end node, and relation type to be correct. Results are reported in Table 3. We compare the results of the six algorithms to two types of baselines. As a lower bound, we implement a random baseline where edges are selected randomly with likelihood tuned on the benchmark+PPDB training set. As an upper bound, we run 'oracle' versions of MAXTRANSFOREST and DMST where we set the score of any edge appearing in WordNet to 1.

The transitive, tree-constrained MAXTRANSFOREST algorithm achieves the best average combined F-score (0.21) over all the local taxonomies, followed closely by the non-transitive, tree-constrained clustering method DMST+CLUS (0.20). These two methods, which are the only two tree-constrained methods that incorporate synonymy, outperform all DAG-constrained methods on this dataset. While they perform similarly in terms of combined F-score, their results are complementary; MAXTRANSFOREST obtains a relatively high score on hypernym edges and lower score for synonym edges, while for DMST+CLUS the results are reversed.

In general, these results suggest that consolidating synonyms into a single node helps tree-constrained methods by improving recall of both hypernym and synonym edges (DMST vs DMST+CLUS), but the same is not true for DAG-constrained methods. To understand why, we examine the output taxonomies. The average depth of the DAG taxonomies is greater than that of the tree taxonomies. When incorrect hyponym attachments are made in a deep taxonomy, the errors in transitive hypernym links can be magnified, which is evident in the low hypernym precision of NOCYC and NOCYC+CLUS. Synonym clustering prior to NOCYC+CLUS can magnify errors further, as synonyms are dragged into the incorrect hypernym relationships (see the NOCYC+CLUS example in Figure 4, where *telephone* is dragged along with *phone* into incorrect hypernym relations with *battery* and *pile*). For the shallower tree-constrained graph outputs, finding correct synonym relations helps the overall accuracy without inducing as many incorrect hypernym relations.

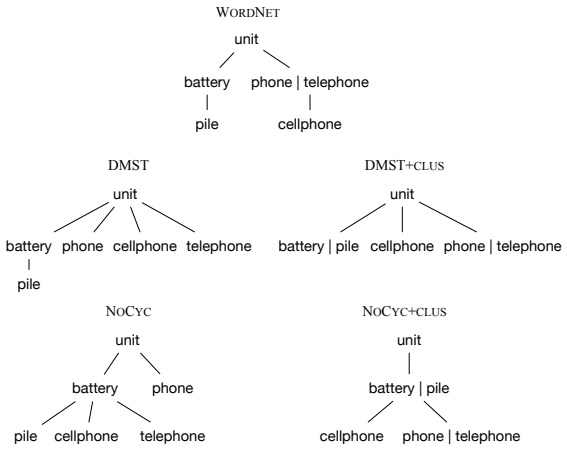


Figure 4: Examples from a portion of the local taxonomy for *cell*, with the WordNet gold standard compared to four of the algorithms' output. The bar notation denotes synonym clusters.

Finally, we note that transitive algorithms consistently out-perform their non-transitive counterparts. For the DAG-constrained algorithms, the transitive version, MAXTRANSGRAPH, improves precision of hypernym and synonym edges over its non-transitive counterparts NOCYC and NOCYC+CLUS. For the tree-constrained algorithms, the transitive MAXTRANSFOREST substantially improves recall of hypernym edges over its non-transitive counterparts DMST and DMST+CLUS.

5.2 Assessing Runtime

Next, we address the question of whether all algorithms are fast enough to be useful in practice. We record the runtime for each algorithm on each local taxonomy, and note the number of runs that timed out at 5 minutes. Results are in Table 4.

Method	Avg Runtime	% Timeout
MAXTRANSGRAPH	0.31	0.0
MAXTRANSFOREST	136.83	24.4
NOCYC	2.04	0.0
DMST	0.04	0.0
NOCYC+CLUS	6.41	0.0
DMST+CLUS	0.02	0.0

Table 4: Average runtime (seconds) over all 45 targets, and percent of targets for which runtime exceeded 5 minutes, by algorithm.

MAXTRANSFOREST, while most accurate on hypernyms and overall, is too slow to be use-

ful on large inputs. The average runtime over all local taxonomies was over two minutes, and the runtime on local taxonomies with as few as 50 nodes reached the five minute limit. Meanwhile, DMST+CLUS, which performed best for synonyms and competitively for hypernyms, has a runtime that is over 6,000 times faster. In practice, this simpler algorithm may be preferable to use.

One surprising result is the speed of MAXTRANSGRAPH, which theoretically has a number of variables and constraints on the same order as that of MAXTRANSFOREST. In practice, we found that the average number of *active* constraints for MAXTRANSGRAPH – those violated at any point in the course of solving the ILP – was less than one percent of the average number of active constraints in MAXTRANSFOREST.

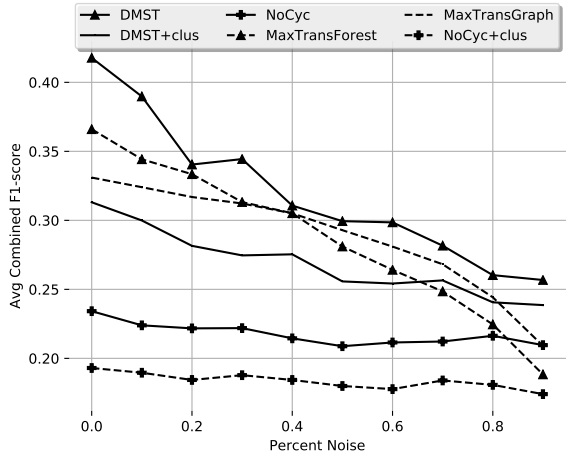


Figure 5: Weighted average combined F1-scores over the test set, where each algorithm is run in an oracle setting with noise percentage (p) settings in the range from 0% to 90%.

5.3 Assessing robustness to noise

Finally, given that hypernym prediction is still an open problem, we are interested in finding out how robust each algorithm is to noise in the input hypernym predictions. To test this, we re-run each taxonomy organization algorithm on the local taxonomies in an oracle setting, where the score of all potential edges that are present as direct or transitive edges in WordNet is set to 1. In each iteration, we set a noise probability p, and randomly perturb edge scores (according to a Gaussian distribution with 0 mean and 0.15 standard deviation) with probability p. We run this experiment with $p \in [0, 90]$. The combined F1-score is plotted against the noise level in Figure 5. We find that the performance of transitive algorithms MAXTRANSGRAPH and MAXTRANSFOREST degrades more quickly than the performance of other algorithms at higher noise levels. DMST performs best in the oracle setting at all levels of noise. The results are shown in Figure 5.

The performance of the top two performing algorithms, MAXTRANSFOREST and DMST+CLUS, in terms of combined F1-score degrades most with the introduction of noise. But even with up to 40% noise, these algorithms still out-perform all others.

6 Conclusion

In this paper we have conducted a direct comparison of six taxonomy organization algorithms that vary in terms of their transitivity and graph structure constraints, and their treatment of synonyms. Evaluating their performance over a dataset of local taxonomies drawn from PPDB, we find that transitive algorithms generally out-perform their non-transitive counterparts. While the best-performing algorithm – an ILP approach that constrains graphs to be transitive and tree-structured – is too slow to use on large inputs, a much simpler maximum spanning tree algorithm that consolidates synonyms into a single taxonomic node has complementary performance, with a small fraction of the runtime. Our results suggest that incorporating synonym detection into tree-constrained taxonomy organization algorithms is a promising area for future research.

Acknowledgments

This material is based in part on research sponsored by DARPA under grant number FA8750-13-2-0017 (the DEFT program) and HR0011-15-C-0115 (the LORELEI program). The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes. The views and conclusions contained in this publication are those of the authors and should not be interpreted as representing official policies or endorsements of DARPA and the U.S. Government. The work has also been supported by the French National Research Agency under project ANR-16-CE33-0013, and by the National Physical Science Consortium.

We are grateful to our anonymous reviewers for their thoughtful and constructive comments.

References

Alfred V. Aho, Michael R Garey, and Jeffrey D. Ullman. 1972. The transitive reduction of a directed graph. *SIAM Journal on Computing* 1(2):131–137.

Mohit Bansal, David Burkett, Gerard De Melo, and Dan Klein. 2014. Structured learning for taxonomy induction with belief propagation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*. Baltimore, Maryland, USA, pages 1041–1051.

Marco Baroni and Alessandro Lenci. 2011. How we BLESSed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*. Edinburgh, Scotland, pages 1–10.

Giulia Benotto. 2015. *Distributional models for semantic relations: A sudy on hyponymy and antonymy*. Ph.D. thesis.

Jonathan Berant, Noga Alon, Ido Dagan, and Jacob Goldberger. 2015. Efficient global learning of entailment graphs. *Computational Linguistics* 41(2):249–291.

Jonathan Berant, Ido Dagan, and Jacob Goldberger. 2012. Learning entailment relations by global graph structure optimization. *Computational Linguistics* 38(1):73–111.

Olivier Bodenreider. 2004. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research* 32(suppl_1):D267–D270.

Georgeta Bordea, Paul Buitelaar, Stefano Faralli, and Roberto Navigli. 2015. Semeval-2015 task 17: Taxonomy extraction evaluation (TExEval). In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval-2015)*. Denver, Colorado, USA, pages 902–910.

Georgeta Bordea, Els Lefever, and Paul Buitelaar. 2016. Semeval-2016 task 13: Taxonomy extraction evaluation (TExEval-2). In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. San Diego, California, USA, pages 1081–1091.

Yoeng-Jin Chu and T.H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica* 14:1396–1400.

Quang Xuan Do and Dan Roth. 2010. Constraints based taxonomic relation classification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Cambridge, Massachusetts, USA, pages 1099–1109.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards B* 71(4):233–240.

Stefano Faralli, Alexander Panchenko, Chris Biemann, and Simone Paolo Ponzetto. 2017. The ContrastMedium algorithm: taxonomy induction from noisy knowledge graphs with just a few links. In *Proceedings of the 15th Conference of the European Association for Computational Linguistics (EACL)*. Valencia, Spain, pages 590–600.

Stefano Faralli, Giovanni Stilo, and Paola Velardi. 2015. Large scale homophily analysis in twitter using a twixonomy. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*. Buenos Aires, Argentina, pages 2334–2340.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. Atlanta, Georgia, USA, pages 758–764.

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th Conference on Computational Linguistics (COLING)-Volume 2*. Nantes, France, pages 539–545.

Zornitsa Kozareva and Eduard Hovy. 2010. A semi-supervised method to learn and construct taxonomies using the web. In *Proceedings of the 2010 conference on Empirical Methods in Natural Language Processing (EMNLP)*. Cambridge, Massachusetts, USA, pages 1110–1118.

Omer Levy, Steffen Remus, Chris Biemann, Ido Dagan, and Israel Ramat-Gan. 2015. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. Denver, Colorado, USA, pages 970–976.

Dekang Lin and Patrick Pantel. 2002. Concept discovery from text. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING)-Volume 1*. Taipei, Taiwan, pages 1–7.

Suresh Manandhar, Ioannis P Klapaftis, Dmitriy Dligach, and Sameer S Pradhan. 2010. Semeval-2010 task 14: Word sense induction & disambiguation. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval-2010)*. Uppsala, Sweden, pages 63–68.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. PATTY: A taxonomy of relational patterns with semantic types. In *Proceedings of the 2012 Joint Conference on Empirical Methods*

in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL). Jeju Island, Korea, pages 1135–1145.

Roberto Navigli, Paola Velardi, and Stefano Faralli. 2011. A graph-based algorithm for inducing lexical taxonomies from scratch. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*. Barcelona, Spain, volume 2:5.1, pages 1872–1877.

Silvia Necsulescu, Sara Mendes, David Jurgens, Núria Bel, and Roberto Navigli. 2015. Reading between the lines: Overcoming data sparsity for accurate classification of lexical relationships. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics (*SEM)*. Denver, Colorado, USA, pages 182–192.

Alexander Panchenko, Stefano Faralli, Eugen Ruppert, Steffen Remus, Hubert Naets, Cédrick Fairon, Simone Paolo Ponzetto, and Chris Biemann. 2016. TAXI at SemEval-2016 Task 13: a taxonomy induction method based on lexico-syntactic patterns, substrings and focused crawling. In *10th International Workshop on Semantic Evaluation (SemEval-2016)*. San Diego, California, USA, pages 1211–1218.

Patrick Pantel and Deepak Ravichandran. 2004. Automatically labeling semantic classes. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*. Boston, Massachusetts, USA, pages 321–328.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevich, and Chris Callison-Burch Ben Van Durme. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*. Beijing, China, pages 425–430.

Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING)*. Dublin, Ireland, pages 1025–1036.

Enrico Santus, Alessandro Lenci, Tin-Shing Chiu, Qin Lu, and Chu-Ren Huang. 2016. Nine features in a random forest to learn taxonomical semantic relations. Portoroz, Slovenia, pages 4557–4564.

Enrico Santus, Frances Yung, Alessandro Lenci, and Chu-Ren Huang. 2015. Evalution 1.0: an evolving semantic dataset for training and evaluation of distributional semantic models. In *Proceedings of the 4th Workshop on Linked Data in Linguistics: Resources and Applications*. Beijing, China, pages 64–69.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*. Berlin, Germany, pages 2389–2398.

Vered Shwartz, Enrico Santus, and Dominik Schlechtweg. 2017. Hypernyms under siege: Linguistically-motivated artillery for hypernymy detection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. Valencia, Spain, pages 65–75.

Rion Snow, Daniel Jurafsky, Andrew Y Ng, et al. 2004. Learning syntactic patterns for automatic hypernym discovery. In *Advances in Neural Information Processing Systems (NIPS)*. Vancouver, Canada, volume 17, pages 1297–1304.

Robert Tarjan. 1972. Depth-first search and linear graph algorithms. *SIAM journal on computing* 1(2):146–160.

Robert Endre Tarjan. 1977. Finding optimum branchings. *Networks* 7(1):25–35.

Peter D. Turney and Saif Mohammad. 2015. Experiments with three approaches to recognizing lexical entailment. *Natural Language Engineering* 21:437–476.

John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Dan Roth. 2015. From paraphrase database to compositional paraphrase model and back. *Transactions of the Association for Computational Linguistics* 3:345–358.

Improving Lexical Choice in Neural Machine Translation

Toan Q. Nguyen and **David Chiang**
Department of Computer Science and Engineeering
University of Notre Dame
{tnguye28,dchiang}@nd.edu

Abstract

We explore two solutions to the problem of mistranslating rare words in neural machine translation. First, we argue that the standard output layer, which computes the inner product of a vector representing the context with all possible output word embeddings, rewards frequent words disproportionately, and we propose to fix the norms of both vectors to a constant value. Second, we integrate a simple lexical module which is jointly trained with the rest of the model. We evaluate our approaches on eight language pairs with data sizes ranging from 100k to 8M words, and achieve improvements of up to +4.3 BLEU, surpassing phrase-based translation in nearly all settings.[1]

1 Introduction

Neural network approaches to machine translation (Sutskever et al., 2014; Bahdanau et al., 2015; Luong et al., 2015a; Gehring et al., 2017) are appealing for their single-model, end-to-end training process, and have demonstrated competitive performance compared to earlier statistical approaches (Koehn et al., 2007; Junczys-Dowmunt et al., 2016). However, there are still many open problems in NMT (Koehn and Knowles, 2017). One particular issue is mistranslation of rare words. For example, consider the Uzbek sentence:

Source: Ammo muammolar hali ko'p, deydi amerikalik olim Entoni Fauchi.

Reference: But still there are many problems, says American scientist Anthony Fauci.

Baseline NMT: But there is still a lot of problems, says James Chan.

At the position where the output should be *Fauci*, the NMT model's top three candidates are *Chan*, *Fauci*, and *Jenner*. All three surnames occur in the training data with reference to immunologists: Fauci is the director of the National Institute of Allergy and Infectious Diseases, Margaret (not James) Chan is the former director of the World Health Organization, and Edward Jenner invented smallpox vaccine. But *Chan* is more frequent in the training data than *Fauci*, and *James* is more frequent than either *Anthony* or *Margaret*.

Because NMT learns word representations in continuous space, it tends to translate words that "seem natural in the context, but do not reflect the content of the source sentence" (Arthur et al., 2016). This coincides with other observations that NMT's translations are often fluent but lack accuracy (Wang et al., 2017b; Wu et al., 2016).

Why does this happen? At each time step, the model's distribution over output words e is

$$p(e) \propto \exp\left(W_e \cdot \tilde{h} + b_e\right)$$

where W_e and b_e are a vector and a scalar depending only on e, and $\tilde{h}$ is a vector depending only on the source sentence and previous output words. We propose two modifications to this layer. First, we argue that the term $W_e \cdot \tilde{h}$, which measures how well e fits into the context $\tilde{h}$, favors common words disproportionately, and show that it helps to fix the norm of both vectors to a constant. Second, we add a new term representing a more direct connection from the source sentence, which allows the model to better memorize translations of rare words.

Below, we describe our models in more detail. Then we evaluate our approaches on eight language pairs, with training data sizes ranging from 100k words to 8M words, and show improvements of up to +4.3 BLEU, surpassing phrase-based translation in nearly all settings. Finally, we provide some analysis to better understand why our modifications work well.

[1]The code for this work can be found at https://github.com/tnq177/improving_lexical_choice_in_nmt

Proceedings of NAACL-HLT 2018, pages 334–343
New Orleans, Louisiana, June 1 - 6, 2018.

	ha-en	tu-en	hu-en
untied embeddings	17.2	11.5	26.5
tied embeddings	17.4	13.8	26.5
don't normalize $\tilde{h}_t$	18.6	14.2	27.1
normalize $\tilde{h}_t$	20.5	16.1	28.8

Table 1: Preliminary experiments show that tying target embeddings with output layer weights performs as well as or better than the baseline, and that normalizing $\tilde{h}$ is better than not normalizing $\tilde{h}$. All numbers are BLEU scores on development sets, scored against tokenized references.

2 Neural Machine Translation

Given a source sequence $f = f_1 f_2 \cdots f_m$, the goal of NMT is to find the target sequence $e = e_1 e_2 \cdots e_n$ that maximizes the objective function:

$$\log p(e \mid f) = \sum_{t=1}^{n} \log p(e_t \mid e_{<t}, f).$$

We use the *global attentional* model with *general scoring function* and *input feeding* by Luong et al. (2015a). We provide only a very brief overview of this model here. It has an encoder, an attention, and a decoder. The encoder converts the words of the source sentence into *word embeddings*, then into a sequence of *hidden states*. The decoder generates the target sentence word by word with the help of the attention. At each time step t, the attention calculates a set of *attention weights* $a_t(s)$. These attention weights are used to form a weighted average of the encoder hidden states to form a *context vector* c_t. From c_t and the hidden state of the decoder are computed the *attentional hidden state* $\tilde{h}_t$. Finally, the predicted probability distribution of the t'th target word is:

$$p(e_t \mid e_{<t}, f) = \text{softmax}(W^o \tilde{h}_t + b^o). \quad (1)$$

The rows of the output layer's weight matrix W^o can be thought of as embeddings of the output vocabulary, and sometimes are in fact tied to the embeddings in the input layer, reducing model size while often achieving similar performance (Inan et al., 2017; Press and Wolf, 2017). We verified this claim on some language pairs and found out that this approach usually performs better than without tying, as seen in Table 1. For this reason, we always tie the target embeddings and W^o in all of our models.

3 Normalization

The output word distribution (1) can be written as:

$$p(e) \propto \exp\left(\|W_e\| \, \|\tilde{h}\| \cos\theta_{W_e,\tilde{h}} + b_e\right),$$

where W_e is the embedding of e, b_e is the e'th component of the bias b^o, and $\theta_{W_e,\tilde{h}}$ is the angle between W_e and $\tilde{h}$. We can intuitively interpret the terms as follows. The term $\|\tilde{h}\|$ has the effect of sharpening or flattening the distribution, reflecting whether the model is more or less certain in a particular context. The cosine similarity $\cos\theta_{W_e,\tilde{h}}$ measures how well e fits into the context. The bias b_e controls how much the word e is generated; it is analogous to the language model in a log-linear translation model (Och and Ney, 2002).

Finally, $\|W_e\|$ also controls how much e is generated. Figure 1 shows that it generally correlates with frequency. But because it is multiplied by $\cos\theta_{W_e,\tilde{h}}$, it has a stronger effect on words whose embeddings have direction similar to $\tilde{h}$, and less effect or even a negative effect on words in other directions. We hypothesize that the result is that the model learns $\|W_e\|$ that are disproportionately large.

For example, returning to the example from Section 1, these terms are:

e	$\|W_e\|$	$\|\tilde{h}\|$	$\cos\theta_{W_e,\tilde{h}}$	b_e	logit
Chan	5.25	19.5	0.144	−1.53	13.2
Fauci	4.69	19.5	0.154	−1.35	12.8
Jenner	5.23	19.5	0.120	−1.59	10.7

Observe that $\cos\theta_{W_e,\tilde{h}}$ and even b_e both favor the correct output word *Fauci*, whereas $\|W_e\|$ favors the more frequent, but incorrect, word *Chan*. The most frequently-mentioned immunologist trumps other immunologists.

To solve this issue, we propose to fix the norm of all target word embeddings to some value r. Followingthe weight normalization approach of Salimans and Kingma (2016), we reparameterize W_e as $r\frac{v_e}{\|v_e\|}$, but keep r fixed.

A similar argument could be made for $\|\tilde{h}_t\|$: because a large $\|\tilde{h}_t\|$ sharpens the distribution, causing frequent words to more strongly dominate rare words, we might want to limit it as well. We compared both approaches on a development set and found that replacing $\tilde{h}_t$ in equation (1) with $r\frac{\tilde{h}_t}{\|\tilde{h}_t\|}$ indeed performs better, as shown in Table 1.

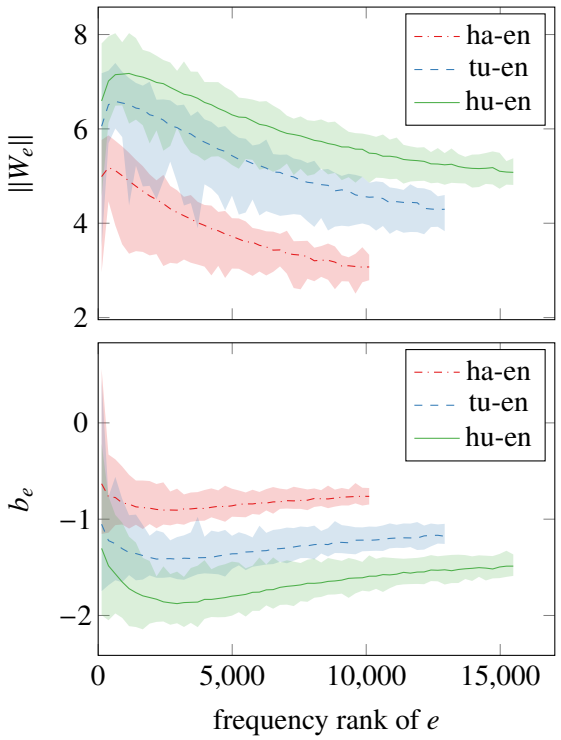


Figure 1: The word embedding norm $\|W_e\|$ generally correlates with the frequency of e, except for the most frequent words. The bias b_e has the opposite behavior. The plots show the median and range of bins of size 256.

4 Lexical Translation

The attentional hidden state $\tilde{h}$ contains information not only about the source word(s) corresponding to the current target word, but also the contexts of those source words and the preceding context of the target word. This could make the model prone to generate a target word that fits the context but doesn't necessarily correspond to the source word(s). Count-based statistical models, by contrast, don't have this problem, because they simply don't model any of this context. Arthur et al. (2016) try to alleviate this issue by integrating a count-based lexicon into an NMT system. However, this lexicon must be trained separately using GIZA++ (Och and Ney, 2003), and its parameters form a large, sparse array, which can be difficult to store in GPU memory.

We propose instead to use a simple feedforward neural network (FFNN) that is trained jointly with the rest of the NMT model to generate a target word based directly on the source word(s). Let f_s ($s = 1, \ldots, m$) be the embeddings of the source words. We use the attention weights to form a weighted average of the embeddings (not the hidden states, as in the main model) to give an average source-word embedding at each decoding time step t:

$$f_t^\ell = \tanh \sum_s a_t(s) f_s.$$

Then we use a one-hidden-layer FFNN with skip connections (He et al., 2016):

$$h_t^\ell = \tanh(W f_t^\ell) + f_t^\ell$$

and combine its output with the decoder output to get the predictive distribution over output words at time step t:

$$p(y_t \mid y_{<t}, x) = \text{softmax}(W^o \tilde{h}_t + b^o + W^\ell h_t^\ell + b^\ell).$$

For the same reasons that were given in Section 3 for normalizing $\tilde{h}_t$ and the rows of W_t^o, we normalize h_t^ℓ and the rows of W^ℓ as well. Note, however, that we do not tie the rows of W^ℓ with the word embeddings; in preliminary experiments, we found this to yield worse results.

	tokens $\times 10^6$	vocab $\times 10^3$	layers num/size
ta-en	0.2/0.1	4.0/3.4	1/512
ur-en	0.2/0.2	4.2/4.2	1/512
ha-en	0.8/0.8	10.6/10.4	2/512
tu-en	0.8/1.1	21.1/13.3	2/512
uz-en	1.5/1.9	29.8/17.4	2/512
hu-en	2.0/2.3	27.3/15.7	2/512
en-vi	2.1/2.6	17.0/7.7	2/512
en-ja (BTEC)	3.6/5.0	17.8/21.8	4/768
en-ja (KFTT)	7.8/8.0	48.2/49.1	4/768

Table 2: Statistics of data and models: effective number of training source/target tokens, source/target vocabulary sizes, number of hidden layers and number of units per layer.

5 Experiments

We conducted experiments testing our normalization approach and our lexical model on eight language pairs using training data sets of various sizes. This section describes the systems tested and our results.

5.1 Data

We evaluated our approaches on various language pairs and datasets:

- Tamil (ta), Urdu (ur), Hausa (ha), Turkish (tu), and Hungarian (hu) to English (en), using data from the LORELEI program.
- English to Vietnamese (vi), using data from the IWSLT 2015 shared task.[2]
- To compare our approach with that of Arthur et al. (2016), we also ran on their English to Japanese (ja) KFTT and BTEC datasets.[3]

We tokenized the LORELEI datasets using the default Moses tokenizer, except for Urdu-English, where the Urdu side happened to be tokenized using Morfessor FlatCat ($w = 0.5$). We used the preprocessed English-Vietnamese and English-Japanese datasets as distributed by Luong et al., and Arthur et al., respectively. Statistics about our data sets are shown in Table 2.

5.2 Systems

We compared our approaches against two baseline NMT systems:

untied, which does not tie the rows of W_o to the target word embeddings, and

tied, which does.

In addition, we compared against two other baseline systems:

Moses: The Moses phrase-based translation system (Koehn et al., 2007), trained on the same data as the NMT systems, with the same maximum sentence length of 50. No additional data was used for training the language model. Unlike the NMT systems, Moses used the full vocabulary from the training data; unknown words were copied to the target sentence.

Arthur: Our reimplementation of the discrete lexicon approach of Arthur et al. (2016). We only tried their `auto` lexicon, using GIZA++ (Och and Ney, 2003), integrated using their `bias` approach. Note that we also tied embedding as we found it also helped in this case.

Against these baselines, we compared our new systems:

fixnorm: The normalization approach described in Section 3.

fixnorm+lex: The same, with the addition of the lexical translation module from Section 4.

[2] https://nlp.stanford.edu/projects/nmt/
[3] http://isw3.naist.jp/~philip-a/emnlp2016/

5.3 Details

Model For all NMT systems, we fed the source sentences to the encoder in reverse order during both training and testing, following Luong et al. (2015a). Information about the number and size of hidden layers is shown in Table 2. The word embedding size is always equal to the hidden layer size.

Following common practice, we only trained on sentences of 50 tokens or less. We limited the vocabulary to word types that appear no less than 5 times in the training data and map the rest to `UNK`. For the English-Japanese and English-Vietnamese datasets, we used the vocabulary sizes reported in their respective papers (Arthur et al., 2016; Luong and Manning, 2015).

For **fixnorm**, we tried $r \in \{3, 5, 7\}$ and selected the best value based on the development set performance, which was $r = 5$ except for English-Japanese (BTEC), where $r = 7$. For **fixnorm+lex**, because $W_s\tilde{h}_t + W^\ell h_t^\ell$ takes on values in $[-2r^2, 2r^2]$, we reduced our candidate r values by roughly a factor of $\sqrt{2}$, to $r \in \{2, 3.5, 5\}$. A radius $r = 3.5$ seemed to work the best for all language pairs.

Training We trained all NMT systems with Adadelta (Zeiler, 2012). All parameters were initialized uniformly from $[-0.01, 0.01]$. When a gradient's norm exceeded 5, we normalized it to 5. We also used dropout on non-recurrent connections only (Zaremba et al., 2014), with probability 0.2. We used minibatches of size 32. We trained for 50 epochs, validating on the development set after every epoch, except on English-Japanese, where we validated twice per epoch. We kept the best checkpoint according to its BLEU on the development set.

Inference We used beam search with a beam size of 12 for translating both the development and test sets. Since NMT often favors short translations (Cho et al., 2014), we followed Wu et al. (2016) in using a modified score $s(e \mid f)$ in place of log-probability:

$$s(e \mid f) = \frac{\log p(e \mid f)}{lp(e)}$$

$$lp(e) = \frac{(5 + |e|)^\alpha}{(5 + 1)^\alpha}$$

We set $\alpha = 0.8$ for all of our experiments.

Finally, we applied a postprocessing step to replace each `UNK` in the target translation with the

source word with the highest attention score (Luong et al., 2015b).

Evaluation For translation into English, we report case-sensitive NIST BLEU against detokenized references. For English-Japanese and English-Vietnamese, we report tokenized, case-sensitive BLEU following Arthur et al. (2016) and Luong and Manning (2015). We measure statistical significance using bootstrap resampling (Koehn, 2004).

6 Results and Analysis

6.1 Overall

Our results are shown in Table 3. First, we observe, as has often been noted in the literature, that NMT tends to perform poorer than PBMT on low resource settings (note that the rows of this table are sorted by training data size).

Our **fixnorm** system alone shows large improvements (shown in parentheses) relative to **tied**. Integrating the lexical module (**fixnorm+lex**) adds in further gains. Our **fixnorm+lex** models surpass Moses on all tasks except Urdu- and Hausa-English, where it is 1.6 and 0.7 BLEU short respectively.

The method of Arthur et al. (2016) does improve over the baseline NMT on most language pairs, but not by as much and as consistently as our models, and often not as well as Moses. Unfortunately, we could not replicate their approach for English-Japanese (KFTT) because the lexical table was too large to fit into the computational graph.

For English-Japanese (BTEC), we note that, due to the small size of the test set, all systems except for Moses are in fact not significantly different from **tied** ($p > 0.01$). On all other tasks, however, our systems significantly improve over **tied** ($p < 0.01$).

6.2 Impact on translation

In Table 4, we show examples of typical translation mistakes made by the baseline NMT systems. In the Uzbek example (top), **untied** and **tied** have confused *34* with *UNK* and *700*, while in the Turkish one (middle), they incorrectly output other proper names, *Afghan* and *Myanmar*, for the proper name *Kenya*. Our systems, on the other hand, translate these words correctly.

The bottom example is the one introduced in Section 1. We can see that our **fixnorm** approach does not completely solve the mistranslation issue, since it translates *Entoni Fauchi* to *UNK UNK* (which is arguably better than *James Chan*). On the other hand, **fixnorm+lex** gets this right. To better understand how the lexical module helps in this case, we look at the top five translations for the word *Fauci* in **fixnorm+lex**:

e	$\cos\theta_{W_e,\tilde{h}}$	$\cos\theta_{W_e^l,h_l}$	$b_e + b_e^l$	logit
Fauci	0.522	0.762	−8.71	7.0
UNK	0.566	−0.009	−1.25	5.6
Anthony	0.263	0.644	−8.70	2.4
Ahmedova	0.555	0.173	−8.66	0.3
Chan	0.546	0.150	−8.73	−0.2

As we can see, while $\cos\theta_{W_e,\tilde{h}}$ might still be confused between similar words, $\cos\theta_{W_e^l,h_l}$ significantly favors *Fauci*.

6.3 Alignment and unknown words

Both our baseline NMT and **fixnorm** models suffer from the problem of shifted alignments noted by Koehn and Knowles (2017). As seen in Figure 2a and 2b, the alignments for those two systems seem to shift by one word to the left (on the source side). For example, *nói* should be aligned to *said* instead of *Telekom*, and so on. Although this is not a problem *per se*, since the decoder can decide to attend to any position in the encoder states as long as the state at that position holds the information the decoder needs, this becomes a real issue when we need to make use of the alignment information, as in unknown word replacement (Luong et al., 2015b). As we can see in Figure 2, because of the alignment shift, both **tied** and **fixnorm** incorrectly replace the two unknown words (in bold) with *But Deutsche* instead of *Deutsche Telekom*. In contrast, under **fixnorm+lex** and the model of Arthur et al. (2016), the alignment is corrected, causing the UNKs to be replaced with the correct source words.

6.4 Impact of r

The single most important hyper-parameter in our models is r. Informally speaking, r controls how much surface area we have on the hypersphere to allocate to word embeddings. To better understand its impact, we look at the training perplexity and dev BLEUs during training with different values of r. Table 6 shows the train perplexity and best tokenized dev BLEU on Turkish-English for **fixnorm** and **fixnorm+lex** with different values of r. As we can see, a smaller r results in

	untied	tied	fixnorm	fixnorm+lex	Moses	Arthur
ta-en	10.3	11.1	14 (+2.9)	15.3 (+4.2)	10.5 (−0.6)	14.1 (+3.0)
ur-en	7.9	10.7	12 (+1.3)	13 (+2.3)	14.6 (+3.9)	12.5 (+1.8)
ha-en	16.0	16.6	20 (+3.4)	21.5 (+4.9)	22.2 (+5.6)	18.7 (+2.1)
tu-en	12.2	12.6	16.4 (+3.8)	19.1 (+6.5)	18.1 (+5.5)	16.3 (+3.7)
uz-en	14.9	15.7	18.2 (+2.5)	19.3 (+3.6)	17.2 (+1.5)	17.1 (+1.4)
hu-en	21.6	23.0	24.0 (+1.0)	25.3 (+2.3)	21.3 (−1.7)	22.7 (-0.3)†
en-vi	25.1	25.3	26.8 (+1.5)	27 (+1.7)	26.7 (+1.4)	26.2 (+0.9)
en-ja (BTEC)	51.2	53.7	52.9 (-0.8)†	51.3 (−2.6)†	46.8 (−6.9)	52.4 (−1.3)†
en-ja (KFTT)	24.1	24.5	26.1 (+1.6)	26.2 (+1.7)	21.7 (−2.8)	—

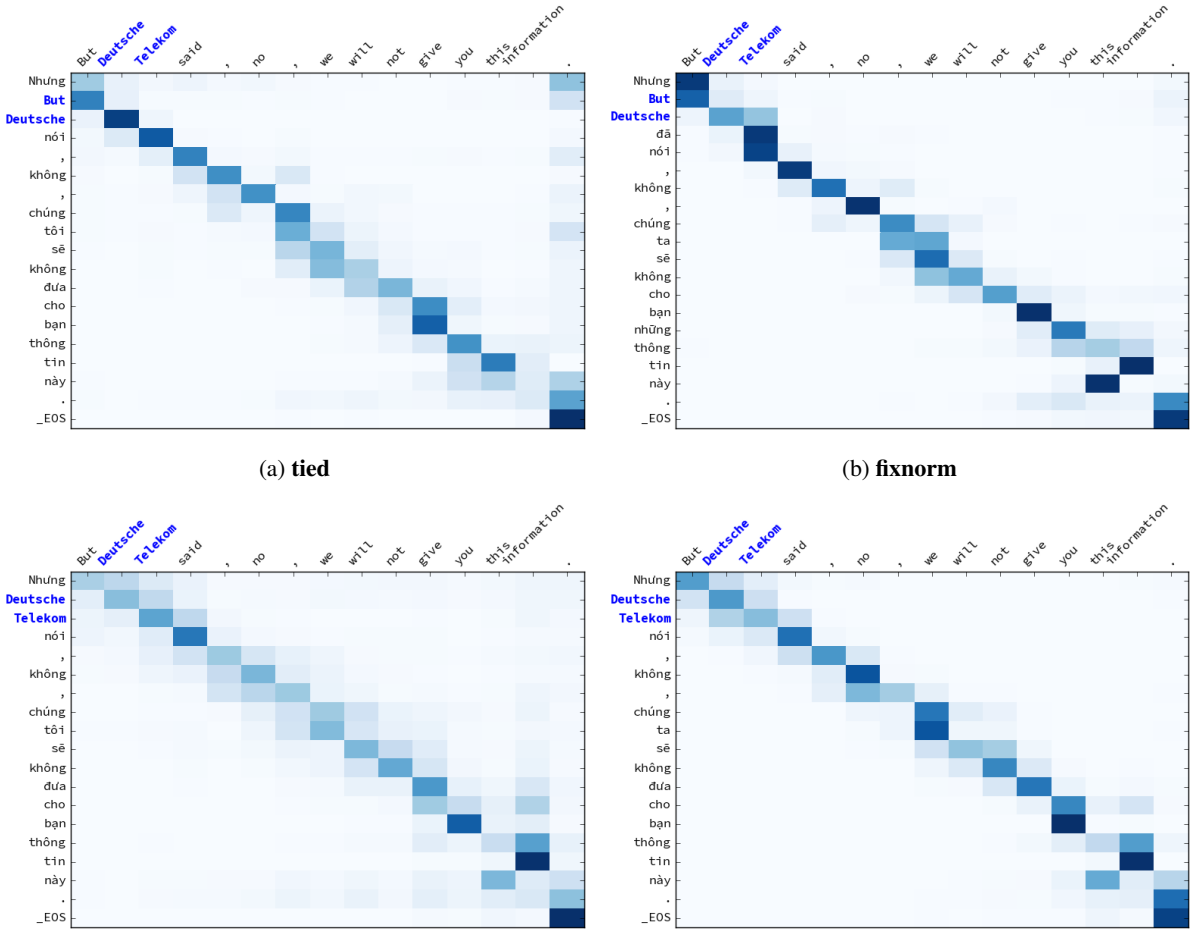
Table 3: Test BLEU of all models. Differences shown in parentheses are relative to **tied**, with a dagger (†) indicating an *insignificant* difference in BLEU ($p > 0.01$). While the method of Arthur et al. (2016) does not always help, **fixnorm** and **fixnorm+lex** consistently achieve significant improvements over **tied** ($p < 0.01$) except for English-Japanese (BTEC). Our models also outperform the method of Arthur et al. on all tasks and outperform Moses on all tasks but Urdu-English and Hausa-English.

input	Dushanba kuni Hindistonda kamida **34** kishi halok bo'lgani xabar qilindi .
reference	At least **34** more deaths were reported Monday in India .
untied	At least **UNK** people have died in India on Monday .
tied	It was reported that at least **700** people died in Monday .
fixnorm	At least **34** people died in India on Monday .
fixnorm+lex	At least **34** people have died in India on Monday .
input	Yarın **Kenya'da** bir yardım konferansı düzenlenecek .
reference	Tomorrow a conference for aid will be conducted in **Kenya** .
untied	Tomorrow there will be an **Afghan** relief conference .
tied	Tomorrow there will be a relief conference in **Myanmar** .
fixnorm	Tomorrow it will be a aid conference in **Kenya** .
fixnorm+lex	Tomorrow there will be a relief conference in **Kenya** .
input	Ammo muammolar hali ko'p , deydi amerikalik olim **Entoni Fauchi** .
reference	But still there are many problems , says American scientist **Anthony Fauci** .
untied	But there is still a lot of problems , says **James Chan** .
tied	However , there is still a lot of problems , says American scientists .
fixnorm	But there is still a lot of problems , says American scientist **UNK UNK** .
fixnorm+lex	But there are still problems , says American scientist **Anthony Fauci** .

Table 4: Example translations, in which **untied** and **tied** generate incorrect, but often semantically related, words, but **fixnorm** and/or **fixnorm+lex** generate the correct ones.

hu-en	244	244 (0.599) document (0.005) By (0.003) by (0.002) offices (0.001)
	befektetéseinek	investments (0.151) investment (0.017) Investments (0.015) all (0.012) investing (0.003)
	kutatás-fejlesztésre	research (0.227) Research (0.040) Development (0.014) researchers (0.008) development (0.007)
tu-en	ifade	expression (0.109) expressed (0.061) express (0.056) speech (0.024) expresses (0.020)
	cumhurbaşkanı	President (0.573) president (0.030) Republic (0.027) Vice (0.010) Abdullah (0.008)
	Göstericiler	protesters (0.115) demonstrators (0.050) Protesters (0.033) UNK (0.004) police (0.003)
ha-en	😒	😒 (0.469) cholera (0.003) EOS (0.001) UNK (0.001) It (0.001)
	Wayoyin	phones (0.414) wires (0.097) mobile (0.088) cellular (0.064) cell (0.061)
	manzonsa	Prophet (0.080) His (0.041) Messenger (0.015) prophet (0.010) his (0.009)

Table 5: Top five translations for some entries of the lexical tables extracted from **fixnorm+lex**. Probabilities are shown in parentheses.



(a) **tied** (b) **fixnorm**

(c) **fixnorm+lex** (d) Arthur et al. (2016)

Figure 2: While the **tied** and **fixnorm** systems shift attention to the left one word (on the source side), our **fixnorm+lex** model and that of Arthur et al. (2016) put it back to the correct position, improving unknown-word replacement for the words *Deutsche Telekom*. Columns are source (English) words and rows are target (Vietnamese) words. Bolded words are unknown.

system	r	train ppl	dev BLEU
fixnorm	3	3.9	13.6
	5	2.5	16.1
	7	2.3	14.4
fixnorm+lex	2	4.2	12.3
	3.5	2.0	17.5
	5	1.4	16.0

Table 6: When r is too small, high train perplexity and low dev BLEU indicate underfitting; when r is too large, low train perplexity and low dev BLEU indicate overfitting.

worse training perplexity, indicating underfitting, whereas if r is too large, the model achieves better training perplexity but decrased dev BLEU, indicating overfitting.

6.5 Lexicon

One byproduct of **lex** is the lexicon, which we can extract and examine simply by feeding each source word embedding to the FFNN module and calculating $p^\ell(y) = \text{softmax}(W^\ell h^\ell + b_\ell)$. In Table 5, we show the top translations for some entries in the lexicons extracted from **fixnorm+lex** for Hungarian, Turkish, and Hausa-English. As expected, the lexical distribution is sparse, with a few top translations accounting for the most probability mass.

6.6 Byte Pair Encoding

Byte-Pair-Encoding (BPE) (Sennrich et al., 2016) is commonly used in NMT to break words into word-pieces, improving the translation of rare words. For this reason, we reran our experiments using BPE on the LORELEI and English-Vietnamese datasets. Additionally, to see if our proposed methods work in high-resource scenarios, we run on the WMT 2014 English-German (en-de) dataset,[4] using *newstest2013* as the development set and reporting tokenized, case-sensitive BLEU on *newstest2014* and *newstest2015*.

We validate across different numbers of BPE operations; specifically, we try {1k, 2k, 3k} merge operations for ta-en and ur-en due to their small sizes, {10k, 12k, 15k} for the other LORELEI datasets and en-vi, and 32k for en-de. Using BPE results in much smaller vocabulary sizes, so we do not apply a vocabulary cut-off. Instead, we train on an additional copy of the training data in which all types that appear once are replaced with `UNK`, and halve the number of epochs accordingly. Our models, training, and evaluation processes are largely the same, except that for en-de, we use a 4-layer decoder and 4-layer bidirectional encoder (2 layers for each direction).

Table 7 shows that our proposed methods also significantly improve the translation when used with BPE, for both high and low resource language pairs. With BPE, we are only behind Moses on Urdu-English.

[4] https://nlp.stanford.edu/projects/nmt/

7 Related Work

The closest work to our **lex** model is that of Arthur et al. (2016), which we have discussed already in Section 4. Recent work by Liu et al. (2016) has very similar motivation to that of our **fixnorm** model. They reformulate the output layer in terms of directions and magnitudes, as we do here. Whereas we have focused on the magnitudes, they focus on the directions, modifying the loss function to try to learn a classifier that separates the classes' directions with something like a margin. Wang et al. (2017a) also make the same observation that we do for the **fixnorm** model, but for the task of face verification.

Handling rare words is an important problem for NMT that has been approached in various ways. Some have focused on reducing the number of `UNK`s by enabling NMT to learn from a larger vocabulary (Jean et al., 2015; Mi et al., 2016); others have focused on replacing `UNK`s by copying source words (Gulcehre et al., 2016; Gu et al., 2016; Luong et al., 2015b). However, these methods only help with unknown words, not rare words. An approach that addresses both unknown and rare words is to use subword-level information (Sennrich et al., 2016; Chung et al., 2016; Luong and Manning, 2016). Our approach is different in that we try to identify and address the root of the rare word problem. We expect that our models would benefit from more advanced `UNK`-replacement or subword-level techniques as well.

Recently, Liu and Kirchhoff (2018) have shown that their baseline NMT system with BPE already outperforms Moses for low-resource translation. However, in their work, they use the Transformer network (Vaswani et al., 2017), which is quite different from our baseline model. It would be interesting to see if our methods benefit the Trans-

	tied	fixnorm	fixnorm+lex	
ta-en	13	15 (+2.0)	15.9	(+2.9)
ur-en	10.5	12.3 (+1.8)	13.7	(+3.2)
ha-en	18	21.7 (+3.7)	22.3	(+4.3)
tu-en	19.3	21 (+1.7)	22.2	(+2.9)
uz-en	18.9	19.8 (+0.9)	21	(+2.1)
hu-en	25.8	27.2 (+1.4)	27.9	(+2.1)
en-vi	26.3	27.3 (+1.0)	27.5	(+1.2)
en-de (newstest2014)	19.7	22.2 (+2.5)	20.4	(+0.7)
en-de (newstest2015)	22.5	25 (+2.5)	23.2	(+0.7)

Table 7: Test BLEU for all BPE-based systems. Our models significantly improve over the baseline ($p < 0.01$) for both high and low resource when using BPE.

former network and other models as well.

8 Conclusion

In this paper, we have presented two simple yet effective changes to the output layer of a NMT model. Both of these changes improve translation quality substantially on low-resource language pairs. In many of the language pairs we tested, the baseline NMT system performs poorly relative to phrase-based translation, but our system surpasses it (when both are trained on the same data). We conclude that NMT, equipped with the methods demonstrated here, is a more viable choice for low-resource translation than before, and are optimistic that NMT's repertoire will continue to grow.

Acknowledgements

This research was supported in part by University of Southern California subcontract 67108176 under DARPA contract HR0011-15-C-0115. Nguyen was supported in part by a fellowship from the Vietnam Education Foundation. We would like to express our great appreciation to Sharon Hu for letting us use her group's GPU cluster (supported by NSF award 1629914), and to NVIDIA corporation for the donation of a Titan X GPU. We also thank Tomer Levinboim for insightful discussions.

References

Philip Arthur, Graham Neubig, and Satoshi Nakamura. 2016. Incorporating discrete translation lexicons into neural machine translation. In *Proc. EMNLP*.

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

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. In *Proc. ACL*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. arXiv:1705.03122.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proc. ACL*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proc. ACL*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proc. CVPR*.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. In *Proc. ICLR*.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proc. ACL-IJCNLP*.

Marcin Junczys-Dowmunt, Tomasz Dwojak, and Hieu Hoang. 2016. Is neural machine translation ready for deployment? A case study on 30 translation directions. In *Proc. IWSLT*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. EMNLP*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. ACL*.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proc. Workshop on Neural Machine Translation*.

Angli Liu and Katrin Kirchhoff. 2018. Context models for oov word translation in low-resource languages. In *Proceedings of AMTA 2018, vol. 1: MT Research Track*. AMTA.

Weiyang Liu, Yandong Wen, Zhiding Yu, and Meng Yang. 2016. Large-margin softmax loss for convolutional neural networks. In *Proc. ICML*.

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford neural machine translation systems for spoken language domain. In *Proc. IWSLT*.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proc. ACL*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015a. Effective approaches to attention-based neural machine translation. In *Proc. EMNLP*.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015b. Addressing the rare word problem in neural machine translation. In *Proc. ACL-IJCNLP*.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016. Vocabulary manipulation for neural machine translation. In *Proc. ACL*.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proc. ACL*.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics* 29(1).

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proc. EACL*.

T. Salimans and D. P. Kingma. 2016. Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. *ArXiv e-prints* .

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proc. ACL*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS 27*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. pages 6000–6010.

Feng Wang, Xiang Xiang, Jian Cheng, and Alan L. Yuille. 2017a. Normface: L2 hypersphere embedding for face verification. In *Proceedings of the 25th ACM international conference on Multimedia*. ACM. https://doi.org/https://doi.org/10.1145/3123266.3123359.

Xing Wang, Zhengdong Lu, Zhaopeng Tu, Hang Li, Deyi Xiong, and Min Zhang. 2017b. Neural machine translation advised by statistical machine translation. In *Proc. AAAI*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv:1609.08144.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. arXiv:1409.2329.

Matthew D. Zeiler. 2012. ADADELTA: An adaptive learning rate method. arXiv:1212.5701v1.

Universal Neural Machine Translation for Extremely Low Resource Languages

Jiatao Gu[†*] **Hany Hassan**[‡] **Jacob Devlin**[□*] **Victor O.K. Li**[†]
[†]The University of Hong Kong [‡]Microsoft Research
{jiataogu, vli}@eee.hku.hk hanyh@microsoft.com
[□]Google Research
jacobdevlin@google.com

Abstract

In this paper, we propose a new universal machine translation approach focusing on languages with a limited amount of parallel data. Our proposed approach utilizes a transfer-learning approach to share lexical and sentence level representations across multiple source languages into one target language. The lexical part is shared through a Universal Lexical Representation to support multi-lingual word-level sharing. The sentence-level sharing is represented by a model of experts from all source languages that share the source encoders with all other languages. This enables the low-resource language to utilize the lexical and sentence representations of the higher resource languages. Our approach is able to achieve 23 BLEU on Romanian-English WMT2016 using a tiny parallel corpus of 6k sentences, compared to the 18 BLEU of strong baseline system which uses multi-lingual training and back-translation. Furthermore, we show that the proposed approach can achieve almost 20 BLEU on the same dataset through fine-tuning a pre-trained multi-lingual system in a zero-shot setting.

1 Introduction

Neural Machine Translation (NMT) (Bahdanau et al., 2015) has achieved remarkable translation quality in various on-line large-scale systems (Wu et al., 2016; Devlin, 2017) as well as achieving state-of-the-art results on Chinese-English translation (Hassan et al., 2018). With such large systems, NMT showed that it can scale up to immense amounts of parallel data in the order of tens of millions of sentences. However, such data is not widely available for all language pairs and domains. In this paper, we propose a novel universal multi-lingual NMT approach focusing mainly on low resource languages to overcome the limitations of NMT and leverage the capabilities of multi-lingual NMT in such scenarios.

Our approach utilizes multi-lingual neural translation system to share lexical and sentence level representations across multiple source languages into one target language. In this setup, some of the source languages may be of extremely limited or even zero data. The lexical sharing is represented by a universal word-level representation where various words from all source languages share the same underlaying representation. The sharing module utilizes monolingual embeddings along with seed parallel data from all languages to build the universal representation. The sentence-level sharing is represented by a model of language experts which enables low-resource languages to utilize the sentence representation of the higher resource languages. This allows the system to translate from any language even with tiny amount of parallel resources.

We evaluate the proposed approach on 3 different languages with tiny or even zero parallel data. We show that for the simulated “zero-resource" settings, our model can consistently outperform a strong multi-lingual NMT baseline with a tiny amount of parallel sentence pairs.

2 Motivation

Neural Machine Translation (NMT) (Bahdanau et al., 2015; Sutskever et al., 2014) is based on Sequence-to-Sequence encoder-decoder model along with an attention mechanism to enable better handling of longer sentences (Bahdanau et al., 2015). Attentional sequence-to-sequence models are modeling the log conditional probability of the

[*]This work was done while the authors at Microsoft.

Proceedings of NAACL-HLT 2018, pages 344–354
New Orleans, Louisiana, June 1 - 6, 2018.

Figure 1: BLEU scores reported on the test set for Ro-En. The amount of training data effects the translation performance dramatically using a single NMT model.

translation Y given an input sequence X. In general, the NMT system θ consists of two components: an encoder θ_e which transforms the input sequence into an array of continuous representations, and a decoder θ_d that dynamically reads the encoder's output with an attention mechanism and predicts the distribution of each target word. Generally, θ is trained to maximize the likelihood on a training set consisting of N parallel sentences:

$$\begin{aligned}\mathcal{L}(\theta) &= \frac{1}{N}\sum_{n=1}^{N}\log p\left(Y^{(n)}|X^{(n)};\theta\right)\\ &= \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T}\log p\left(y_t^{(n)}|y_{1:t-1}^{(n)}, f_t^{\text{att}}(h_{1:T_s}^{(n)})\right)\end{aligned} \quad (1)$$

where at each step, f_t^{att} builds the attention mechanism over the encoder's output $h_{1:T_s}$. More precisely, let the vocabulary size of source words as V

$$h_{1:T_s} = f^{\text{ext}}\left[e_{x_1}, ..., e_{x_{T_s}}\right], \quad e_x = E^I(x) \quad (2)$$

where $E^I \in \mathbb{R}^{V\times d}$ is a look-up table of source embeddings, assigning each individual word a unique embedding vector; f^{ext} is a sentence-level feature extractor and is usually implemented by a multi-layer bidirectional RNN (Bahdanau et al., 2015; Wu et al., 2016), recent efforts also achieved the state-of-the-art using non-recurrence f^{ext}, e.g. ConvS2S (Gehring et al., 2017) and Transformer (Vaswani et al., 2017).

Extremely Low-Resource NMT Both θ_e and θ_d should be trained to converge using parallel training examples. However, the performance is highly correlated to the amount of training data. As shown in Figure. 1, the system cannot achieve reasonable translation quality when the number of the parallel examples is extremely small ($N \approx 13k$ sentences, or not available at all $N = 0$).

Multi-lingual NMT Lee et al. (2017) and Johnson et al. (2017) have shown that NMT is quite efficient for multilingual machine translation. Assuming the translation from K source languages into one target language, a system is trained with maximum likelihood on the mixed parallel pairs $\{X^{(n,k)}, Y^{(n,k)}\}_{k=1...K}^{n=1...N_k}$, that is

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{k=1}^{K}\sum_{n=1}^{N_k}\log p\left(Y^{(n,k)}|X^{(n,k)};\theta\right) \quad (3)$$

where $N = \sum_{k=1}^{K} N_k$. As the input layer, the system assumes a multilingual vocabulary which is usually the union of all source language vocabularies with a total size as $V = \sum_{k=1}^{K} V_k$. In practice, it is essential to shuffle the multilingual sentence pairs into mini-batches so that different languages can be trained equally. Multi-lingual NMT is quite appealing for low-resource languages; several papers highlighted the characteristic that make it a good fit for that such as Lee et al. (2017), Johnson et al. (2017), Zoph et al. (2016) and Firat et al. (2016). Multi-lingual NMT utilizes the training examples of multiple languages to regularize the models avoiding over-fitting to the limited data of the smaller languages. Moreover, the model transfers the translation knowledge from high-resource languages to low-resource ones. Finlay, the decoder part of the model is sufficiently trained since it shares multilingual examples from all languages.

2.1 Challenges

Despite the success of training multi-lingual NMT systems; there are a couple of challenges to leverage them for zero-resource languages:

Lexical-level Sharing Conventionally, a multi-lingual NMT model has a vocabulary that represents the union of the vocabularies of all source languages. Therefore, the multi-lingual words do not practically share the same embedding space since each word has its own representation. This does not pose a problem for languages with sufficiently large amount of data, yet it is a major limitation for extremely low resource languages since most of the vocabulary items will not have enough, if any, training examples to get a reliably trained models.

A possible solution is to share the surface form of all source languages through sharing sub-units

such as subwords (Sennrich et al., 2016b) or characters (Kim et al., 2016; Luong and Manning, 2016; Lee et al., 2017). However, for an arbitrary low-resource language we cannot assume significant overlap in the lexical surface forms compared to the high-resource languages. The low-resource language may not even share the same character set as any high-resource language. It is crucial to create a shared semantic representation across all languages that does not rely on surface form overlap.

Sentence-level Sharing It is also crucial for low-resource languages to share source sentence representation with other similar languages. For example, if a language shares syntactic order with another language it should be feasible for the low-resource language to share such representation with another high recourse language. It is also important to utilize monolingual data to learn such representation since the low or zero resource language may have monolingual resources only.

3 Universal Neural Machine Translation

We propose a Universal NMT system that is focused on the scenario where minimal parallel sentences are available. As shown in Fig. 2, we introduce two components to extend the conventional multi-lingual NMT system (Johnson et al., 2017): Universal Lexical Representation (ULR) and Mixture of Language Experts (MoLE) to enable both word-level and sentence-level sharing, respectively.

3.1 Universal Lexical Representation (ULR)

As we highlighted above, it is not straightforward to have a universal representation for all languages. One potential approach is to use a shared source vocabulary, but this is not adequate since it assumes significant surface-form overlap in order being able to generalize between high-resource and low-resource languages. Alternatively, we could train monolingual embeddings in a shared space and use these as the input to our MT system. However, since these embeddings are trained on a monolingual objective, they will not be optimal for an NMT objective. If we simply allow them to change during NMT training, then this will not generalize to the low-resource language where many of the words are unseen in the parallel data. Therefore, our goal is to create a shared embedding space which (a) is trained towards NMT rather than a monolingual objective, (b) is not based on lexical surface forms, and (c) will generalize from the high-resource languages to the low-resource language.

We propose a novel representation for multilingual embedding where each word from any language is represented as a probabilistic mixture of universal-space word embeddings. In this way, semantically similar words from different languages will naturally have similar representations. Our method achieves this utilizing a discrete (but probabilistic) "universal token space", and then learning the embedding matrix for these universal tokens directly in our NMT training.

Lexicon Mapping to the Universal Token Space We first define a discrete universal token set of size M into which all source languages will be projected. In principle, this could correspond to any human or symbolic language, but all experiments here use English as the basis for the universal token space. As shown in Figure 2, we have multiple embedding representations. E^Q is language-specific embedding trained on monolingual data and E^K is universal tokens embedding. The matrices E^K and E^Q are created beforehand and are not trainable during NMT training. E^U is the embedding matrix for these universal tokens which is learned during our NMT training. It is worth noting that shaded parts in Figure2 are trainable during NMT training process.

Therefore, each source word e_x is represented as a mixture of universal tokens M of E^U.

$$e_x = \sum_{i=1}^{M} E^U(u_i) \cdot q(u_i|x) \tag{4}$$

where E^U is an NMT embedding matrix, which is learned during NMT training.

The mapping q projects the multilingual words into the universal space based on their semantic similarity. That is, $q(u|x)$ is a distribution based on the distance $D_s(u, x)$ between u and x as:

$$q(u_i|x) = \frac{e^{D(u_i,x)/\tau}}{\sum_{u_j} e^{D(u_j,x)/\tau}} \tag{5}$$

where τ is a temperature and $D(u_i, x)$ is a scalar score which represents the similarity between source word x and universal token u_i:

$$D(u, x) = E^K(u) \cdot A \cdot E^Q(x)^T \tag{6}$$

where $E^K(u)$ is the "key" embedding of word u, $E^Q(x)$ is the "query" embedding of source word x.

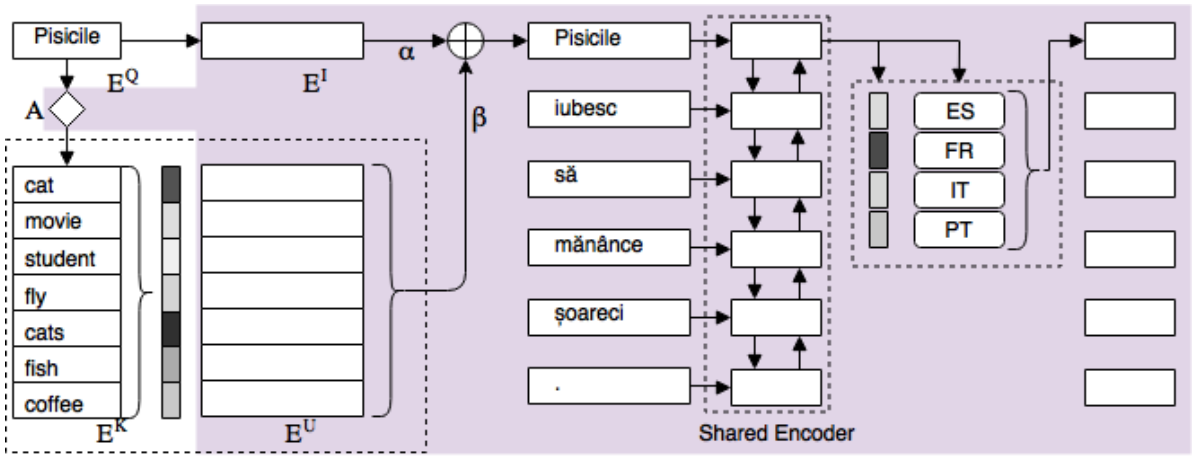


Figure 2: An illustration of the proposed architecture of the ULR and MoLE. Shaded parts are trained within NMT model while unshaded parts are not changed during training.

The transformation matrix A, which is initialized to the identity matrix, is learned during NMT training and shared across all languages.

This is a key-value representation, where the queries are the monolingual language-specific embedding, the keys are the universal tokens embeddings and the values are a probabilistic distribution over the universal NMT embeddings. This can represent unlimited multi-lingual vocabulary that has never been observed in the parallel training data. It is worth noting that the trainable transformation matrix A is added to the query matching mechanism with the main purpose to tune the similarity scores towards the translation task. A is shared across all languages and optimized discriminatively during NMT training such that the system can fine-tune the similarity score $q()$ to be optimal for NMT.

Shared Monolingual Embeddings In general, we create one E^Q matrix per source language, as well as a single E^K matrix in our universal token language. For Equation 6 to make sense and generalize across language pairs, all of these embedding matrices must live in a similar semantic space. To do this, we first train off-the-shelf monolingual word embeddings in each language, and then learn one projection matrix per source language which maps the original monolingual embeddings into E^K space. Typically, we need a list of *source word - universal token* pairs (seeds S_k) to train the projection matrix for language k. Since vectors are normalized, learning the optimal projection is equivalent to finding an orthogonal transformation O_k that makes the projected word vectors as close as to its corresponded universal tokens:

$$\max_{O_k} \sum_{(\tilde{x},\tilde{y})\in S_k} \left(E^{Q_k}(\tilde{x}) \cdot O_k\right) \cdot E^K(\tilde{y})^T \quad \text{s.t. } O_k^T O_k = I, \quad k = 1, ..., K \tag{7}$$

which can be solved by SVD decomposition based on the seeds (Smith et al., 2017). In this paper, we chose to use a short list of seeds from automatic word-alignment of parallel sentences to learn the projection. However, recent efforts (Artetxe et al., 2017; Conneau et al., 2018) also showed that it is possible to learn the transformation without any seeds, which makes it feasible for our proposed method to be utilized in purely zero parallel resource cases.

It is worth noting that O_k is a language-specific matrix which maps the monolingual embeddings of each source language into a similar semantic space as the universal token language.

Interpolated Embeddings Certain lexical categories (e.g. function words) are poorly captured by Equation 4. Luckily, function words often have very high frequency, and can be estimated robustly from even a tiny amount of data. This motivates an interpolated e_x where embeddings for very frequent words are optimized directly and not through the universal tokens:

$$\alpha(x)E^I(x) + \beta(x)\sum_{i=1}^{M} E^U(u_i) \cdot q(u_i|x) \tag{8}$$

Where $E^I(x)$ is a language-specific embedding of word x which is optimized during NMT training. In general, we set $\alpha(x)$ to 1.0 for the top k most frequent words in each language, and 0.0 otherwise,

where k is set to 500 in this work. It is worth noting that we do not use an absolute frequency cutoff because this would cause a mismatch between high-resource and low-resource languages, which we want to avoid. We keep $\beta(x)$ fixed to 1.0.

An Example To give a concrete example, imagine that our target language is English (En), our high-resource auxiliary source languages are Spanish (Es) and French (Fr), and our low-resource source language is Romanian (Ro). En is also used for the universal token set. We assume to have 10M+ parallel Es-En and Fr-En, and a few thousand in Ro-En. We also have millions of monolingual sentences in each language.

We first train word2vec embeddings on monolingual corpora from each of the four languages. We next align the Es-En, Fr-En, and Ro-En parallel corpora and extract a seed dictionary of a few hundred words per language, e.g., `gato` $\rightarrow$ `cat`, `chien` $\rightarrow$ `dog`. We then learn three matrices O_1, O_2, O_3 to project the Es, Fr and Ro embeddings ($E^{Q_1}, E^{Q_2}, E^{Q_3}$), into En ($E^K$) based on these seed dictionaries. At this point, Equation 5 should produce *reasonable* alignments between the source languages and En, e.g., $q(\texttt{horse}|\texttt{magar}) = 0.5$, $q(\texttt{donkey}|\texttt{magar}) = 0.3$, $q(\texttt{cow}|\texttt{magar}) = 0.2$, where `magar` is the Ro word for `donkey`.

3.2 Mixture of Language Experts (MoLE)

As we paved the road for having a universal embedding representation; it is crucial to have a language-sensitive module for the encoder that would help in modeling various language structures which may vary between different languages. We propose a Mixture of Language Experts (MoLE) to model the sentence-level universal encoder. As shown in Fig. 2, an additional module of mixture of experts is used after the last layer of the encoder. Similar to (Shazeer et al., 2017), we have a set of expert networks and a gating network to control the weight of each expert. More precisely, we have a set of expert networks as $f_1(h), ..., f_K(h)$ where for each expert, a two-layer feed-forward network which reads the output hidden states h of the encoder is utilized. The output of the MoLE module h' will be a weighted sum of these experts to replace the encoder's representation:

$$h' = \sum_{k=1}^{K} f_k(h) \cdot \text{softmax}(g(h))_k, \qquad (9)$$

where an one-layer feed-forward network $g(h)$ is used as a gate to compute scores for all the experts.

In our case, we create one expert per auxiliary language. In other words, we train to only use expert f_i when training on a parallel sentence from auxiliary language i. Assume the language $1...K-1$ are the auxiliary languages. That is, we have a multi-task objective as:

$$\mathcal{L}^{\text{gate}} = \sum_{k=1}^{K-1} \sum_{n=1}^{N_k} \log \left[\text{softmax}\left(g(h)\right)_k\right] \qquad (10)$$

We do not update the MoLE module for training on a sentence from the low-resource language. Intuitively, this allows us to represent each token in the low-resource language as a context-dependent mixture of the auxiliary language experts.

4 Experiments

We extensively study the effectiveness of the proposed methods by evaluating on three "almost-zero-resource" language pairs with variant auxiliary languages. The vanilla single-source NMT and the multi-lingual NMT models are used as baselines.

4.1 Settings

Dataset We empirically evaluate the proposed Universal NMT system on 3 languages – Romanian (Ro) / Latvian (Lv) / Korean (Ko) – translating to English (En) in near zero-resource settings. To achieve this, single or multiple auxiliary languages from Czech (Cs), German (De), Greek (El), Spanish (Es), Finnish (Fi), French (Fr), Italian (It), Portuguese (Pt) and Russian (Ru) are jointly trained. The detailed statistics and sources of the available parallel resource can be found in Table 1, where we further down-sample the corpora for the targeted languages to simulate zero-resource.

It also requires additional large amount of monolingual data to obtain the word embeddings for each language, where we use the latest Wikipedia dumps [5] for all the languages. Typically, the monolingual corpora are much larger than the parallel corpora. For validation and testing, the standard validation and testing sets are utilized for each targeted language.

[1] http://www.statmt.org/wmt16/translation-task.html
[2] https://sites.google.com/site/koreanparalleldata/
[3] http://www.statmt.org/europarl/
[4] http://opus.lingfil.uu.se/MultiUN.php (subset)
[5] https://dumps.wikimedia.org/

	Zero-Resource Translation			Auxiliary High-Resource Translation								
source	Ro	Ko	Lv	Cs	De	El	Es	Fi	Fr	It	Pt	Ru
corpora	WMT16[1]	KPD[2]		Europarl v8[3]								UN [4]
size	612k	97k	638k	645k	1.91m	1.23m	1.96m	1.92m	2.00m	1.90m	1.96m	11.7m
subset	0/6k/60k	10k	6k	/								2.00m

Table 1: Statistics of the available parallel resource in our experiments. All the languages are translated to English.

Preprocessing All the data (parallel and monolingual) have been tokenized and segmented into subword symbols using byte-pair encoding (BPE) (Sennrich et al., 2016b). We use sentences of length up to 50 subword symbols for all languages. For each language, a maximum number of $40,000$ BPE operations are learned and applied to restrict the size of the vocabulary. We concatenate the vocabularies of all source languages in the multilingual setting where special a "language marker " have been appended to each word so that there will be no embedding sharing on the surface form. Thus, we avoid sharing the representation of words that have similar surface forms though with different meaning in various languages.

Architecture We implement an attention-based neural machine translation model which consists of a one-layer bidirectional RNN encoder and a two-layer attention-based RNN decoder. All RNNs have 512 LSTM units (Hochreiter and Schmidhuber, 1997). Both the dimensions of the source and target embedding vectors are set to 512. The dimensionality of universal embeddings is also the same. For a fair comparison, the same architecture is also utilized for training both the vanilla and multilingual NMT systems. For multilingual experiments, $1 \sim 5$ auxiliary languages are used. When training with the universal tokens, the temperature τ (in Eq. 6) is fixed to 0.05 for all the experiments.

Learning All the models are trained to maximize the log-likelihood using Adam (Kingma and Ba, 2014) optimizer for 1 million steps on the mixed dataset with a batch size of 128. The dropout rates for both the encoder and the decoder is set to 0.4. We have open-sourced an implementation of the proposed model. [6]

4.2 Back-Translation

We utilize back-translation (BT) (Sennrich et al., 2016a) to encourage the model to use more information of the zero-resource languages. More concretely, we build the synthetic parallel corpus by translating on monolingual data[7] with a trained translation system and use it to train a backward direction translation model. Once trained, the same operation can be used on the forward direction. Generally, BT is difficult to apply for zero resource setting since it requires a reasonably good translation system to generate good quality synthetic parallel data. Such a system may not be feasible with tiny or zero parallel data. However, it is possible to start with a trained multi-NMT model.

4.3 Preliminary Experiments

Training Monolingual Embeddings We train the monolingual embeddings using `fastText`[8] (Bojanowski et al., 2017) over the Wikipedia corpora of all the languages. The vectors are set to 300 dimensions, trained using the default setting of skip-gram . All the vectors are normalized to norm 1.

Pre-projection In this paper, the pre-projection requires initial word alignments (seeds) between words of each source language and the universal tokens. More precisely, for the experiments of Ro/Ko/Lv-En, we use the target language (En) as the universal tokens; `fast_align`[9] is used to automatically collect the aligned words between the source languages and English.

5 Results

We show our main results of multiple source languages to English with different auxiliary languages in Table 2. To have a fair comparison, we use only 6k sentences corpus for both Ro and Lv with all the settings and 10k for Ko. It is obvious that applying both the universal tokens and mixture of experts modules improve the overall translation quality for all the language pairs and the improvements are additive.

To examine the influence of auxiliary languages, we tested four sets of different combinations of auxiliary languages for Ro-En and two sets for Lv-En.

[6] https://github.com/MultiPath/NA-NMT/tree/universal_translation

[7] We used News Crawl provided by WMT16 for Ro-En.

[8] https://github.com/facebookresearch/fastText

[9] https://github.com/clab/fast_align

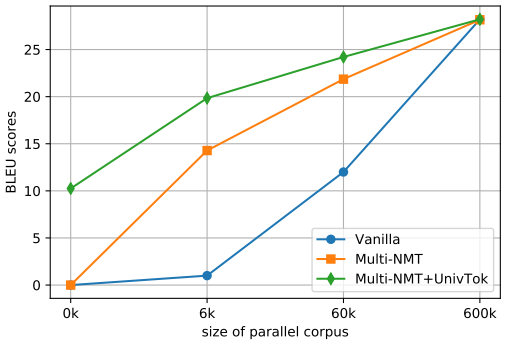


Figure 3: BLEU score vs corpus size

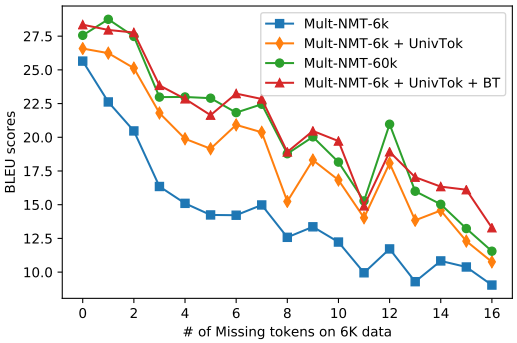


Figure 4: BLEU score vs unknown tokens

Src	Aux	Multi	+ULR	+ MoLE
Ro	Cs De El Fi		18.02	18.37
	Cs De El Fr		19.48	19.52
	De El Fi It		19.11	19.33
	Es Fr It Pt	14.83	20.01	**20.51**
Lv	Es Fr It Pt	7.68	10.86	11.02
	Es Fr It Pt Ru	7.88	12.40	**13.16**
Ko	Es Fr It Pt	2.45	5.49	**6.14**

Table 2: Scores over variant source languages (6k sentences for Ro & Lv, and 10k for Ko). "Multi" means the Multi-lingual NMT baseline.

Models	BLEU
Vanilla	1.21
Multi-NMT	14.94
Closest Uni-Token Only	5.83
Multi-NMT + ULR + (A=I)	18.61
Multi-NMT + ULR	**20.01**
Multi-NMT + BT	17.91
Multi-NMT + ULR + BT	**22.35**
Multi-NMT + ULR + MoLE	20.51
Multi-NMT + ULR + MoLE + BT	**22.92**
Full data (612k) NMT	**28.34**

Table 3: BLEU scores evaluated on test set (6k), compared with ULR and MoLE. "vanilla" is the standard NMT system trained only on Ro-En training set

It shows that Ro performs best when the auxiliary languages are all selected in the same family (Ro, Es, Fr, It and Pt are all from the Romance family of European languages) which makes sense as more knowledge can be shared across the same family. Similarly, for the experiment of Lv-En, improvements are also observed when adding Ru as additional auxiliary language as Lv and Ru share many similarities because of the geo-graphical influence even though they don't share the same alphabet.

We also tested a set of Ko-En experiments to examine the generalization capability of our approach on non-European languages while using languages of Romance family as auxiliary languages. Although the BLEU score is relatively low, the proposed methods can consistently help translating less-related low-resource languages. It is more reasonable to have similar languages as auxiliary languages.

5.1 Ablation Study

We perform thorough experiments to examine effectiveness of the proposed method; we do ablation study on Ro-En where all the models are trained based on the same Ro-En corpus with 6k sentences.

As shown in Table 3, it is obvious that 6k sentences of parallel corpora completely fails to train a vanilla NMT model. Using Multi-NMT with the assistance of 7.8M auxiliary language sentence pairs, Ro-En translation performance gets a substantial improvement which, however, is still limited to be usable. By contrast, the proposed ULR boosts the Multi-NMT significantly with +5.07 BLEU, which is further boosted to +7.98 BLEU when incorporating sentence-level information using both MoLE and BT. Furthermore, it is also shown that ULR works better when a trainable transformation matrix A is used (4th vs 5th row in the table). Note that, although still $5 \sim 6$ BLEU scores lower than the full data ($\times 100$ large) model.

We also measure the translation quality of simply training the vanilla system while replacing each token of the Ro sentence with its closet universal token in the projected embedding space, considering we are using the target languages (En) as

the universal tokens. Although the performance is much worse than the baseline Multi-NMT, it still outperforms the vanilla model which implies the effectiveness of the embedding alignments.

Monolingual Data In Table. 3, we also showed the performance when incorporating the monolingual Ro corpora to help the UniNMT training in both cases with and without ULR. The back-translation improves in both cases, while the ULR still obtains the best score which indicates that the gains achieved are additive.

Corpus Size As shown in Fig. 3, we also evaluated our methods with varied sizes – 0k[10], 6k, 60k and 600k – of the Ro-En corpus. The vanilla NMT and the multi-lingual NMT are used as baselines. It is clear in all cases that the performance gets better when the training corpus is larger. However, the multilingual with ULR works much better with a small amount of training examples. Note that, the usage of ULR universal tokens also enables us to directly work on a "pure zero" resource translation with a shared multilingual NMT model.

Unknown Tokens One explanation on how ULR help the translation for almost zero resource languages is it greatly cancel out the effects of missing tokens that would cause out-of-vocabularies during testing. As in Fig. 4, the translation performance heavily drops when it has more "unknown" which cannot be found in the given 6k training set, especially for the typical multilingual NMT. Instead, these "unknown" tokens will naturally have their embeddings based on ULR projected universal tokens even if we never saw them in the training set. When we apply back-translation over the monolingual data, the performance further improves which can almost catch up with the model trained with 60k data.

5.2 Qualitative Analysis

Examples Figure 5 shows some cherry-picked examples for Ro-En. Example (a) shows how the lexical selection get enriched when introducing ULR (Lex-6K) as well as when adding Back Translation (Lex-6K-BT). Example (b) shows the effect of using romance vs non-romance languages as the supporting languages for Ro. Example (c) shows the importance of having a trainable A as have been discussed; without trainable A the model confuses "india" and "china" as they may have close representation in the mono-lingual embeddings.

Visualization of MoLE Figure 6 shows the activations along with the same source sentence with various auxiliary languages. It is clear that MoLE is effectively switching between the experts when dealing with zero-resource language words. For this particular example of Ro, we can see that the system is utilizing various auxiliary languages based on their relatedness to the source language. We can approximately rank the relatedness based of the influence of each language. For instance, the influence can be approximately ranked as $Es \approx Pt > Fr \approx It > Cs \approx El > De > Fi$, which is interestingly close to the grammatical relatedness of Ro to these languages. On the other hand, Cs has a strong influence although it does not fall in the same language family with Ro, we think this is due to the geo-graphical influence between the two languages since Cs and Ro share similar phrases and expressions. This shows that MoLE learns to utilize resources from similar languages.

5.3 Fine-tuning a Pre-trained Model

All the described experiments above had the low resource languages jointly trained with all the auxiliary high-resource languages, where the training of the large amount of high-resource languages can be seen as a sort of regularization. It is also common to train a model on high-resource languages first, and then fine-tune the model on a small resource language similar to transfer learning approaches (Zoph et al., 2016). However, it is not trivial to effectively fine-tune NMT models on extremely low resource data since the models easily over-fit due to over-parameterization of the neural networks.

In this experiment, we have explored the fine-tuning tasks using our approach. First, we train a Multi-NMT model (with ULR) on {Es, Fr, It, Pt}-En languages only to create a zero-shot setting for Ro-En translation. Then, we start fine-tuning the model with $6k$ parallel corpora of Ro-En, with and without ULR. As shown in Fig. 7, both models improve a lot over the baseline. With the help of ULR, we can achieve a BLEU score of around 10.7 (also shown in Fig. 3) for Ro-En translation with "zero-resource" translation. The BLEU score can further improve to almost 20 BLEU after 3 epochs of training on $6k$ sentences using ULR. This is almost 6 BLEU higher than the best score of the

[10]For 0k experiments, we used the pre-projection learned from 6k data. It is also possible to use unsupervised learned dictionary.

(a)	Source	situatia este putin diferita atunci cand sunt analizate separat raspunsurile barbatilor si ale femeilor .
	Reference	the situation is slightly different when responses are analysed separately for men and women .
	Mul-6k	the situation is less different when it comes to issues of men and women .
	Mul-60k	the situation is at least different when it is weighed up separately by men and women .
	Lex-6k	the situation is somewhat different when we have a separate analysis of women ‘s and women ‘s responses .
	Lex-6k +BT	the situation is slightly different when it is analysed separately from the responses of men and women .

(b)	Source	ce nu stim este in cat timp se va intampla si cat va dura .
	Reference	what we don ' t know is how long all of that will take and how long it will last .
	Lex (Romance)	what we do not know is how long it will be and how long it will take .
	Lex (Non-Rom)	what we know is as long as it will happen and how it will go

(c)	Source	limita de greutate pentru acestea dateaza din anii ' 80 , cand air india a inceput sa foloseasca grafice cu greutatea si inaltimea ideale .
	Reference	he weight limit for them dates from the ' 80s , when air india began using ideal weight and height graphics .
	Lex (A = I)	the weight limit for these dates back from the 1960s , when the chinese air began to use physiars with weight and the right height .
	Lex	the weight limit for these dates dates from the 1980s , when air india began to use the standard of its standard and height .

Figure 5: Three sets of examples on Ro-En translation with variant settings.

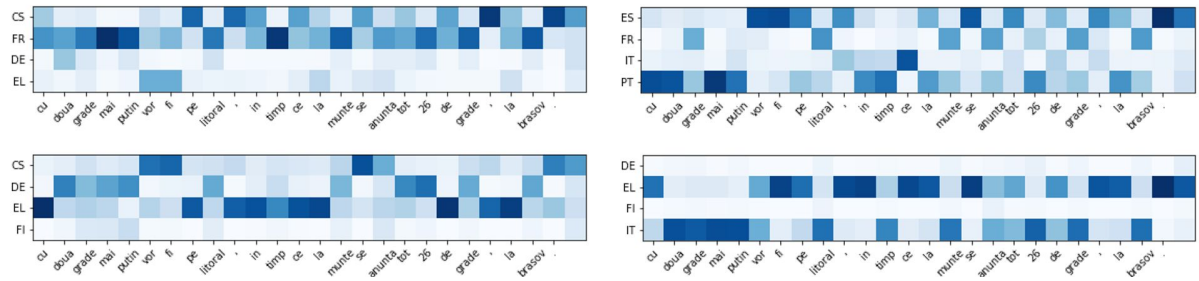


Figure 6: The activation visualization of mixture of language experts module on one randomly selected Ro source sentences trained together with different auxiliary languages. Darker color means higher activation score.

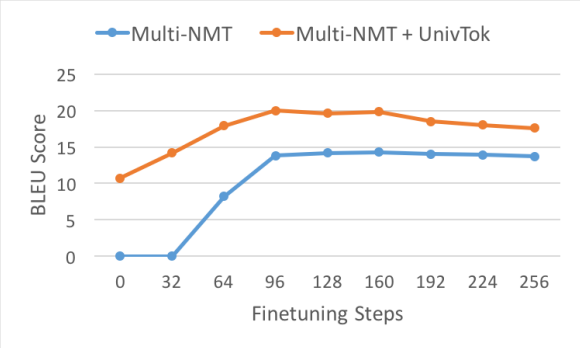


Figure 7: Performance comparison of Fine-tuning on 6K RO sentences.

baseline. It is worth noting that this fine-tuning is a very efficient process since it only takes less than 2 minutes to train for 3 epochs over such tiny amount of data. This is very appealing for practical applications where adapting a per-trained system on-line is a big advantage. As a future work, we will further investigate a better fine-tuning strategy such as meta-learning (Finn et al., 2017) using ULR.

6 Related Work

Multi-lingual NMT has been extensively studied in a number of papers such as Lee et al. (2017), Johnson et al. (2017), Zoph et al. (2016) and Firat et al. (2016). As we discussed, these approaches have significant limitations with zero-resource cases. Johnson et al. (2017) is more closely related to our current approach, our work is extending it to overcome the limitations with very low-resource languages and enable sharing of lexical and sentence representation across multiple languages.

Two recent related works are targeting the same problem of minimally supervised or totally unsupervised NMT. Artetxe et al. (2018) proposed a totally unsupervised approach depending on multi-lingual embedding similar to ours and dual-learning and reconstruction techniques to train the model from mono-lingual data only. Lample et al. (2018) also proposed a quite similar approach while utilizing adversarial learning.

7 Conclusion

In this paper, we propose a new universal machine translation approach that enables sharing resources between high resource languages and extremely low resource languages. Our approach is able to achieve 23 BLEU on Romanian-English WMT2016 using a tiny parallel corpus of 6k sentences, compared to the 18 BLEU of strong multi-lingual baseline system.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *ACL*.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. Unsupervised neural machine translation. In *Proceedings of International Conference on Learning Representations (ICLR)*. Vancouver, Canada.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5:135–146.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *ICLR*.

Jacob Devlin. 2017. Sharp models on dull hardware: Fast and accurate neural machine translation decoding on the cpu. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 2810–2815.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of International Conference on Machine Learning (ICML)*.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, Shujie Liu, Tie-Yan Liu, Renqian Luo, Arul Menezes, Tao Qin, Frank Seide, Xu Tan, Fei Tian, Lijun Wu, Shuangzhi Wu, Yingce Xia, Dongdong Zhang, Zhirui Zhang, and Ming Zhou. 2018. Achieving human parity on automatic chinese to english news translation. *CoRR* abs/1803.05567.

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

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics* 5:339–351.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press, AAAI'16, pages 2741–2749.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. In *Proceedings of International Conference on Learning Representations (ICLR)*. Vancouver, Canada.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. *TACL* 5:365–378.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1054–1063.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Edinburgh neural machine translation systems for wmt 16. In *Proceedings of the First Conference on Machine Translation*. Association for Computational Linguistics, Berlin, Germany, pages 371–376.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 1715–1725.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*.

Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Ł. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *ArXiv e-prints* .

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1568–1575.

Classical Structured Prediction Losses for Sequence to Sequence Learning

Sergey Edunov*, **Myle Ott***,
Michael Auli, David Grangier, Marc'Aurelio Ranzato
Facebook AI Research
Menlo Park, CA and New York, NY

Abstract

There has been much recent work on training neural attention models at the sequence-level using either reinforcement learning-style methods or by optimizing the beam. In this paper, we survey a range of classical objective functions that have been widely used to train linear models for structured prediction and apply them to neural sequence to sequence models. Our experiments show that these losses can perform surprisingly well by slightly outperforming beam search optimization in a like for like setup. We also report new state of the art results on both IWSLT'14 German-English translation as well as Gigaword abstractive summarization. On the large WMT'14 English-French task, sequence-level training achieves 41.5 BLEU which is on par with the state of the art.[1]

1 Introduction

Sequence to sequence models are usually trained with a simple token-level likelihood loss (Sutskever et al., 2014; Bahdanau et al., 2014). However, at test time, these models do not produce a single token but a whole sequence. In order to resolve this inconsistency and to potentially improve generation, recent work has focused on training these models at the sequence-level, for instance using REINFORCE (Ranzato et al., 2015), actor-critic (Bahdanau et al., 2016), or with beam search optimization (Wiseman and Rush, 2016).

Before the recent work on sequence level training for neural networks, there has been a large body of research on training linear models at the sequence level. For example, direct loss optimization has been popularized in machine translation with the Minimum Error Rate Training algorithm (MERT; Och 2003) and expected risk minimization has an extensive history in NLP (Smith and Eisner, 2006; Rosti et al., 2010; Green et al., 2014). This paper revisits several objective functions that have been commonly used for structured prediction tasks in NLP (Gimpel and Smith, 2010) and apply them to a neural sequence to sequence model (Gehring et al., 2017b) (§2). Specifically, we consider likelihood training at the sequence-level, a margin loss as well as expected risk training. We also investigate several combinations of global losses with token-level likelihood. This is, to our knowledge, the most comprehensive comparison of structured losses in the context of neural sequence to sequence models (§3).

We experiment on the IWSLT'14 German-English translation task (Cettolo et al., 2014) as well as the Gigaword abstractive summarization task (Rush et al., 2015). We achieve the best reported accuracy to date on both tasks. We find that the sequence level losses we survey perform similarly to one another and outperform beam search optimization (Wiseman and Rush, 2016) on a comparable setup. On WMT'14 English-French, we also illustrate the effectiveness of risk minimization on a larger translation task. Classical losses for structured prediction are still very competitive and effective for neural models (§5, §6).

2 Sequence to Sequence Learning

The general architecture of our sequence to sequence models follows the encoder-decoder approach with soft attention first introduced in (Bahdanau et al., 2014). As a main difference, in most of our experiments we parameterize the encoder and the decoder as convolutional neural

* Equal contribution.

[1] An implementation of the losses is available as part of fairseq at https://github.com/facebookresearch/fairseq-py/tree/classic_seqlevel

Proceedings of NAACL-HLT 2018, pages 355–364
New Orleans, Louisiana, June 1 - 6, 2018.

networks instead of recurrent networks (Gehring et al., 2017a,b). Our use of convolution is motivated by computational and accuracy considerations. However, the objective functions we present are model agnostic and equally applicable to recurrent and convolutional models. We demonstrate the applicability of our objective functions to recurrent models (LSTM) in our comparison to Wiseman and Rush (2016) in §6.6.

Notation. We denote the source sentence as $\mathbf{x}$, an output sentence of our model as $\mathbf{u}$, and the reference or *target* sentence as $\mathbf{t}$. For some objectives, we choose a pseudo reference $\mathbf{u}^*$ instead, such as a model output with the highest BLEU or ROUGE score among a set of candidate outputs, $\mathcal{U}$, generated by our model.

Concretely, the encoder processes a source sentence $\mathbf{x} = (x_1, \ldots, x_m)$ containing m words and outputs a sequence of states $\mathbf{z} = (z_1 \ldots, z_m)$. The decoder takes $\mathbf{z}$ and generates the output sequence $\mathbf{u} = (u_1, \ldots, u_n)$ left to right, one element at a time. For each output u_i, the decoder computes hidden state h_i based on the previous state h_{i-1}, an embedding g_{i-1} of the previous target language word u_{i-1}, as well as a conditional input c_i derived from the encoder output $\mathbf{z}$. The attention context c_i is computed as a weighted sum of $(z_1, \ldots, z_m)$ at each time step. The weights of this sum are referred to as attention scores and allow the network to focus on the most relevant parts of the input at each generation step. Attention scores are computed by comparing each encoder state z_j to a combination of the previous decoder state h_i and the last prediction u_i; the result is normalized to be a distribution over input elements. At each generation step, the model scores for the V possible next target words u_i by transforming the decoder output h_i via a linear layer with weights W_o and bias b_o: $s_i = W_o h_i + b_o$. This is turned into a distribution via a softmax: $p(u_i|u_1, \ldots, u_{i-1}, \mathbf{x}) = \text{softmax}(s_i)$.

Our encoder and decoder use gated convolutional neural networks which enable fast and accurate generation (Gehring et al., 2017b). Fast generation is essential to efficiently train on the model output as is done in this work as sequence-level losses require generating at training time. Both encoder and decoder networks share a simple block structure that computes intermediate states based on a fixed number of input tokens and we stack several blocks on top of each other. Each block contains a 1-D convolution that takes as input k feature vectors and outputs another vector; subsequent layers operate over the k output elements of the previous layer. The output of the convolution is then fed into a gated linear unit (Dauphin et al., 2017). In the decoder network, we rely on causal convolution which rely only on states from the previous time steps. The parameters θ of our model are all the weight matrices in the encoder and decoder networks. Further details can be found in Gehring et al. (2017b).

3 Objective Functions

We compare several objective functions for training the model architecture described in §2. The corresponding loss functions are either computed over individual tokens (§3.1), over entire sequences (§3.2) or over a combination of tokens and sequences (§3.3). An overview of these loss functions is given in Figure 1.

3.1 Token-Level Objectives

Most prior work on sequence to sequence learning has focused on optimizing token-level loss functions, i.e., functions for which the loss is computed additively over individual tokens.

Token Negative Log Likelihood (`TokNLL`)

Token-level likelihood (`TokNLL`, Equation 1) minimizes the negative log likelihood of individual reference tokens $\mathbf{t} = (t_1, \ldots, t_n)$. It is the most common loss function optimized in related work and serves as a baseline for our comparison.

Token NLL with Label Smoothing (`TokLS`)

Likelihood training forces the model to make extreme zero or one predictions to distinguish between the ground truth and alternatives. This may result in a model that is too confident in its training predictions, which may hurt its generalization performance. Label smoothing addresses this by acting as a regularizer that makes the model less confident in its predictions. Specifically, we smooth the target distribution with a prior distribution f that is independent of the current input $\mathbf{x}$ (Szegedy et al., 2015; Pereyra et al., 2017; Vaswani et al., 2017). We use a uniform prior distribution over all words in the vocabulary, $f = \frac{1}{V}$. One may also use a unigram distribution which has been shown to work better on some tasks (Pereyra et al., 2017). Label smoothing is equivalent to adding the KL divergence between f and the model prediction

$$\mathcal{L}_{\texttt{TokNLL}} = -\sum_{i=1}^{n} \log p(t_i|t_1, \ldots, t_{i-1}, \mathbf{x}) \tag{1}$$

$$\mathcal{L}_{\texttt{TokLS}} = -\sum_{i=1}^{n} \log p(t_i|t_1, \ldots, t_{i-1}, \mathbf{x}) - D_{KL}(f \| p(t_i|t_1, \ldots, t_{i-1}, \mathbf{x})) \tag{2}$$

$$\mathcal{L}_{\texttt{SeqNLL}} = -\log p(\mathbf{u}^*|\mathbf{x}) + \log \sum_{\mathbf{u} \in \mathcal{U}(\mathbf{x})} p(\mathbf{u}|\mathbf{x}) \tag{3}$$

$$\mathcal{L}_{\texttt{Risk}} = \sum_{\mathbf{u} \in \mathcal{U}(\mathbf{x})} \text{cost}(\mathbf{t}, \mathbf{u}) \frac{p(\mathbf{u}|\mathbf{x})}{\sum_{\mathbf{u}' \in \mathcal{U}(\mathbf{x})} p(\mathbf{u}'|\mathbf{x})} \tag{4}$$

$$\mathcal{L}_{\texttt{MaxMargin}} = \max\left[0, \text{cost}(\mathbf{t}, \hat{\mathbf{u}}) - \text{cost}(\mathbf{t}, \mathbf{u}^*) - s(\mathbf{u}^*|\mathbf{x}) + s(\hat{\mathbf{u}}|\mathbf{x})\right] \tag{5}$$

$$\mathcal{L}_{\texttt{MultiMargin}} = \sum_{\mathbf{u} \in \mathcal{U}(\mathbf{x})} \max\left[0, \text{cost}(\mathbf{t}, \mathbf{u}) - \text{cost}(\mathbf{t}, \mathbf{u}^*) - s(\mathbf{u}^*|\mathbf{x}) + s(\mathbf{u}|\mathbf{x})\right] \tag{6}$$

$$\mathcal{L}_{\texttt{SoftmaxMargin}} = -\log p(\mathbf{u}^*|\mathbf{x}) + \log \sum_{\mathbf{u} \in \mathcal{U}(\mathbf{x})} \exp\left[s(\mathbf{u}|\mathbf{x}) + \text{cost}(\mathbf{t}, \mathbf{u})\right] \tag{7}$$

Figure 1: Token and sequence negative log-likelihood (Equations 1 and 3), token-level label smoothing (Equation 2), expected risk (Equation 4), max-margin (Equation 5), multi-margin (Equation 6), softmax-margin (Equation 7). We denote the source as $\mathbf{x}$, the reference target as $\mathbf{t}$, the set of candidate outputs as $\mathcal{U}$ and the best candidate (pseudo reference) as $\mathbf{u}^*$. For max-margin we denote the candidate with the highest model score as $\hat{\mathbf{u}}$.

$p(\mathbf{u}|\mathbf{x})$ to the negative log likelihood (TokLS, Equation 2). In practice, we implement label smoothing by modifying the ground truth distribution for word u to be $q(u) = 1 - \epsilon$ and $q(u') = \frac{\epsilon}{V}$ for $u' \neq u$ instead of $q(u) = 1$ and $q(u') = 0$ where ϵ is a smoothing parameter.

3.2 Sequence-Level Objectives

We also consider a class of objective functions that are computed over entire sequences, i.e., sequence-level objectives. Training with these objectives requires generating and scoring multiple candidate output sequences for each input sequence during training, which is computationally expensive but allows us to directly optimize task-specific metrics such as BLEU or ROUGE.

Unfortunately, these objectives are also typically defined over the entire space of possible output sequences, which is intractable to enumerate or score with our models. Instead, we compute our sequence losses over a subset of the output space, $\mathcal{U}(\mathbf{x})$, generated by the model. We discuss approaches for generating this subset in §4.

Sequence Negative Log Likelihood (SeqNLL)

Similar to TokNLL, we can minimize the negative log likelihood of an entire sequence rather than individual tokens (SeqNLL, Equation 3). The log-likelihood of sequence $\mathbf{u}$ is the sum of individual token log probabilities, normalized by the number of tokens to avoid bias towards shorter sequences:

$$p(\mathbf{u}|\mathbf{x}) = \exp \frac{1}{n} \sum_{i=1}^{n} \log p(u_i|u_1, \ldots, u_{i-1}, \mathbf{x})$$

As target we choose a pseudo reference[2] amongst the candidates which maximizes either BLEU or ROUGE with respect to $\mathbf{t}$, the gold reference:

$$\mathbf{u}^*(\mathbf{x}) = \arg\max_{\mathbf{u} \in \mathcal{U}(\mathbf{x})} \text{BLEU}(\mathbf{t}, \mathbf{u})$$

As is common practice when computing BLEU at the sentence-level, we smooth all initial counts to one (except for unigram counts) so that the geometric mean is not dominated by zero-valued n-gram match counts (Lin and Och, 2004).

Expected Risk Minimization (Risk)

This objective minimizes the expected value of a given cost function over the space of candidate sequences (Risk, Equation 4). In this work we use task-specific cost functions designed to maximize BLEU or ROUGE (Lin, 2004), e.g., $\text{cost}(\mathbf{t}, \mathbf{u}) =$

[2] Another option is to use the gold reference target, $\mathbf{t}$, but in practice this can lead to degenerate solutions in which the model assigns low probabilities to nearly all outputs. This is discussed further in §4.

$1 - \text{BLEU}(\mathbf{t}, \mathbf{u})$, for a given a candidate sequence $\mathbf{u}$ and target $\mathbf{t}$. Different to SeqNLL (§3.2), this loss may increase the score of several candidates that have low cost, instead of focusing on a single sequence which may only be marginally better than any alternatives. Optimizing this loss is a particularly good strategy if the reference is not always reachable, although compared to classical phrase-based models, this is less of an issue with neural sequence to sequence models that predict individual words or even sub-word units.

The Risk objective is similar to the REINFORCE objective used in Ranzato et al. (2015), since both objectives optimize an expected cost or reward (Williams, 1992). However, there are a few important differences: (1) whereas REINFORCE typically approximates the expectation with a single sampled sequence, the Risk objective considers multiple sequences; (2) whereas REINFORCE relies on a *baseline reward*[3] to determine the sign of the gradients for the current sequence, for the Risk objective we instead estimate the expected cost over a set of candidate output sequences (see §4); and (3) while the baseline reward is different for every word in REINFORCE, the expected cost is the same for every word in risk minimization since it is computed on the sequence level based on the actual cost.

Max-Margin

MaxMargin (Equation 5) is a classical margin loss for structured prediction (Taskar et al., 2003; Tsochantaridis et al., 2005) which enforces a margin between the model scores of the highest scoring candidate sequence $\hat{\mathbf{u}}$ and a reference sequence. We replace the human reference $\mathbf{t}$ with a pseudo-reference $\mathbf{u}^*$ since this setting performed slightly better in early experiments; $\mathbf{u}^*$ is the candidate sequence with the highest BLEU score. The size of the margin *varies* between samples and is given by the difference between the cost of $\mathbf{u}^*$ and the cost of $\hat{\mathbf{u}}$. In practice, we scale the margin by a hyper-parameter β determined on the validation set: $\beta(\text{cost}(\mathbf{t}, \hat{\mathbf{u}}) - \text{cost}(\mathbf{t}, \mathbf{u}^*))$. For this loss we use the unnormalized scores computed by the model before the final softmax:

$$s(\mathbf{u}|\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} s(u_i|u_1, \ldots, u_{i-1}, \mathbf{x})$$

[3]Ranzato et al. (2015) estimate the baseline reward for REINFORCE with a separate linear regressor over the model's current hidden state.

Multi-Margin

MaxMargin only updates two elements in the candidate set. We therefore consider MultiMargin (Equation 6) which enforces a margin between *every* candidate sequence $\mathbf{u}$ and a reference sequence (Herbrich et al., 1999), hence the name Multi-Margin. Similar to MaxMargin, we replace the reference $\mathbf{t}$ with the pseudo-reference $\mathbf{u}^*$.

Softmax-Margin

Finally, SoftmaxMargin (Equation 7) is another classic loss that has been proposed by Gimpel and Smith (2010) as another way to optimize task-specific costs. The loss augments the scores inside the exp of SeqNLL (Equation 3) by a cost. The intuition is that we want to penalize high cost outputs proportional to their cost.

3.3 Combined Objectives

We also experiment with two variants of combining sequence-level objectives (§3.2) with token-level objectives (§3.1). First, we consider a weighted combination (Weighted) of both a sequence-level and token-level objective (Wu et al., 2016), e.g., for TokLS and Risk we have:

$$\mathcal{L}_{\texttt{Weighted}} = \alpha\mathcal{L}_{\texttt{TokLS}} + (1-\alpha)\mathcal{L}_{\texttt{Risk}} \quad (8)$$

where α is a scaling constant that is tuned on a held-out validation set.

Second, we consider a constrained combination (Constrained), where for any given input we use either the token-level or sequence-level loss, but not both. The motivation is to maintain good token-level accuracy while optimizing on the sequence-level. In particular, a sample is processed with the sequence loss if the token loss under the current model is at least as good as the token loss of a baseline model $\mathcal{L}^b_{\texttt{TokLS}}$. Otherwise, we update according to the token loss:

$$\mathcal{L}_{\texttt{Constrained}} = \begin{cases} \mathcal{L}_{\texttt{Risk}} & \mathcal{L}_{\texttt{TokLS}} \leq \mathcal{L}^b_{\texttt{TokLS}} \\ \mathcal{L}_{\texttt{TokLS}} & \text{otherwise} \end{cases} \quad (9)$$

In this work we use a fixed baseline model that was trained with a token-level loss to convergence.

4 Candidate Generation Strategies

The sequence-level objectives we consider (§3.2) are defined over the entire space of possible output sequences, which is intractable to enumerate or

score with our models. We therefore use a subset of K candidate sequences $\mathcal{U}(\mathbf{x}) = \{u_1, \ldots, u_K\}$, which we generate with our models.

We consider two search strategies for generating the set of candidate sequences. The first is *beam search*, a greedy breadth-first search that maintains a "beam" of the top-K scoring candidates at each generation step. Beam search is the *de facto* decoding strategy for achieving state-of-the-art results in machine translation. The second strategy is *sampling* (Chatterjee and Cancedda, 2010), which produces K independent output sequences by sampling from the model's conditional distribution. Whereas beam search focuses on high probability candidates, sampling introduces more diverse candidates (see comparison in §6.5).

We also consider both online and offline candidate generation settings in §6.4. In the online setting, we regenerate the candidate set every time we encounter an input sentence $\mathbf{x}$ during training. In the offline setting, candidates are generated before training and are never regenerated. Offline generation is also embarrassingly parallel because all samples use the same model. The disadvantage is that the candidates become stale. Our model may perfectly be able to discriminate between them after only a single update, hindering the ability of the loss to correct eventual search errors.[4]

Finally, while some past work has added the reference target to the candidate set, i.e., $\mathcal{U}'(\mathbf{x}) = \mathcal{U}(\mathbf{x}) \cup \{\mathbf{t}\}$, we find this can destabilize training since the model learns to assign low probabilities nearly everywhere, ruining the candidates generated by the model, while still assigning a slightly higher score to the reference (cf. Shen et al. (2016)). Accordingly we do not add the reference translation to our candidate sets.

5 Experimental Setup

5.1 Translation

We experiment on the IWSLT'14 German to English (Cettolo et al., 2014) task using a similar setup as Ranzato et al. (2015), which allows us to compare to other recent studies that also adopted this setup, e.g., Wiseman and Rush (2016).[5] The training data consists of 160K sentence pairs and the validation set comprises 7K sentences randomly sampled and held-out from the train data. We test on the concatenation of *tst2010, tst2011, tst2012, tst2013* and *dev2010* which is of similar size to the validation set. All data is lowercased and tokenized with a byte-pair encoding (BPE) of 14,000 types (Sennrich et al., 2016) and we evaluate with case-insensitive BLEU.

We also experiment on the much larger WMT'14 English-French task. We remove sentences longer than 175 words as well as pairs with a source/target length ratio exceeding 1.5 resulting in 35.5M sentence-pairs for training. The source and target vocabulary is based on 40K BPE types. Results are reported on both newstest2014 and a validation set held-out from the training data comprising 26,658 sentence pairs.

We modify the *fairseq-py* toolkit to implement the objectives described in §3.[6] Our translation models have four convolutional encoder layers and three convolutional decoder layers with a kernel width of 3 and 256 dimensional hidden states and word embeddings. We optimize these models using Nesterov's accelerated gradient method (Sutskever et al., 2013) with a learning rate of 0.25 and momentum of 0.99. Gradient vectors are renormalized to norm 0.1 (Pascanu et al., 2013).

We train our baseline token-level models for 200 epochs and then anneal the learning by shrinking it by a factor of 10 after each subsequent epoch until the learning rate falls below 10^{-4}. All sequence-level models are initialized with parameters of a token-level model before annealing. We then train sequence-level models for another 10 to 20 epochs depending on the objective. Our batches contain 8K tokens and we normalize gradients by the number of non-padding tokens per mini-batch. We use weight normalization for all layers except for lookup tables (Salimans and Kingma, 2016). Besides dropout on the embeddings and the decoder output, we also apply dropout to the input of the convolutional blocks at a rate of 0.3 (Srivastava et al., 2014). We tuned the various parameters above and report accuracy on the test set by choosing the best configuration based on the validation set.

We length normalize all scores and probabilities in the sequence-level losses by dividing by the number of tokens in the sequence so that scores are comparable between different lengths. Ad-

[4] We can mitigate this issue by regenerating infrequently, i.e., once every b batches but we leave this to future work.

[5] Different to Ranzato et al. (2015) we train on sentences of up to 175 rather than 50 tokens.

[6] https://github.com/facebookresearch/fairseq-py.

ditionally, when generating candidate output sequences during training we limit the output sequence length to be less than 200 tokens for efficiency. We generally use 16 candidate sequences per training example, except for the ablations where we use 5 for faster experimental turnaround.

5.2 Abstractive Summarization

For summarization we use the Gigaword corpus as training data (Graff et al., 2003) and pre-process it identically to Rush et al. (2015) resulting in 3.8M training and 190K validation examples. We evaluate on a Gigaword test set of 2,000 pairs identical to the one used by Rush et al. (2015) and report F1 ROUGE similar to prior work. Our results are in terms of three variants of ROUGE (Lin, 2004), namely, ROUGE-1 (RG-1, unigrams), ROUGE-2 (RG-2, bigrams), and ROUGE-L (RG-L, longest-common substring). Similar to Ayana et al. (2016) we use a source and target vocabulary of 30k words. Our models for this task have 12 layers in the encoder and decoder each with 256 hidden units and kernel width 3. We train on batches of 8,000 tokens with a learning rate of 0.25 for 20 epochs and then anneal as in §5.1.

6 Results

6.1 Comparison of Sequence Level Losses

First, we compare all objectives based on a weighted combination with token-level label smoothing (Equation 8). We also show the likelihood baseline (MLE) of Wiseman and Rush (2016), their beam search optimization method (BSO), the actor critic result of Bahdanau et al. (2016) as well as the best reported result on this dataset to date by Huang et al. (2017). We show a like-for-like comparison to Wiseman and Rush (2016) with a similar baseline model below (§6.6).

Table 1 shows that all sequence-level losses outperform token-level losses. Our baseline token-level results are several points above other figures in the literature and we further improve these results by up to 0.61 BLEU with `Risk` training.

6.2 Combination with Token-Level Loss

Next, we compare various strategies to combine sequence-level and token-level objectives (cf. §3.3). For these experiments we use 5 candidate sequences per training example for faster experimental turnaround. We consider `Risk` as sequence-level loss and label smoothing as token-level loss. Table 2 shows that combined objectives perform better than pure `Risk`. The weighted combination (Equation 8) with $\alpha = 0.3$ performs best, outperforming constrained combination (Equation 9). We also compare to randomly choosing between token-level and sequence-level updates and find it underperforms the more principled constrained strategy. In the remaining experiments we use the weighted strategy.

	test	std
MLE (W & R, 2016) [T]	24.03	
BSO (W & R, 2016) [S]	26.36	
Actor-critic (B, 2016) [S]	28.53	
Huang et al. (2017) [T]	28.96	
Huang et al. (2017) (+LM) [T]	29.16	
`TokNLL` [T]	31.78	0.07
`TokLS` [T]	32.23	0.10
`SeqNLL` [S]	32.68	0.09
`Risk` [S]	32.84	0.08
`MaxMargin` [S]	32.55	0.09
`MultiMargin` [S]	32.59	0.07
`SoftmaxMargin` [S]	32.71	0.07

Table 1: Test accuracy in terms of BLEU on IWSLT'14 German-English translation with various loss functions cf. Figure 1. W & R (2016) refers to Wiseman and Rush (2016), B (2016) to Bahdanau et al. (2016), [S] indicates sequence level-training and [T] token-level training. We report averages and standard deviations over five runs with different random initialization.

	valid	test
`TokLS`	33.11	32.21
`Risk` only	33.55	32.45
`Weighted`	33.91	32.85
`Constrained`	33.77	32.79
Random	33.70	32.61

Table 2: Validation and test BLEU for loss combination strategies. We either use token-level `TokLS` and sequence-level `Risk`individually or combine them as a weighted combination, a constrained combination, a random choice for each sample, cf. §3.3.

6.3 Effect of initialization

So far we initialized sequence-level models with parameters from a token-level model trained with label smoothing. Table 3 shows that initializing weighted `Risk` with token-level label smoothing

	valid	test
TokNLL	32.96	31.74
Risk init with TokNLL	33.27	32.07
Δ	+0.31	+0.33
TokLS	33.11	32.21
Risk init with TokLS	33.91	32.85
Δ	+0.8	+0.64

Table 3: Effect of initializing sequence-level training (Risk) with parameters from token-level likelihood (TokNLL) or label smoothing (TokLS).

	valid	test
Online generation	33.91	32.85
Offline generation	33.52	32.44

Table 4: Generating candidates online or offline.

achieves 0.7-0.8 better BLEU compared to initializing with parameters from token-level likelihood. The improvement of initializing with TokNLL is only 0.3 BLEU with respect to the TokNLL baseline, whereas, the improvement from initializing with TokLS is 0.6-0.8 BLEU. We believe that the regularization provided by label smoothing leads to models with less sharp distributions that are a better starting point for sequence-level training.

6.4 Online vs. Offline Candidate Generation

Next, we consider the question if refreshing the candidate subset at every training step (online) results in better accuracy compared to generating candidates before training and keeping the set static throughout training (offline). Table 4 shows that offline generation gives lower accuracy. However the online setting is much slower, since regenerating the candidate set requires incremental (left to right) inference with our model which is very slow compared to efficient forward/backward over large batches of pre-generated hypothesis. In our setting, offline generation has 26 times higher throughput than the online generation setting, despite the high inference speed of fairseq (Gehring et al., 2017b).

6.5 Beam Search vs. Sampling and Candidate Set Size

So far we generated candidates with beam search, however, we may also sample to obtain a more diverse set of candidates (Shen et al., 2016). Figure 2 compares beam search and sampling for various candidate set sizes on the validation set. Beam search performs better for all candidate set sizes considered. In other experiments, we rely on a candidate set size of 16 which strikes a good balance between efficiency and accuracy.

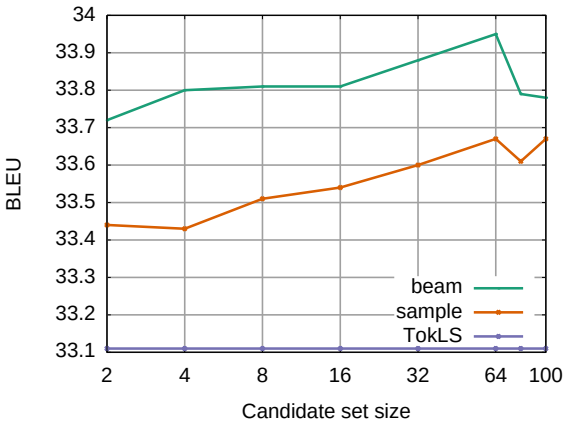


Figure 2: Candidate set generation with beam search and sampling for various candidate set sizes during sequence-level training in terms of validation accuracy. Token-level label smoothing (TokLS) is the baseline.

	BLEU	Δ
MLE	24.03	
+ BSO	26.36	+2.33
MLE Reimplementation	23.93	
+ Risk	26.68	+2.75

Table 5: Comparison to Beam Search Optimization. We report the best likelihood (MLE) and BSO results from Wiseman and Rush (2016), as well as results from our MLE reimplementation and training with Risk. Results based on unnormalized beam search ($k = 5$).

6.6 Comparison to Beam-Search Optimization

Next, we compare classical sequence-level training to the recently proposed Beam Search Optimization (Wiseman and Rush, 2016). To enable a fair comparison, we re-implement their baseline, a single layer LSTM encoder/decoder model with 256-dimensional hidden layers and word embeddings as well as attention and input feeding (Luong et al., 2015). This baseline is trained with Adagrad (Duchi et al., 2011) using a learning rate of 0.05 for five epochs, with batches of 64 sequences. For sequence-level training we initialize weights with the baseline parameters and train

	RG-1	RG-2	RG-L
ABS+ [T]	29.78	11.89	26.97
RNN MLE [T]	32.67	15.23	30.56
RNN MRT [S]	36.54	16.59	33.44
WFE [T]	36.30	17.31	33.88
SEASS [T]	36.15	17.54	33.63
DRGD [T]	36.27	17.57	33.62
TokLS	36.53	18.10	33.93
+ Risk RG-1	36.96	17.61	34.18
+ Risk RG-2	36.65	18.32	34.07
+ Risk RG-L	36.70	17.88	34.29

Table 6: Accuracy on Gigaword abstractive summarization in terms of F-measure Rouge-1 (RG-1), Rouge-2 (RG-2), and Rouge-L (RG-L) for token-level label smoothing, and Risk optimization of all three ROUGE F1 metrics. [T] indicates a token-level objective and [S] indicates a sequence level objectives. ABS+ refers to Rush et al. (2015), RNN MLE/MRT (Ayana et al., 2016), WFE (Suzuki and Nagata, 2017), SEASS (Zhou et al., 2017), DRGD (Li et al., 2017).

with Adam (Kingma and Ba, 2014) for another 10 epochs with learning rate 0.00003 and 16 candidate sequences per training example. We conduct experiments with Risk since it performed best in trial experiments.

Different from other sequence-level experiments (§5), we rescale the BLEU scores in each candidate set by the difference between the maximum and minimum scores of each sentence. This avoids short sentences dominating the sequence updates, since candidate sets for short sentences have a wider range of BLEU scores compared to longer sentences; a similar rescaling was used by Bahdanau et al. (2016).

Table 5 shows the results from Wiseman and Rush (2016) for their token-level likelihood baseline (MLE), best beam search optimization results (BSO), as well as our reimplemented baseline. Risk significantly improves BLEU compared to our baseline at +2.75 BLEU, which is slightly better than the +2.33 BLEU improvement reported for Beam Search Optimization (cf. Wiseman and Rush (2016)). This shows that classical objectives for structured prediction are still very competitive.

6.7 WMT'14 English-French results

Next, we experiment on the much larger WMT'14 English-French task using the same model setup as Gehring et al. (2017b). We TokLSfor 15 epochs and then switch to sequence-level training for another epoch. Table 7 shows that sequence-level training can improve an already very strong model by another +0.37 BLEU. Next, we improve the baseline by adding *self-attention* (Paulus et al., 2017; Vaswani et al., 2017) to the decoder network (TokLS + selfatt) which results in a smaller gain of +0.2 BLEU by Risk. If we train Risk only on the news-commentary portion of the training data, then we achieve state of the art accuracy on this dataset of 41.5 BLEU (Xia et al., 2017).

	valid	test
TokLS	34.06	40.58
+ Risk	34.20	40.95
TokLS + selfatt	34.24	41.02
+ in domain	34.51	41.26
+ Risk	34.30	41.22
+ Risk in domain	34.50	41.47

Table 7: Test and valid BLEU on WMT'14 English-French with and without decoder self-attention.

6.8 Abstractive Summarization

Our final experiment evaluates sequence-level training on Gigaword headline summarization. There has been much prior art on this dataset originally introduced by Rush et al. (2015) who experiment with a feed-forward network (ABS+). Ayana et al. (2016) report a likelihood baseline (RNN MLE) and also experiment with risk training (RNN MRT). Different to their setup we did not find a softmax temperature to be beneficial, and we use beam search instead of sampling to obtain the candidate set (cf. §6.5). Suzuki and Nagata (2017) improve over an MLE RNN baseline by limiting generation of repeated phrases. Zhou et al. (2017) also consider an MLE RNN baseline and add an additional gating mechanism for the encoder. Li et al. (2017) equip the decoder of a similar network with additional latent variables to accommodate the uncertainty of this task.

Table 6 shows that our baseline (TokLS) outperforms all prior approaches in terms of ROUGE-2 and ROUGE-L and it is on par to the best previous result for ROUGE-1. We optimize all three ROUGE metrics separately and find that Risk can further improve our strong baseline. We also compared Risk only training to Weighted on this dataset (cf. §6.2) but accuracy was generally lower on the validation set: RG-1

(36.59 `Risk` only vs. 36.67 `Weighted`), RG-2 (17.34 vs. 18.05), and RG-L (33.66 vs. 33.98).

7 Conclusion

We present a comprehensive comparison of classical losses for structured prediction and apply them to a strong neural sequence to sequence model. We found that combining sequence-level and token-level losses is necessary to perform best, and so is training on candidates decoded with the current model.

We show that sequence-level training improves state-of-the-art baselines both for IWSLT'14 German-English translation and Gigaword abstractive sentence summarization. Structured prediction losses are very competitive to recent work on reinforcement or beam optimization. Classical expected risk can slightly outperform beam search optimization (Wiseman and Rush, 2016) in a like-for-like setup. Future work may investigate better use of already generated candidates since invoking generation for each batch slows down training by a large factor, e.g., mixing with fresh and older candidates inspired by MERT (Och, 2003).

References

Ayana, Shiqi Shen, Yu Zhao, Zhiyuan Liu, Maosong Sun, et al. 2016. Neural headline generation with sentence-wise optimization. *arXiv preprint arXiv:1604.01904* .

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. An Actor-Critic Algorithm for Sequence Prediction. In *arXiv preprint arXiv:1607.07086*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th IWSLT evaluation campaign. In *Proc. of IWSLT*.

Samidh Chatterjee and Nicola Cancedda. 2010. Minimum error rate training by sampling the translation lattice.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language Modeling with Gated Convolutional Networks. In *Proc. of ICML*.

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

Jonas Gehring, Michael Auli, David Grangier, and Yann N Dauphin. 2017a. A Convolutional Encoder Model for Neural Machine Translation. In *Proc. of ACL*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017b. Convolutional Sequence to Sequence Learning. In *Proc. of ICML*.

Kevin Gimpel and Noah Smith. 2010. Softmax-margin crfs: Training log-linear models with cost functions. In *Proc. of ACL*.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia* .

Spence Green, Daniel Cer, and Christopher Manning. 2014. An Empirical Comparison of Features and Tuning for Phrase-based Machine Translation. In *Proc. of WMT*. Association for Computational Linguistics.

Ralf Herbrich, Thore Graepel, and Klaus Obermayer. 1999. Support vector learning for ordinal regression .

Po-Sen Huang, Chong Wang, Dengyong Zhou, and Li Deng. 2017. Neural Phrase-based Machine Translation. In *arXiv preprint arXiv:1706.05565*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *Proc. of ICLR* .

Piji Li, Wai Lam, Lidong Bing, and Zihao Wang. 2017. Deep recurrent generative decoder for abstractive text summarization. *arXiv* .

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*.

Chin-Yew Lin and Franz Josef Och. 2004. Orange: a method for evaluating automatic evaluation metrics for machine translation. In *Proc. of COLING*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proc. of EMNLP*.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. Sapporo, Japan, pages 160–167.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of The 30th International Conference on Machine Learning*. pages 1310–1318.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304* .

Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey E. Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. In *Proc. of ICLR Workshop*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level Training with Recurrent Neural Networks. In *Proc. of ICLR*.

Antti-Veikko I Rosti, Bing Zhang, Spyros Matsoukas, and Richard Schwartz. 2010. BBN System Description for WMT10 System Combination Task. In *Proc. of WMT*. Association for Computational Linguistics, pages 321–326.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proc. of EMNLP*.

Tim Salimans and Diederik P Kingma. 2016. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *arXiv preprint arXiv:1602.07868* .

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proc. of ACL*.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum Risk Training for Neural Machine Translation. In *Proc. of ACL*.

David A. Smith and Jason Eisner. 2006. Minimum Risk Annealing for Training Log-Linear Models. In *Proc. of ACL*.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent Neural Networks from overfitting. *JMLR* 15:1929–1958.

Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. 2013. On the importance of initialization and momentum in deep learning. In *ICML*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Proc. of NIPS*. pages 3104–3112.

Jun Suzuki and Masaaki Nagata. 2017. Cutting-off redundant repeating generations for neural abstractive summarization. *arXiv preprint arXiv:1701.00138* .

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2015. Rethinking the inception architecture for computer vision. *arXiv* .

Ben Taskar, Carlos Guestrin, and Daphne Koller. 2003. Max-margin markov networks. In *NIPS*.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research* 6:1453—-1484.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv* .

R. J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8:229—-256.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proc. of ACL*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv preprint arXiv:1609.08144* .

Yingce Xia, Fei Tian, Lijun Wu, Jianxin Lin, Tao Qin, Nenghai Yu, and Tie-Yan Liu. 2017. Deliberation networks: Sequence generation beyond one-pass decoding. In *Proc. of NIPS*.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. *arXiv* .

Deep Dirichlet Multinomial Regression

Adrian Benton Mark Dredze
Center for Language and Speech Processing
Johns Hopkins University
Baltimore, MD 21218, USA
{adrian,mdredze}@cs.jhu.edu

Abstract

Dirichlet Multinomial Regression (*DMR*) and other supervised topic models can incorporate arbitrary document-level features to inform topic priors. However, their ability to model corpora are limited by the representation and selection of these features – a choice the topic modeler must make. Instead, we seek models that can learn the feature representations upon which to condition topic selection. We present *deep Dirichlet Multinomial Regression* (*dDMR*), a generative topic model that simultaneously learns document feature representations and topics. We evaluate *dDMR* on three datasets: New York Times articles with fine-grained tags, Amazon product reviews with product images, and Reddit posts with subreddit identity. *dDMR* learns representations that outperform *DMR* and *LDA* according to heldout perplexity and are more effective at downstream predictive tasks as the number of topics grows. Additionally, human subjects judge *dDMR* topics as being more representative of associated document features. Finally, we find that supervision leads to faster convergence as compared to an *LDA* baseline and that *dDMR*'s model fit is less sensitive to training parameters than *DMR*.

1 Introduction

Fifteen years of research on topic models, starting from Latent Dirichlet Allocation (*LDA*) (Blei et al., 2003), have led to a variety of models for numerous data settings. These models identify sets (distributions) of related words that reflect semantic topics in a large corpus of text data. Topic models are now routinely used in the social sciences and humanities to analyze text collections (Schmidt, 2012).

Document collections are often accompanied by metadata and annotations, such as a book's author, an article's topic descriptor tags, images associated with a product review, or structured patient information associated with clinical records. These document-level annotations can provide additional supervision for guiding topic model learning. Additional information can be integrated into topic models using either *downstream* or *upstream* models. Downstream models, such as supervised *LDA* (Mcauliffe and Blei, 2008), assume that these additional document features are generated from each document's topic distribution. These models are most helpful when you desire topics that are predictive of the output, such as models for predicting the sentiment of product reviews. Upstream models, such as Dirichlet Multinomial Regression (*DMR*), condition each document's topic distribution on document features, such as author (Rosen-Zvi et al., 2004), social network (McCallum et al., 2007), or document labels (Ramage et al., 2009). Previous work has demonstrated that upstream models tend to outperform downstream models in terms of model fit, as well as extracting topics that are useful in prediction of related tasks (Benton et al., 2016).

DMR is an upstream topic model with a particularly attractive method for incorporating arbitrary document features. Rather than defining specific random variables in the graphical model for each new document feature, *DMR* treats the document annotations as features in a log-linear model. The log-linear model parameterizes the Dirichlet prior for the document's topic distribution, making the Dirichlet's hyperparameter (typically α) document-specific. By making no assumptions on model structure of new random variables, *DMR* is flexible to incorporating different types of features.

Despite this flexibility, *DMR* models are typically restricted to a small number of document features. Several reasons account for this restriction: 1) Many text corpora only have a small number of document-level features; 2) Model hyperparameters become less interpretable as the dimensionality grows; and 3) *DMR* is liable to overfit the hyperparameters when the dimensionality of document features is high. In practice, applications of *DMR* are limited to settings with a small number of features, or where the analyst selects a few meaningful features

Proceedings of NAACL-HLT 2018, pages 365–374
New Orleans, Louisiana, June 1 - 6, 2018.

by hand.

A solution to this restriction is to learn low-dimensional representations of document features. Neural networks have shown wide-spread success at learning generalizable representations, often obviating the need for hand designed features (Collobert and Weston, 2008). A prime example is word embedding features in natural language processing, which supplant traditional lexical features (Brown et al., 1992; Mikolov et al., 2013; Pennington et al., 2014). Jointly learning networks that construct feature representations along with the parameters of a standard NLP model has become a common approach. For example, (Yu et al., 2015) used a tensor decomposition to jointly learn features from both word embeddings and traditional NLP features, along with the parameters of a relation extraction model. Additionally, neural networks can handle a variety of data types, including text, images and general metadata features. This makes them appropriate for addressing dimensionality reduction in *DMR*.

We propose **deep** Dirichlet Multinomial Regression (*dDMR*), a model that extends *DMR* by introducing a deep neural network that learns a transformation of the input metadata into features used to form the Dirichlet hyperparameter. Whereas *DMR* parameterizes the document-topic priors as a log-linear function of document features, *dDMR* jointly learns a feature representation for each document along with a log-linear function that best captures the distribution over topics. Since the function mapping document features to topic prior is a neural network, we can jointly optimize the topic model and the neural network parameters by gradient ascent and back-propagation. We show that *dDMR* can use network architectures to better fit text corpora with high-dimensional document features as compared to other supervised topic models. The topics learned by *dDMR* are judged as being more representative of document features by human subjects. We also find that *dDMR* tends to converge in many fewer iterations than *LDA*, and also does not suffer from tuning difficulties that *DMR* encounters when applied to high-dimensional document features.

2 Model

Our model builds on the generative model of *DMR*: an LDA-style topic model that replaces the hyperparameter (vector) of the topic distribution Dirichlet prior with a hyperparameter that is output from a log-linear model given the document features. Our model deep DMR (*dDMR*) replaces this log-linear model with an arbitrary function f that maps a real-valued vector of dimension F to a representation of dimension K. For simplicity we make no assumptions on the choice of this function, only

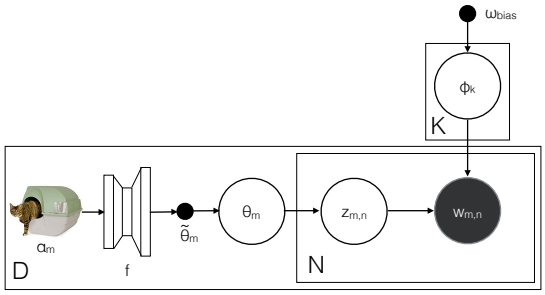


Figure 1: The graphical model for *dDMR*. f is shown as a feedforward fully-connected network, and the document features are given by the image (a cat carrier).

that it can be optimized to minimize a cost on its output by gradient ascent. In practice, we define this function as a neural network, where the architecture of this network is informed by the type of document features, e.g. a convolutional neural network for images. We use neural networks since they are expressive, generalize well to unseen data, and can be jointly trained using straightforward gradient ascent with back-propagation.

The generative story for *dDMR* is as follows:

1. Representation function $f \in \mathbb{R}^F \rightarrow \mathbb{R}^K$
2. Topic-word prior parameters: $\omega^{bias} \in \mathbb{R}^V$
3. For each document m with features $\alpha_m \in \mathbb{R}^F$, generate document prior:
   (a) $\widetilde{\theta}_m = exp(f(\alpha_m))$
   (b) $\theta_m \sim Dirichlet(\widetilde{\theta}_m)$
4. For each topic k, generate word distribution:
   (a) $\widetilde{\phi}_k = exp(\omega^{bias})$
   (b) $\phi_k \sim Dirichlet(\widetilde{\phi}_k)$
5. For each token (m, n), generate data:
   (a) Topic (unobserved): $z_{m,n} \sim \theta_m$
   (b) Word (observed): $w_{m,n} \sim \phi_{z_{m,n}}$

where V is the vocabulary size and K are the number of topics. In practice, the document features need not be restricted to fixed-length feature vectors, e.g. f may be an RNN that maps from a sequence of characters to a fixed length vector in $\mathbb{R}^k$. *DMR* is a special case of *dDMR* with the choice of a linear function for f. Figure 1 displays the graphical model diagram for *dDMR*.

2.1 Inference and Parameter Estimation

We infer the random variables of the topic model using collapsed Gibbs sampling, and estimate the model parameters using gradient ascent with back-propagation. We use alternating optimization: one

iteration of collapsed Gibbs sampling (sample topics for each word) and then an update of the parameters of f by gradient ascent to maximize the log-likelihood of the tokens and topic assignments. Given the parameters, the sampling step remains unchanged from *LDA* (Griffiths and Steyvers, 2004). The network parameters are estimated via back-propagation through the network for a fixed sample. Eq. 1 shows the gradient of the data log-likelihood, $\mathscr{L}$, with respect to $\widetilde{\theta}_{m,k} = exp(f(\alpha_m)_k)$, the prior weight of topic k for document m. ψ is the digamma function (derivative of the log-gamma function), n_m is the number of tokens in document m, and $n_{m,k}$ is the count of how many tokens topic k was assigned to in document m.

$$\frac{\delta \mathscr{L}}{\delta \widetilde{\theta}_{m,k}} = \psi(\sum_{k=1}^{K} \widetilde{\theta}_{m,k}) - \psi(\sum_{k=1}^{K} \widetilde{\theta}_{m,k} + n_m) + \psi(\widetilde{\theta}_{m,k} + n_{m,k}) - \psi(\widetilde{\theta}_{m,k}) \quad (1)$$

3 Data

We explore the flexibility of our model by considering three different datasets that include different types of metadata associated with each document. For each dataset, we describe the documents and metadata.

New York Times The New York Times Annotated Corpus (Sandhaus, 2008) contains articles with extensive metadata used for indexing by the newspaper. For supervision, we used the "descriptor" tags associated with each article assigned by archivists. These tags reflect the topic of an article, as well as organizations or people mentioned in the article. We selected all articles published in 1998, and kept those tags that were associated with at least 3 articles in that year – 2424 unique tags. 20 of the 200 most frequent tags were held out from training for validation purposes: { *"education and schools", "law and legislation", "advertising", "budgets and budgeting", "freedom and human rights", "telephones and telecommunications", "bombs and explosives", "sexual harassment", "reform and reorganization", "teachers and school employees", "tests and testing", "futures and options trading", "boxing", "firearms", "company reports", "embargoes and economic sanctions", "hospitals", "states (us)", "bridge (card game)",* and *"auctions"*}. Articles contained a mean of 2.1 tags, with 738 articles not containing any of these tags. Tags were represented using a one-hot encoding.

Articles were tokenized by non-alphanumeric characters and numerals were replaced by a special token. Words occurring in more than 40% of documents were removed, and only the 15,000 most frequent types were retained. There were a total of 89,397 articles with an average length of 158 tokens per article.

Amazon Reviews The Amazon product reviews corpus(McAuley and Yang, 2016) contains reviews of products as well as images of the product. We sampled 100,000 Amazon product reviews: 20,000 reviews sampled uniformly from the *Musical Instruments*, *Patio, Lawn, & Garden*, *Grocery & Gourmet Food*, *Automotive*, and *Pet Supplies* product categories. We hypothesize that knowing information about the product's appearance will indicate which words appear in the review, especially for product images occurring in these categories. 66 of the reviews we sampled contained only highly infrequent tokens, and were therefore removed from our data, leaving 99,934 product reviews. Articles were preprocessed identically to the New York Times data.

We include images as supervision by using the 4096-dimensional second fully-connected layer of the Caffe convolutional neural network reference model, trained to predict ImageNet object categories[1]. Using these features as supervision to *dDMR* is similar to fine-tuning a pre-trained CNN to predict a new set of labels. Since the Caffe reference model is already trained on a large corpus of images, we chose to fine-tune only the final layers so as to learn a transformation of the already learned representation.

Reddit We selected a sample of Reddit posts made in January 2016. A standard stop list was used to remove frequent function words and we restricted the vocabulary to the 30,000 most frequent types. We restricted posts made to subreddits, collections of topically-related threads, with at least ten comments in this month (26,830 subreddits), and made by users with at least five comments across these subreddits (total of 1,351,283 million users). We then sampled 10,000 users uniformly at random and used all their comments as a corpus, for a total of 389,234 comments over 7,866 subreddits (token length mean: 16.3, median: 9)[2].

This corpus differs from the others in two ways. First, Reddit documents are very short, which is problematic for topic models that rely on detecting correlations in token use. Second, the Reddit metadata that may be useful for topic modeling is necessarily high-dimensional (e.g. subreddit identity, a proxy for topical content). *DMR* may have trouble exploiting high-dimensional supervision.

4 Experiments

Model Estimation We used the same procedure for training topic models on each dataset. Hyperparameter gradient updates were performed after

[1]Features used directly from http://jmcauley.ucsd.edu/data/amazon/

[2]The sampled comment IDs can be found here: https://github.com/abenton/deep-dmr/blob/master/resources/reddit_comment_ids.txt

a burnin period of 100 Gibbs sampling iterations. Hyperparameters were updated with the adaptive learning rate algorithm Adadelta (Zeiler, 2012), with a tuned base learning rate and fixed $\rho = 0.95$[3]. All models were trained for a maximum of 15,000 epochs, with early stopping if heldout perplexity showed no improvements after 200 epochs (evaluated once every 20 epochs). Hyperparameters were fit on every other token in the corpus, and (heldout) log-likelihood/perplexity was calculated on the remaining tokens.

For the architecture of the *dDMR* model we used single-hidden-layer multi-layer perceptrons (MLPs), with rectified linear unit (ReLU) activations on the hidden layer, and linear activation on the output layer. We sampled three architectures for each dataset, by drawing layer widths independently at random from $[10, 500]$, and also included two architectures with $(50, 10)$ and $(100, 50)$, *(hidden, output)* layers [4] . We compare the performance of *dDMR* to *DMR* trained on the same feature set as well as *LDA*.

For the New York Times dataset, we also compare *dDMR* to *DMR* trained on features after applying principal components analysis (PCA) to reduce the dimensionality of descriptor feature supervision, sweeping over PCA projection width in $\{10, 50, 100, 250, 500, 1000\}$. Comparing performance of *dDMR* to PCA-reduced *DMR* tests two modeling choices. First, it tests the hypothesis that explicitly learning a representation for document annotations to maximize data likelihood produces a "better-fit" topic model than learning this annotation representation in unsupervised fashion – a two-step process. It also lets us determine if a linear dimensionality reduction technique is sufficient to learning a good feature representation for topic modeling, as opposed to learning a non-linear transformation of the document supervision. Note that we cannot apply PCA to reduce the dimensionality for subreddit id in Reddit since it is a one-hot feature.

Documents in each dataset were partitioned into ten equally-sized folds. Model training parameters of L1 and L2 regularization penalties on feature weights for *DMR* and *dDMR* and the base learning rate for each model class were tuned to minimize heldout perplexity on the first fold. These were tuned *independently for each model*, with number of topics fixed to 10, and *dDMR* architecture fixed to narrow layer widths (50, 10). Model selection was based on the macro-averaged performance on the next eight folds, and we report performance on the remaining fold. We selected models separately for each evaluation metric. For *dDMR*, model selection amounts to selecting the document prior architecture, and for *DMR* with PCA-reduced feature supervision, model selection involved selecting the PCA projection width.

Evaluation Each model was evaluated according to heldout perplexity, topic coherence by normalized pointwise mutual information (NPMI) (Lau et al., 2014), and a dataset-specific predictive task.

Heldout perplexity was computed by only aggregating document-topic and topic-word counts from every other token in the corpus, and evaluating perplexity on the remaining heldout tokens. This corresponds to the "document completion" evaluation method as described in (Wallach et al., 2009), where instead of holding out the words in the second half of a document, every other word is held out.

NPMI (Lau et al., 2014) computes a an automatic measure of topic quality, the sum of pointwise mutual information between pairs of m most likely words normalized by the negative log of each pair jointly occurring within a document (Eq. 2). We calculated this topic quality metric on the top 20 most probable words in each topic, and averaged over the most coherent 1, 5, 10, and over all learned topics. However, models were selected to only maximize average NPMI over all topics.

$$\texttt{NPMI} = \sum_{i=1}^{m} \sum_{j=i+1}^{m} \frac{\log \frac{P(w_i, w_j))}{P(w_i)P(w_j)}}{-\log P(w_i, w_j)} \quad (2)$$

For prediction tasks, we used the sampled topic distribution associated with a document, averaged over the last 100 iterations, as features to predict a document-level label. For New York Times articles we predicted 10 of the 200 most frequent descriptor tags restricting to articles with exactly one of these descriptors. For Amazon, we predicted the product category a document belonged to (one of five), and for Reddit we predicted a heldout set of document subreddit IDs. In the case of Reddit, these heldout subreddits were 10 out of the 100 most prevalent in our data, and were held out similar to the New York Times evaluation. SVM models were fit on inferred topic distribution features and were then evaluated according to accuracy, F1-score, and area under the ROC curve. The SVM slack parameter was tuned by 4-fold cross-validation on 60% of the documents, and evaluated on the remaining 40%.

We also collected human topic judgments using Amazon Mechanical Turk (Callison-Burch and

[3]We found this adaptive learning rate algorithm improved model fit in many fewer iterations than gradient descent with tuned step size and decay rate for all models.

[4]We included these two very narrow architectures to ensure that some architecture learned a small feature representation, generalizing better when features are very noisy or only provide a weak signal for topic modeling. We restricted ourselves to only train *dDMR* models with single-hidden-layer MLPs in the priors for simplicity and to avoid model fishing.

Z	Model	NYT		Amazon		Reddit	
10	*LDA*	3429	(5)	2300	(7)	3811	(15)
	DMR	**3385**	(6)	2475	(9)	3753	(10)
	DMR-PCA	3417	(8)				
	dDMR	3395	(7)	**2272**	(68)	**3624**	(13)
20	*LDA*	3081	(6)	2275	(7)	3695	(19)
	DMR	3018	(4)	2556	(48)	3650	(8)
	DMR-PCA	3082	(8)				
	dDMR	**3023**	(7)	**2222**	(7)	**3581**	(16)
50	*LDA*	2766	(8)	2269	(9)	3695	(17)
	DMR	2797	(34)	2407	(20)	3640	(40)
	DMR-PCA	2773	(9)				
	dDMR	**2657**	(8)	**2197**	(13)	**3597**	(17)
100	*LDA*	2618	(8)	2246	(10)	3676	(19)
	DMR	2491	(27)	2410	(75)	3832	(30)
	DMR-PCA	2644	(52)				
	dDMR	**2433**	(10)	**2215**	(6)	**3642**	(18)
200	*LDA*	2513	(8)	2217	(7)	3653	(19)
	DMR	2630	(13)	2480	(65)	3909	(15)
	DMR-PCA	2525	(14)				
	dDMR	**2394**	(9)	**2214**	(12)	**3587**	(11)

Table 1: Test fold heldout perplexity for each dataset and model for number of topics Z. Standard error of mean heldout perplexity over all cross-validation folds in parentheses.

Dredze, 2010). Each subject was presented with a human-readable version of the features used for supervision. For New York Times articles we showed the descriptor tags, for Amazon the product image, and for Reddit the name, title, and public description of the subreddit. We showed the top twenty words for the most probable topic sampled for the document with those features, as learned by two different models. One topic was learned by *dDMR* and the other was either learned by *LDA* or *DMR*. The topics presented were from the 200-topic model architecture that maximized NPMI on development folds. Annotators were asked "*to choose which word list best describes a document* ..." with the displayed features. The topic learned by *dDMR* was shuffled to lie on either the right or left for each Human Intelligence Task (HIT). We obtained judgments on 1,000 documents for each dataset and each model evaluation pair – 6,000 documents in all. This task can be difficult for many of the features, which may be unclear (e.g. descriptor tags without context) or difficult to interpret (e.g. images of automotive parts). We excluded the document text since we did not want subjects to evaluate topic quality based on token overlap with the actual document.

5 Results

Model Fitting *dDMR* achieves lower perplexity than *LDA* or *DMR* for most combinations of number of topics and dataset (Table 1). It is striking that *DMR* achieves higher perplexity than *LDA* in many of these conditions. This is particularly true for the Amazon dataset, where *DMR* consistently lags behind *LDA*. *Supervision alone does not improve topic model fit if it is too high-dimensional for learning.* Perplexity is higher on the Reddit data for all models due to both a larger vocabulary size and shorter documents.

It is also worth noting that finding a low-dimensional linear projection of the supervision features with PCA does not improve model fit as well as *dDMR*. *dDMR* benefits both from joint learning to maximize corpus log-likelihood and possibly by the flexibility of learning non-linear projection (through the hidden layer ReLU activations).

Another striking result is the difference in speed of convergence between the supervised models and *LDA* (Figure 2). Even supervision that provides a weak signal for topic modeling, such as Amazon product image features, can speed convergence over *LDA*. In certain cases (Figure 2 left), training *dDMR* for 1,000 iterations results in a lower perplexity model than *LDA* trained for over 10,000 iterations.

In terms of actual run time, parallelization of model training differs between the supervised model and *LDA*. Gradient updates necessary for learning the representation can be trivially distributed across multiple cores using optimized linear algebra libraries (e.g. BLAS), mitigating the additional cost incurred by hyperparameter updates in supervised models. In contrast, the Gibbs sampling iterations can also be parallelized, but not as easily, ultimately making resampling topics the most expensive step in model training. Because of this, the potential difference in runtime for a single iteration between *dDMR* and *LDA* is small, with the former converging in far fewer iterations. In our experiments, per iteration time taken by *DMR* or *dDMR* was at most twice as long as *LDA* across all experiments.

dDMR performance is also insensitive to training parameters relative to *DMR*. While *DMR* requires heavy L1 and L2 regularization and a very small step size to achieve low heldout perplexity, *dDMR* is relatively insensitive to the penalty on regularization and benefits from a higher base learning rate (Figure 3). We found that dDMR *is easier to tune than* DMR, requiring less exploration of the training parameters. This is also corroborated by higher variance in perplexity achieved by *DMR* across different cross-validation folds (Table 1).

Topic Quality Results for the automatic topic quality evaluation, NPMI, are mixed across datasets. In many cases, *LDA* and *DMR* score highly according to NPMI, despite achieving higher heldout perplexity than *dDMR* (Table 2). This may not be surprising as previous work has found that perplexity does not correlate well with human judgments of topic coherence (Lau et al., 2014).

However, in the human evaluation, subjects find that *dDMR*-learned topics are more representative of document annotations than *DMR* (Table 3). While subjects only statistically significantly favored *dDMR* models over *LDA* on the Reddit data, they favored *dDMR* topics over *LDA* across all datasets, and significantly preferred *dDMR* top-

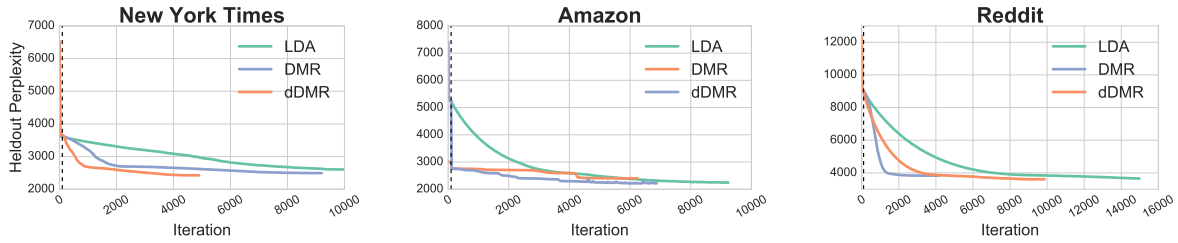


Figure 2: Heldout perplexity as a function of iteration for lowest-perplexity models with $Z = 100$. The vertical dashed line indicates when models are burned in and hyperparameter optimization begins.

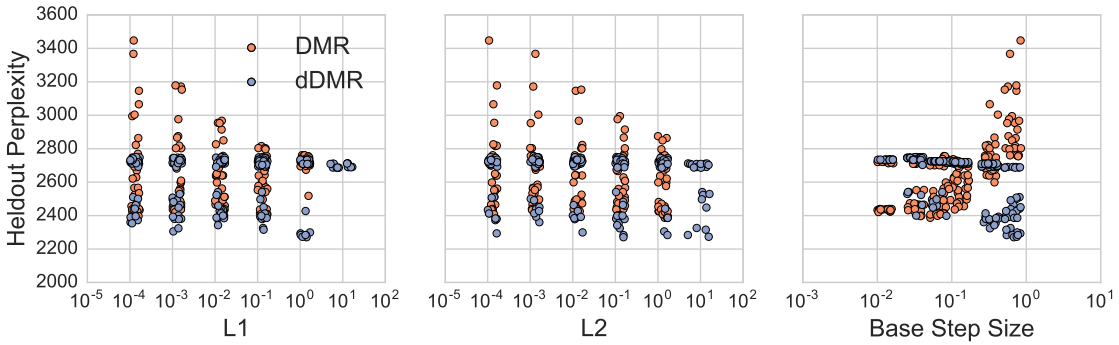


Figure 3: Heldout perplexity on the Amazon data tuning fold for *DMR* (orange) and *dDMR* (purple) with a (50, 10) layer architecture as a function of training parameters: L1, L2 feature weight regularization, and base learning rate. All models were trained for a fixed 5,000 iterations, with horizontal jitter added to each point.

		New York Times				Amazon				Reddit			
Z	Model	1	5	10	Overall	...				...			
10	*LDA*	52	49	43	43	25	23	20	20	**125**	**82**	**56**	**56**
	DMR	53	50	42	42	**58**	**43**	**31**	**31**	43	35	30	30
	DMR-PCA	**63**	**53**	**45**	**45**								
	dDMR	57	51	44	44	24	21	19	19	109	62	46	46
20	*LDA*	62	59	54	45	27	25	23	20	**121**	**87**	**59**	**42**
	DMR	63	60	56	45	66	56	**53**	**43**	81	49	41	34
	DMR-PCA	**76**	**61**	**57**	**47**								
	dDMR	69	60	55	45	**97**	**61**	**53**	40	109	66	49	38
50	*LDA*	80	66	62	44	30	27	25	20	**135**	**96**	**64**	34
	DMR	80	**67**	**63**	**46**	**136**	**81**	**73**	**58**	51	46	41	33
	DMR-PCA	**82**	**67**	63	45								
	dDMR	76	65	61	45	71	65	62	44	121	74	54	**36**
100	*LDA*	77	71	66	40	58	34	30	20	135	74	54	31
	DMR	**80**	**74**	70	**45**	**147**	**83**	**75**	**59**	111	67	50	**34**
	DMR-PCA	79	69	**75**	**45**								
	dDMR	77	73	68	44	68	67	66	55	**135**	**78**	**55**	31
200	*LDA*	78	74	70	36	60	39	34	18	**135**	**100**	**67**	29
	DMR	91	**76**	80	42	69	67	67	**61**	132	84	59	**32**
	DMR-PCA	**94**	**76**	**81**	42								
	dDMR	78	70	66	**45**	**85**	**73**	**69**	39	**135**	87	61	30

Table 2: Top-1, 5, 10, and overall topic NPMI across all datasets. Models that maximized overall NPMI across dev folds were chosen and the best-performing model is in bold.

	LDA	*DMR*
New York Times	51.1%	51.9%
Amazon	51.9%	61.4%*
Reddit	55.5%*	57.6%*

Table 3: % HITs where humans preferred *dDMR* topics as more representative of document supervision than the competing model. $*$ denotes statistical significance according to a one-tailed binomial test at the $p = 0.05$ level.

ics over *DMR* on two of the three datasets. This is contrary to themodel rankings according to NPMI, which suggest that *DMR* topics are often higher quality when it comes to human interpretability.

We also qualitatively explored the product image representations *DMR* and *dDMR* learned on the Amazon data. To do so, we computed and normalized the prior document distribution for a sample of documents for lowest perplexity *DMR* and *dDMR* $Z = 200$ topic models: $p(k|m) = \frac{\widetilde{\theta}_m}{\sum_{k=1}^{Z} \widetilde{\theta}_{m,k}}$, the prior probability of sampling topic k, conditioned on the features for document m. We then marginalize over topics to yield the conditional probability of a word w given document m: $p(w|m) = \sum_{k=1}^{Z} p(w|k)p(k|m)$. Table 4 contains a sample of these probable words given document supervision. We find that *dDMR* identifies words likely to appear in a review of the product pictured. However, some images lead *dDMR* down a garden path. For example, a bottle of "Turtle Food" should not be associated with words for human consumables like "coffee" and "chocolate", despite the container resembling some of these products. However, the image-specific document priors *DMR* learned are not as sensitive to the actual product image as those learned by *dDMR*. The prior conditional probabilities $p(w|m)$ for "Turtle Food", "Slushy Magic Cup", and "Rawhide Dog Bones" product images are all ranked identically by *DMR*.

Predictive Performance Finally, we consider the utility of the learned topic distributions for downstream prediction tasks, a common use of topic models. Although token perplexity is a standard measure of topic model fit, it has no direct relationship with how topic models are typically used: to identify consistent themes or reduce the dimensionality of a document corpus. We found that features based on topic distributions from *dDMR* outperform *LDA* and *DMR* on the Amazon and Reddit data when the number of topics fit is large, although they fail to outperform *DMR* on New York Times (Table 5). Heldout perplexity is strongly correlated with predictive performance, with a Pearson correlation coefficient, $\rho = 0.898$ between F1-score and heldout perplexity on the Amazon data. This strong correlation is likely due to the tight relationship between words used in product reviews and product category: a model that assigns high likelihood to a words in a product review corpus should also be informative of the product categories. Prior work showed that upstream supervised topic models, such as *DMR*, learn topic distributions that are effective at downstream prediction tasks (Benton et al., 2016). We find that topic distributions learned by *dDMR* improve over *DMR* in certain cases, particularly as the number of topics increases.

6 Related Work

With the widespread adoption of neural networks, others have sought to combine topic and neural models. One line of work replaces generative, *LDA*-based, topic models with discriminatively-trained models based on neural networks. (Cao et al., 2015) model θ and ϕ using neural networks with softmax output layers and learn network parameters that maximize data likelihood. They also learn n-gram embeddings to identify topics whose elements are not restricted to unigrams. (Chen et al., 2015) similarly expresses the (smoothed) supervised LDA (Mcauliffe and Blei, 2008) generative model as a neural network, and give an algorithm to discriminatively train it. (Wan et al., 2012) take a similar approach to *dDMR* where they use a neural network to extract image representations that maximize the probability of SIFT descriptors extracted from the image. However, this model is used for image classification, not for exploring a corpus of documents as is typical of topic models. These models are computationally attractive in that they avoid approximating the posterior distribution of topic assignments given tokens by dropping the assumption that θ and ϕ are drawn from Dirichlet priors. Model fitting is performed by back-propagation of a max-margin cost. In contrast, we use neural networks to learn feature representations for documents, not as a replacement for the *LDA* generative story. This is similar to variants of SPRITE (Paul and Dredze, 2015), where many document-level factors are combined to generate a document-topic prior. In contrast to several of these models, the core of our topic model remains unchanged, meaning that *dDMR* is agnostic to many other extensions of *LDA*.

There has been extensive work in modeling both textual and visual topics. Models such as Corr-LDA (Blei and Jordan, 2003) suppose that a text document and associated image features are generated by a shared latent topic. This property is shared by other topic models over images, such as STM-TwitterLDA (Cai et al., 2015) and (Zhang et al., 2015). While these models try to model images, we instead use images in the Amazon data to better estimate topic distributions.

Our experiment on using images to model Ama-

Image	Item	*dDMR* Probable Words	*DMR* Probable Words
	**Guitar Foot Rest**	**grill** easy cover well fit **mower** fits **job gas hose** light **heavy easily stand back** nice works **use enough pressure**	fit easy well works **car** light **sound quality work guitar would 0000** cover nice **looks bought install battery 00** fits
	**Bark Collar**	fit battery 0000 light install car sound easy work **unit amp** 00 **lights mic power** works **000 took replace installed**	fit easy **well** works car light work **quality** sound **would guitar** 0000 **cover nice bought looks** install battery 00 **fits**
	**Turtle Food**	taste coffee flavor food like love cat tea product tried dog eat chocolate **litter** cats **good best bag** sugar loves	taste coffee dog like love flavor food cat product tea cats tried **water dogs** loves eat chocolate **toy mix** sugar
	**Slushy Magic Cup**	food taste cat coffee flavor love like dog tea **litter** cats eat tried product chocolate loves **bag** good **best smell**	taste coffee dog like love flavor food cat product tea cats tried **water dogs** loves eat chocolate **toy mix** good
	**Rawhide Dog Bones**	food cat dog cats **litter** dogs loves love product **smell** eat **box** tried **pet bag hair** taste **vet** like **seeds**	taste **coffee** dog like love **flavor** food cat product **tea** cats tried **water** dogs loves eat **chocolate toy mix good**
	**Instrument Cable**	sound **amp** guitar **mic pedal sounds price volume** quality **cable great bass microphone strings** music **play recording 000 tone** unit	sound guitar **fit easy well 0000 works car** quality **light** music **cover work one set nice looks 00 install** unit

Table 4: Top twenty words associated with each of the product images – learned by *dDMR* vs. *DMR* ($Z = 200$). These images were drawn at random from the Amazon corpus (no cherry-picking involved). Word lists were generated by marginalizing over the prior topic distribution associated with that image, and then normalizing each word's probability by subtracting off its mean marginal probability across all images in the corpus. This is done to avoid displaying highly frequent words. Words that differ between each model's ranked list are in bold.

zon product reviews resembles work on image caption generation, yet the similarity is superficial. The relationship between an image and its caption is relatively tight (Fang et al., 2015) – objects in the image will likely be referenced in the caption. For Amazon product reviews, visual features of the product, like color, may be explicitly mentioned in the review, but then again, they may not. Also, the aim of topic models is to extract common themes of co-occurring words, and how those themes are distributed across each document. The similarity between our work and captioning lies only in the fact that we extract image features from a CNN trained as an object recognizer to inform document-topic distributions.

7 Conclusion

We present deep Dirichlet Multinomial Regression, a supervised topic model which both learns a representation of document-level features and how to use that representation for informing a topic distribution. We demonstrate the flexibility of our model on three corpora with different types of metadata: topic descriptor tags, images, and subreddit IDs. *dDMR* is better able to fit text corpora with high-dimensional supervision compared to *LDA* or *DMR*. Furthermore, we find that document supervision greatly reduces the number of Gibbs sampling iterations for a topic model to converge, and that the *dDMR* prior architecture makes it more robust to training parameters than *DMR*. We also find that the topic distributions learned by *dDMR* are more predictive of external

		New York Times			Amazon			Reddit		
Z	Model	F1	Accuracy	AUC	...			...		
10	*LDA*	0.208	**0.380**	0.767	**0.662**	**0.667**	**0.891**	0.130	0.276	0.565
	DMR	0.236	0.367	0.781	0.311	0.407	0.619	0.092	0.229	**0.597**
	DMR-PCA	**0.280**	0.347	0.758						
	dDMR	0.154	0.347	**0.790**	0.608	0.656	0.864	**0.170**	**0.300**	0.596
20	*LDA*	0.315	0.463	0.784	0.657	0.659	0.887	**0.121**	0.258	**0.579**
	DMR	0.319	0.477	0.805	0.294	0.405	0.647	0.057	0.245	0.520
	DMR-PCA	0.343	**0.540**	**0.831**						
	dDMR	**0.424**	0.523	0.797	**0.706**	**0.711**	**0.911**	0.071	**0.274**	0.566
50	*LDA*	0.455	0.613	0.849	0.630	0.634	0.870	0.131	0.199	0.542
	DMR	0.478	0.650	0.877	0.396	0.499	0.619	**0.145**	0.261	**0.580**
	DMR-PCA	0.505	**0.667**	**0.887**						
	dDMR	**0.507**	0.657	0.856	**0.716**	**0.726**	**0.916**	0.118	**0.272**	0.551
100	*LDA*	0.531	0.657	0.874	0.646	0.649	0.874	0.148	0.201	0.538
	DMR	0.552	0.683	0.898	0.392	0.463	0.688	0.107	0.233	0.512
	DMR-PCA	**0.602**	**0.687**	**0.917**						
	dDMR	0.514	0.653	0.893	**0.650**	**0.660**	**0.893**	**0.172**	**0.316**	**0.614**
200	*LDA*	0.566	0.683	0.903	0.646	0.651	0.882	0.111	0.227	0.517
	DMR	0.576	0.670	**0.917**	0.288	0.401	0.697	0.089	0.229	0.499
	DMR-PCA	**0.648**	**0.762**	0.915						
	dDMR	0.605	0.730	0.903	**0.716**	**0.721**	**0.909**	**0.198**	**0.323**	**0.580**

Table 5: Top F-score, accuracy, and AUC on prediction tasks for all datasets.

document labels such as known topic tags or product category as the number of topics grows and that *dDMR* topics are judged as more representative of the document metadata by human subjects. Source code for training *dDMR* can be found at `http://www.github.com/abenton/deep-dmr`.

References

Adrian Benton, Michael J Paul, Braden Hancock, and Mark Dredze. 2016. Collective supervision of topic models for predicting surveys with social media. In *Proceedings of the AAAI Conference on Artificial Intelligence*. pages 2892–2898.

David M Blei and Michael I Jordan. 2003. Modeling annotated data. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*. ACM, pages 127–134.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research* 3(Jan):993–1022.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics* 18(4):467–479.

Hongyun Cai, Yang Yang, Xuefei Li, and Zi Huang. 2015. What are popular: exploring twitter features for event detection, tracking and visualization. In *Proceedings of the 23rd ACM International Conference on Multimedia*. ACM, pages 89–98.

Chris Callison-Burch and Mark Dredze. 2010. Creating speech and language data with amazon's mechanical turk. In *NAACL-HLT Workshop on Creating Speech and Language Data With Mechanical Turk*. pages 1–12.

Ziqiang Cao, Sujian Li, Yang Liu, Wenjie Li, and Heng Ji. 2015. A novel neural topic model and its supervised extension. In *Proceedings of the AAAI conference on Artificial Intelligence*. pages 2210–2216.

Jianshu Chen, Ji He, Yelong Shen, Lin Xiao, Xiaodong He, Jianfeng Gao, Xinying Song, and Li Deng. 2015. End-to-end learning of LDA by mirror-descent back propagation over a deep architecture. In *Advances in Neural Information Processing Systems*. pages 1765–1773.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*. ACM, pages 160–167.

Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C Platt, et al. 2015. From captions to visual concepts and back. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 1473–1482.

Thomas L Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences* 101(suppl 1):5228–5235.

Jey Han Lau, David Newman, and Timothy Baldwin. 2014. Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. pages 530–539.

Julian McAuley and Alex Yang. 2016. Addressing complex and subjective product-related queries with customer reviews. In *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, pages 625–635.

Jon D Mcauliffe and David M Blei. 2008. Supervised topic models. In *Advances in Neural Information Processing Systems*. pages 121–128.

Andrew McCallum, Xuerui Wang, and Andres Corrada-Emmanuel. 2007. Topic and role discovery in social networks with experiments on enron and academic email. *Journal of Artificial Intelligence Research* .

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*. pages 3111–3119.

Michael J Paul and Mark Dredze. 2015. Sprite: Generalizing topic models with structured priors. *Transactions of the Association for Computational Linguistics* 3:43–57.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods on Natural Language Processing*. volume 14, pages 1532–1543.

Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D Manning. 2009. Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 248–256.

Michal Rosen-Zvi, Thomas Griffiths, Mark Steyvers, and Padhraic Smyth. 2004. The author-topic model for authors and documents. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*. AUAI Press, pages 487–494.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia* 6(12):e26752.

Benjamin M Schmidt. 2012. Words alone: Dismantling topic models in the humanities. *Journal of Digital Humanities* 2(1):49–65.

Hanna M Wallach, Iain Murray, Ruslan Salakhutdinov, and David Mimno. 2009. Evaluation methods for topic models. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, pages 1105–1112.

Li Wan, Leo Zhu, and Rob Fergus. 2012. A hybrid neural network-latent topic model. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics*. volume 12, pages 1287–1294.

Mo Yu, Matthew R Gormley, and Mark Dredze. 2015. Combining word embeddings and feature embeddings for fine-grained relation extraction. In *Proceedings of the 14th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 1374–1379.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

Hao Zhang, Gunhee Kim, and Eric P Xing. 2015. Dynamic topic modeling for monitoring market competition from online text and image data. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pages 1425–1434.

Microblog Conversation Recommendation via Joint Modeling of Topics and Discourse

Xingshan Zeng[1,3], **Jing Li**[2], **Lu Wang**[4],
Nicholas Beauchamp[5,6], **Sarah Shugars**[6], **Kam-Fai Wong**[1,3]
[1]The Chinese University of Hong Kong, [2]Tencent AI Lab
[3]MoE Key Laboratory of High Confidence Software Technologies, China
[4]College of Computer and Information Science, Northeastern University, US
[5]Department of Political Science, Northeastern University, US
[6]Network Science Institute, Northeastern University, US
[1,3]{xszeng,kfwong}@se.cuhk.edu.hk
[2]ameliajli@tencent.com, [4]luwang@ccs.neu.edu
[5]n.beauchamp@northeastern.edu, [6]shugars.s@husky.neu.edu

Abstract

Millions of conversations are generated every day on social media platforms. With limited attention, it is challenging for users to select which discussions they would like to participate in. Here we propose a new method for microblog conversation recommendation. While much prior work has focused on post-level recommendation, we exploit both the conversational context, and user content and behavior preferences. We propose a statistical model that jointly captures: (1) *topics* for representing user interests and conversation content, and (2) *discourse modes* for describing user replying behavior and conversation dynamics. Experimental results on two Twitter datasets demonstrate that our system outperforms methods that only model content without considering discourse.

1 Introduction

Online platforms have revolutionized the way individuals collect and share information (O'Connor et al., 2010; Lee and Ma, 2012; Bakshy et al., 2015), but the vast bulk of online content is irrelevant or unpalatable to any given individual. A user interested in political discussion, for instance, might prefer content concerning a specific candidate or issue, and only then if discussed in a positive light without controversy (Adamic and Glance, 2005; Bakshy et al., 2015).

How do individuals facing such large quantities of superfluous material select which conversations to engage in, and how might we better algorithmically recommend conversations suited to individual users? We approach this problem from a *microblog conversation recommendation* framework. Where prior work has focused on the content of individual posts for recommendation (Chen et al., 2012; Yan et al., 2012; Vosecky et al., 2014; He and Tan, 2015), we examine the entire history and context of a conversation, including both topical content and *discourse modes* such as agreement, question-asking, argument and other dialogue acts (Ritter et al., 2010).[1] And where Backstrom et al. (2013) leveraged conversation reply structure (such as previous user engagement), their model is unable to predict first entry into new conversations, while ours is able to predict both new

Conversation 1
...
[$\mathbf{U}_1$]: The sheer cognitive dissonance required for a "liberal" to say Clinton is as bad as Trump is just staggering.
[U_2]: Hillarists, Troll; they insult Liberals trying to distract from Hillary's Conseratism.
[U_3]: I still prefer Hillarist b/c it describes their Cultish and ideological aspects.
———————
...

Conversation 2
...
[U_4]: I do not like trump at all, but Comey left her in place knowing Bernie is much stronger.
[$\mathbf{U}_1$]: If you're going to actively start rooting against the Democrats, get off my mentions. I have enough GOP doing that.
[U_5]: Your tweets are an example of why open primaries are stupid. You're not a Dem, you're just for one guy.
———————
[$\mathbf{U}_1$]: No offense, but you've been wrong about pretty much everything so far. Why would I trust your prognostication now?
...

Figure 1: Two snippets of conversations on Twitter. [U_i]: The message is posted by user U_i. "—" is the dividing line between training history and test part. U_1 did not reengage in Conversation 1 but reengaged in Conversation 2.

[1]In this paper, *discourse mode* refers to a certain type of dialogue act, e.g., agreement or argument. The *discourse structure* of a conversation means some combination (or a probability distribution) of discourse modes.

Proceedings of NAACL-HLT 2018, pages 375–385
New Orleans, Louisiana, June 1 - 6, 2018.

and repeated entry into conversations based on a combination of topical and discourse features.

To illustrate the interplay between topics and discourse, Figure 1 displays two snippets of conversations on Twitter collected during the 2016 United States presidential election. User U_1 participates in both conversations. The first conversation is centered around Clinton, and U_1, who is more typically involved with conversations about candidate Sanders, does not return. In the second conversation, however, U_1 is involved in a heated back-and-forth debate, and thus is drawn back to a conversation that they may otherwise have abandoned but for their enjoyment of adversarial discourse.

Effective conversation prediction and recommendation requires an understanding of both user interests and discourse behaviors, such as agreement, disagreement, inquiry, backchanneling, and emotional reactions. However, acquiring manual labels for both is a time-consuming process and hard to scale for new datasets. We instead propose a unified statistical learning framework for conversation recommendation, which jointly learns (1) hidden factors that reflect *user interests* based on conversation history, and (2) *topics* and *discourse modes* in ongoing conversations, as discovered by a novel probabilistic latent variable model. Our model is built on the success of collaborative filtering (CF) in recommendation systems, where latent dimensions of product ratings or movie reviews are extracted to better capture user preferences (Linden et al., 2003; Salakhutdinov and Mnih, 2008; Wang and Blei, 2011; McAuley and Leskovec, 2013). To the best of our knowledge, we are the first to model *both* topics and discourse modes as part of a CF framework and apply it to microblog conversation recommendation.[2]

Experimental results on two Twitter conversation datasets show that our proposed model yields significantly better performance than state-of-the-art post-level recommendation systems. For example, by leveraging both topical content and discourse structure, our model achieves a mean average precision (MAP) of 0.76 on conversations about the U.S. presidential election, compared with 0.70 by McAuley and Leskovec (2013), which only considers topics. We further conducted detailed analysis on the latent topics and discourse modes and find that our model can discover reasonable topic and discourse representations, which play an important role in characterizing reply behaviors. Finally, we also provide a pilot study on recommendation for first time replies, which shows that our model outperforms comparable recommendation systems.

The rest of this paper is structured as follows. The related work is discussed in Section 2. We then present our microblog conversation recommendation model in Section 3. The experimental setup and results are described in Sections 4 and 5. Finally, we conclude in Section 6.

2 Related Work

Social media has attracted increasing attention in digital communication research (Agichtein et al., 2008; Kwak et al., 2010; Wu et al., 2011). The problem studied here is closely related to work on recommendation and response prediction in microblogs (Artzi et al., 2012; Hong et al., 2013), where the goal is to predict whether a user will share or reply to a given post. Existing methods focus on measuring features that reflect personalized user interests, including topics (Hong et al., 2013) and network structures (Pan et al., 2013; He and Tan, 2015). These features have been investigated under a learning to rank framework (Duan et al., 2010; Artzi et al., 2012), graph ranking models (Yan et al., 2012; Feng and Wang, 2013; Alawad et al., 2016), and neural network-based representation learning methods (Yu et al., 2016).

Distinguishing from prior work that focuses on post-level recommendation, we tackle the challenges of predicting user reply behaviors at the conversation-level. In addition, our model not only captures latent factors such as the topical interests of users, but also leverages the automatically learned discourse structure. Much of the previous work on discourse structure and dialogue acts has relied on labeled data (Jurafsky et al., 1997; Stolcke et al., 2000), while unsupervised approaches have not been applied to the problem of conversation recommendation (Woszczyna and Waibel, 1994; Crook et al., 2009; Ritter et al., 2010; Joty et al., 2011).

Our work is also in line with conversation modeling for social media discussions (Ritter et al., 2010; Budak and Agrawal, 2013; Louis and Cohen, 2015; Cheng et al., 2017). Topic modeling

[2]To ensure the general applicability of our approach to domains lacking such information, we do not utilize external features such as network structure, but it may certainly be added in future, more narrowly targeted applications.

has been employed to identify conversation content on Twitter (Ritter et al., 2010). In this work, we propose a probabilistic model to capture both topics and discourse modes as latent variables. A further line of work studies the reposting and reply structure of conversations (Gómez et al., 2011; Laniado et al., 2011; Backstrom et al., 2013; Budak and Agrawal, 2013). But none of this work distinguishes the rich discourse functions of replies, which is modeled and exploited in our work.

3 The Joint Model of Topic and Discourse for Recommendation

Our proposed microblog conversation recommendation framework is based on collaborative filtering and a novel probabilistic graphical model. Concretely, our objective function takes the form:

$$\min \mathcal{L} + \mu \cdot NLL(\mathcal{C} \mid \mathbf{\Theta}) \tag{1}$$

This function encodes two types of information. First, $\mathcal{L}$ models user reply preference in a similar fashion to collaborative filtering (CF) (Hu et al., 2008; Pan et al., 2008). It captures topics of interests and discourse structures users are commonly involved (e.g., argumentation), and takes the form of mean square error (MSE) based on user reply history. This part is detailed in Section 3.1.

The second term, $NLL(\mathcal{C} \mid \mathbf{\Theta})$, denotes the negative log-likelihood of a set of conversations $\mathcal{C}$, with $\mathbf{\Theta}$ containing all parameters. A probabilistic model is described in Section 3.2 that shows how the topical content and discourse structures of conversations are captured by these latent variables.

The hyperparameter μ controls the trade-off between the two effects. ℓ_2 regularization is also added for parameters to avoid model overfitting.

For the rest of this section, we first present the construction of $\mathcal{L}$ and $NLL(\mathcal{C} \mid \mathbf{\Theta})$ in Sections 3.1 and 3.2. We then discuss how these two components can be mutually informed by each other in Section 3.3. Finally, the generative process and parameter learning are described in Section 3.4.

3.1 Reply Preference ($\mathcal{L}$)

Our user reply preference modeling is built on the success of collaborative filtering (CF) for product ratings. However, classic CF problems, such as product recommendation, generally rely on explicit user feedback. Unlike user ratings on products, our input lacks explicit feedback from users about negative preferences and non-response. Therefore, we follow one-class Collaborative Filtering (Hu et al., 2008; Pan et al., 2008), which weights positive instances higher during training and is thus suited to our data. Formally, for user u and conversation c, we measure reply preference based on the MSE between predicted preference score $p_{u,c}$ and reply history $r_{u,c}$. $r_{u,c}$ equals 1 if u is in the conversation history; otherwise, it is 0. The first term of objective (Eq. 1) takes the following form:

$$\mathcal{L} = \sum_{u=1}^{|\mathcal{U}|} \sum_{c=1}^{|\mathcal{C}|} f_{u,c} \cdot (p_{u,c} - r_{u,c})^2 \tag{2}$$

where $\mathcal{U}$ consists of users $\{u\}$ and $\mathcal{C}$ is a set of conversations $\{c\}$ in a dataset. $f_{u,c}$ is the corresponding weight for a conversation c and a target user u. Intuitively, it has a large value if positive feedback (user replied) is observed. Therefore, we adapt the formulation from Pan et al. (2008):

$$f_{u,c} = \begin{cases} s & \text{if } r_{u,c} = 1 \text{ (i.e., user replied)} \\ 1 & \text{if } r_{u,c} = 0 \end{cases} \tag{3}$$

where $s > 1$, an integer hyperparameter to be tuned.

Inspired by prior models (Koren et al., 2009; McAuley and Leskovec, 2013), we propose the following latent factor model to describe $p_{u,c}$:

$$p_{u,c} = \lambda \cdot \gamma_u^U \cdot \gamma_c^C + (1-\lambda) \cdot \delta_u^U \cdot \delta_c^C + b_u + b_c + a \tag{4}$$

γ_u^U and γ_c^C are K-dimensional latent vectors that encode topic-specific information (where K is the number of latent topics) for users and conversations. Specifically, γ_u^U reflects the topical interests of u, with higher value $\gamma_{u,k}^U$ indicating greater interest by u in topic k. γ_c^C captures the extents that topics are discussed in conversation c.

Similarly, D-dimensional vectors δ_u^U and δ_c^C capture discourse structures in shaping reply behaviors (where D is the number of discourse clusters). δ_u^U reflects the discourse behaviors u prefers, such as u_1 often enjoys arguments as in the second conversation of Figure 1, while δ_c^C captures the discourse modes used throughout conversation c. By multiplying user and conversation factors, we can measure the corresponding similarity. The predicted score $p_{u,c}$ thereby reflects the tendency for a user u to be involved in conversation c.

As pointed out by McAuley and Leskovec (2013), these latent vectors often encode hidden factors that are hard to interpret under a CF framework. Therefore, in Section 3.2, we present a novel probabilistic model which can extract interpretable topics and discourse modes as word

distributions. We then describe how they can be aligned with the latent vectors of γ^C and δ^U.

Parameter a is an offset parameter, b_u and b_c are user and conversation biases, and $\lambda \in [0, 1]$ serves as the weight for trading offs of topic and discourse factors in reply preference modeling.

3.2 Corpus Likelihood $NLL(\mathcal{C} \mid \mathbf{\Theta})$

Here we present a novel probabilistic model that learns coherent word distributions for latent topics and discourse modes of conversations. Formally, we assume that each conversation $c \in \mathcal{C}$ contains M_c messages, and each message m has $N_{c,m}$ words. We distinguish three latent components – *discourse*, *topic*, and *background* – underlying conversations, each with their own type of word distribution. At the corpus level, there are K topics represented by word distribution ϕ_k^T ($k = 1, 2, ..., K$), while ϕ_d^D ($d = 1, 2, ..., D$) represents the D *discourse modes* embedded in corpus. In addition, we add a *background* word distribution ϕ^B to capture general information (e.g., common words), which do not indicate either discourse or topic information. ϕ_d^D, ϕ_k^T, and ϕ^B are all multinomial word distributions over vocabulary size V. Below describes more details.

Message-level Modeling. Our model assigns two types of message-level multinomial variables to each message: $z_{c,m}$ reflects its latent *topic* and $d_{c,m}$ represents its *discourse mode*.

Topic assignments. Due to the short nature of microblog posts, we assume each message m in conversation c contains only one topic, indexed as $z_{c,m}$. This strategy has been proven useful to alleviate data sparsity for topic inference (Quan et al., 2015). We further assume messages in the same conversation would focus on similar topics. We thus draw topic $z_{c,m} \sim \theta_c$, where θ_c denotes the fractions of topics discussed in conversation c.

Discourse assignments. To capture discourse behaviors of u, distribution π_u is used to represent the discourse modes in messages posted by u. The discourse mode $d_{c,m}$ for message m is then generated from $\pi_{u_{c,m}}$, where $u_{c,m}$ is the author of m in c.

Word-level Modeling. We aim to separate *discourse*, *topic*, and *background* information for conversations. Therefore, for each word $w_{c,m,n}$ of message m, a ternary switcher $x_{c,m,n} \in \{\text{DISC}, \text{TOPIC}, \text{BACK}\}$ controls word $w_{c,m,n}$ to fall into one of the three types: *discourse*, *topic*, and *background*.

Discourse words (DISC) are indicative of the discourse modes of messages. When $x_{c,m,n} = \text{DISC}$ (i.e., $w_{c,m,n}$ is assigned as a discourse word), word $w_{c,m,n}$ is generated from the discourse word distribution $\phi_{d_{c,m}}^D$ where $d_{c,m}$ is discourse assignment to message m.

Topic words (TOPIC) describe the topical focus of a conversation. When $x_{c,m,n} = \text{TOPIC}$, $w_{c,m,n}$ is assigned as a topic word and generated from $\phi_{z_{c,m}}^T$ – word distribution given topic of m.

Background words (BACK) capture the general information that is not related to discourse or topic. When word $w_{c,m,n}$ is assigned as a background word ($x_{c,m,n} = \text{BACK}$), it is drawn from background distribution ϕ^B.

Switching among Topic, Discourse, and Background. We further assume the word type switcher $x_{c,m,n}$ is sampled from a multinomial distribution which depends on the current discourse mode $d_{c,m}$. The intuition is that messages of different discourse modes may show different distributions of the three word types. For instance, a statement message may contain more content words than a rhetorical question. Specifically, $x_{c,m,n} \sim Multi(\tau_{d_{c,m}})$, where τ_d is a 3-dimension stochastic vector that expresses the appearing probabilities of three kinds of words (DISC, TOPIC, BACK), when the discourse assignment is d. Stop words and punctuations are forced to be labeled as discourse or background. By explicitly distinguishing different types of words with switcher $x_{c,m,n}$, we can thus separate word distributions that reflect discourse, topic, and background information.

Likelihood. Based on the message-level and the word-level generation process, the probability of observing words in the given corpus is:

$$\begin{aligned} & Pr(\mathcal{C} \mid \boldsymbol{\theta}, \boldsymbol{\pi}, \boldsymbol{\phi}, \boldsymbol{\tau}, \mathbf{z}, \mathbf{d}, \mathbf{x}) \\ = & \prod_{c=1}^{C} \prod_{m=1}^{M_c} \theta_{c,z_{c,m}} \pi_{u_{c,m},d_{c,m}} \\ & \times \prod_{x_{c,m,n}=\text{BACK}} \tau_{d_{c,m},\text{BACK}} \phi_{w_{c,m,n}}^B \\ & \times \prod_{x_{c,m,n}=\text{DISC}} \tau_{d_{c,m},\text{DISC}} \phi_{d_{c,m},w_{c,m,n}}^D \\ & \times \prod_{x_{c,m,n}=\text{TOPIC}} \tau_{d_{c,m},\text{TOPIC}} \phi_{z_{c,m},w_{c,m,n}}^T \end{aligned} \tag{5}$$

And we use negative log likelihood to model corpus likelihood effect in Eq. 1, i.e., $NLL(\mathcal{C} \mid \mathbf{\Theta}) =$

$-\log(Pr(\mathcal{C} \mid \mathbf{\Theta})$, where parameters set $\mathbf{\Theta} = \{\boldsymbol{\theta}, \boldsymbol{\pi}, \boldsymbol{\phi}, \boldsymbol{\tau}, \mathbf{z}, \mathbf{d}, \mathbf{x}\}$.

3.3 Mutually Informed User Preference and Latent Variables

As mentioned above, the hidden factors discovered in Section 3.1 lack interpretability, which can be boosted by the learned latent topics and discourse modes in Section 3.2. However, it is non-trivial to link the topic-related parameters of γ_c^C to the conversation topic distributions of θ_c, since the former takes real values from $-\infty$ to $+\infty$ while the latter is a stochastic vector. Therefore, we follow the strategy from McAuley and Leskovec (2013) to apply a softmax function over γ_c^C:

$$\theta_{c,k} = \frac{exp(\kappa^T \gamma_{c,k}^C)}{\sum_{k'=1}^{K} exp(\kappa^T \gamma_{c,k'}^C)} \tag{6}$$

We further assume that the discourse mode preference by users, δ_u^U, can also be informed by the discourse mode distribution captured by π_u, i.e., a user who enjoys arguments may be willing to participate another. So similarly, we define:

$$\pi_{u,d} = \frac{exp(\kappa^D \delta_{u,d}^U)}{\sum_{d'=1}^{D} exp(\kappa^D \delta_{u,d'}^U)} \tag{7}$$

where κ^T and κ^D are learnable parameters that control the "peakiness" of the transformation. For example, a larger κ^T indicates a more focused conversation, while a smaller κ^T means users discuss diverse topics.

Finally, softmax transformation is also applied to ϕ_k^T, ϕ_d^D, ϕ^B, and τ_d, as done in McAuley and Leskovec (2013), with additional parameters ψ_k^T, ψ_d^D, ψ^B, and χ_d (as shown in Figure 2). This is to ensure that the distributions ϕ_*^* and τ_d are stochastic vectors. In doing so, these distributions can be learned via optimizing ψ_*^* and χ_d, which take any value and thus ensure that the cost function in Eq. 1 is optimized without considering any parameter constraints.

3.4 Generative Process and Model Learning

Our word generation process is displayed in Figure 2 and described as follows:

- Compute topic distribution θ_c by Eq. 6
- For message $m = 1$ to M_c:
  - Compute discourse distribution $\pi_{u_{c,m}}$ by Eq. 7
  - Draw topic assignment $z_{c,m} \sim Multi(\theta_c)$
  - Draw discourse mode $d_{c,m} \sim Multi(\pi_{u_{c,m}})$
  - For word index $n = 1$ to $N_{c,m}$:
    - Draw word type $x_{c,m,n} \sim Multi(\tau_d)$

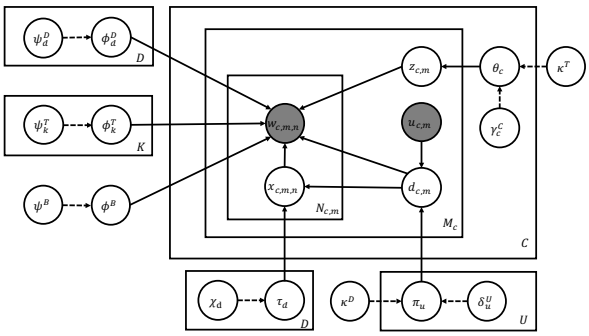


Figure 2: Generative process of our joint model of topic and discourse. u represents users. c represents conversations. Dotted arrows represent the softmax linkings, while solid arrows mean conditional priors.

    - if $x_{c,m,n} == BACK$:
      Draw word $w_{c,m,n} \sim Multi(\phi^B)$
    - if $x_{c,m,n} == DISC$:
      Draw word $w_{c,m,n} \sim Multi(\phi_{d_{c,m}}^D)$
    - if $x_{c,m,n} == TOPIC$:
      Draw word $w_{c,m,n} \sim Multi(\phi_{z_{c,m}}^T)$

Parameter Learning. For learning, we randomly initialize all learnable parameters and then alternate between the following two steps:

Step 1. Fix topic and discourse assignments z and d, and word type switcher x, then optimize the remaining parameters in Eq. 1 by L-BFGS (Nocedal, 1980):

$$\begin{aligned} &\text{Update } a, b, \gamma^*, \delta^*, \kappa^*, \psi^*, \chi = \\ &\qquad argmin\, \mathcal{L} + \mu \cdot NLL(\mathcal{C} \mid \mathbf{\Theta}) \end{aligned} \tag{8}$$

Step 2. Sample topic and discourse assignments z and d at the message level and word type switcher x at the word level, using the distributions, computed according to parameters optimized in step 1:

$$\begin{aligned} &\text{Sample } z_{c,m},\ d_{c,m},\ x_{c,m,n} \text{ with probabilities} \\ &\quad p(z_{c,m} = k) = \theta_{c,k} \\ &\quad p(d_{c,m} = d) = \pi_{u_{c,m},d} \\ &\quad p(x_{c,m,n} = BACK) = \phi_{w_{c,m,n}}^B \tau_{d_{c,m},BACK} \\ &\quad p(x_{c,m,n} = DISC) = \phi_{d_{c,m},w_{c,m,n}}^D \tau_{d_{c,m},DISC} \\ &\quad p(x_{c,m,n} = TOPIC) = \phi_{z_{c,m},w_{c,m,n}}^T \tau_{d_{c,m},TOPIC} \end{aligned} \tag{9}$$

Step 2 is analogous to Gibbs Sampling (Griffiths, 2002) in probabilistic graphical models, such as LDA (Blei et al., 2003). However, distinguishing from previous models, the multinomial distributions in our models are not drawn from a Dirichlet prior. Instead, they are computed based on the parameters learned in Step 1.

Our learning process stops when the change of parameters is small (i.e., below a pre-specified

Dataset	# of user	# of conv	# of msg	Avg msg per user	Avg conv per user
US Election	4,300	2,013	22,092	5.14	1.23
TREC	10,122	7,500	38,999	3.85	1.71

Table 1: Statistics of two datasets.

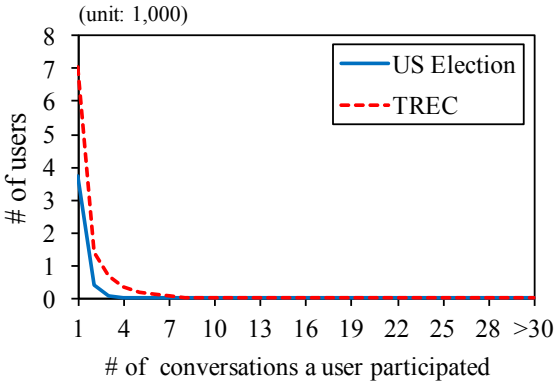


Figure 3: Horizontal axis: number of conversations that a user is involved. Vertical axis: number of users fall in the category (unit: 1,000). Notice that most of users (about 98%) participate in less than 10 conversations.

threshold). Multiple restarts are tried, and similar results are achieved.

4 Experimental Setup

Datasets. We collected two microblog conversation datasets from Twitter for experiments[3]: one contains discussions about the U.S. presidential election (henceforth **US Election**), the other gathers conversations of diverse topics based on the tweets released by TREC 2011 microblog track (henceforth **TREC**)[4]. US Election was collected from January to June of 2016 using Twitter's Streaming API[5] with a small set of political keywords.[6] To recover conversations, Tweet Search API[7] was used to retrieve messages with the "in-reply-to" relations to collect tweets in a recursive way until full conversations were recovered.

Statistics of the datasets are shown in Table 1. Figure 3 displays the number of conversations individual users participated in. As can be seen, most users are involved in only a few conversations. Simply leveraging personal chat history will not produce good performance for conversation recommendation.

In our experiments, we predict whether a user will engage in a conversation given the previous messages in that conversation and past conversations the user is involved. For model training and testing, we divide conversations into three ordered segments, corresponding to training, development, and test sets at 75%, 12.5%, and 12.5%.[8]

Preprocessing and Hyperparameter Tuning. For preprocessing, links, mentions (i.e., @username), and hashtags in tweets were replaced with generic tags of "URL", "MENTION", and "HASHTAG". We then utilized the Twitter NLP tool[9] (Gimpel et al., 2011; Owoputi et al., 2013) for tokenization and non-alphabetic token removal. We removed stop words and punctuations for all comparisons to ensure comparable performance. We maintain a vocabulary with the 5,000 most frequent words.

Our model parameters are tuned on the development set based on grid search, i.e. the parameters that give the lowest value for our objective are selected. Specifically, the number of discourse modes (D) and topics (K) are tuned to be 10. The trade-off parameter μ between user preference and corpus negative log-likelihood takes value of 0.1, and λ, the parameter for balancing topic and discourse, is set to 0.5. Finally, the confidence parameter s takes a value of 200 to give higher weight for positive instances, i.e., a user replied to a conversation.

Evaluation Metrics. Following prior work on social media post recommendation (Chen et al., 2012; Yan et al., 2012), we treat our task on conversation recommendation as a ranking problem. Therefore, popular information retrieval evaluation metrics, including precision at K (P@K), mean average precision (MAP) (Manning et al., 2008), and normalized Discounted Cumulative Gain at K (nDCG@K) (Järvelin and Kekäläinen, 2002) are reported. The metrics are computed per user in the dataset and then averaged over all users. The values range from 0.0 to 1.0, with higher values indicating better performance.

Baselines and Comparisons. For comparison, we first consider three baselines: 1) ranking

[3]The datasets are available at http://www.ccs.neu.edu/home/luwang/

[4] http://trec.nist.gov/data/tweets/

[5]https://developer.twitter.com/en/docs/tweets/filter-realtime/api-reference/post-statuses-filter.html

[6]Keyword list: "trump", "hillary", "clinton", "president", "politics", and "election."

[7]https://developer.twitter.com/en/docs/tweets/search/api-reference/get-saved_searches-show-id

[8]At least one turn per conversation is retained for training. It is possible that one user only replies in either development set or test set, but it is rather infrequent.

[9]http://www.cs.cmu.edu/~ark/TweetNLP/

Models	US Election			TREC		
	MAP	P@1	nDCG@5	MAP	P@1	nDCG@5
Baselines						
RANDOM	0.018	0.004	0.009	0.006	0.001	0.002
LENGTH	0.025	0.002	0.003	0.013	0.002	0.004
POPULARITY	0.050	0.010	0.025	0.023	0.005	0.010
Comparisons						
OCCF	0.637	0.589	0.649	0.410	0.385	0.425
RSVM	0.687	0.680	0.690	0.554	0.575	0.559
CTR	0.673	0.649	0.678	0.475	0.431	0.495
ADAPTED HFT	0.698	0.652	0.706	0.487	0.447	0.504
Our model	**0.762**	**0.750**	**0.757**	**0.591**	**0.591**	**0.600**

Table 2: Conversation recommendation results on US Election and TREC. The best result for each column is highlighted in **bold**. Our model performs significantly better than all the comparisons ($p < 0.01$, paired t-test).

conversations randomly (RANDOM); 2) longer conversations (i.e., more words) ranked higher (LENGTH); 3) conversations with more distinct users ranked higher (POPULARITY).

We further compare results with three established recommendation models:

• OCCF: one-class Collaborative Filtering (Pan et al., 2008), which only considers users' reply history without modeling content in conversations.

• RSVM: ranking SVM (Joachims, 2002), which ranks conversations for each user with the content and Twitter features as in Duan et al. (2010).

• CTR: messages in one conversation are aggregated into one post and a state-of-the art Collaborative Filtering-based post recommendation model is applied (Chen et al., 2012).

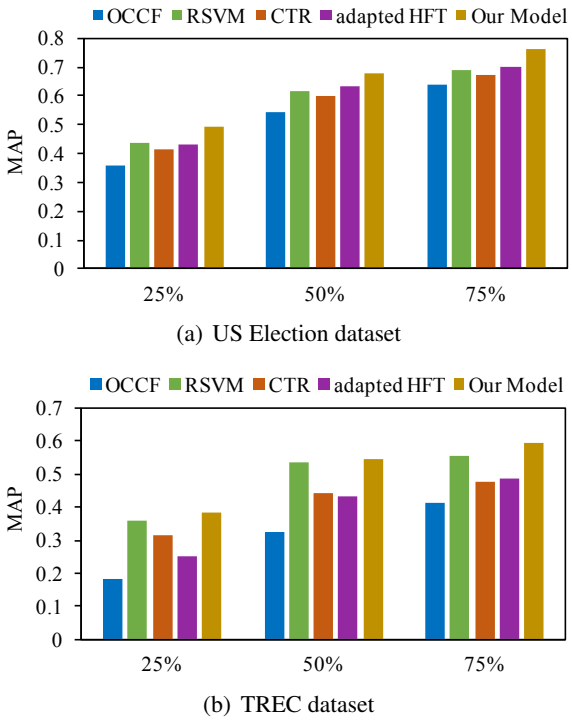
Finally, we also adapt the "hidden factors as topics" (HFT) model proposed in McAuley and Leskovec (2013) (henceforth ADAPTED HFT). Because the original model leverages the ratings for all product reviews and does not handle implicit user feedback well, we replace their user preference objective function with ours (Eq. 2).

5 Experimental Results

In this section, we first discuss our main evaluation in Section 5.1. A case study and corresponding discussion are provided in Section 5.2 to provide further insights, which is followed by an analysis of the topics and discourse modes discovered by our model (Section 5.3). We also examine our performance on first time replies (Section 5.4).

5.1 Conversation Recommendation Results

Experimental results are displayed in Table 2, where our model yields statistically significantly better results than baselines and comparisons



(a) US Election dataset

(b) TREC dataset

Figure 4: MAP scores for models trained on 25%, 50%, and 75% of conversation history. For each quantile, from left to right shows the result of OCCF, RSVM, CTR, ADAPTED HFT, and our model. In general, longer conversation history leads to better performance, and our model outperforms compared systems in all settings.

(paired t-tests, $p < 0.01$). For P@K, we only report P@1, because a significant amount of users participate only in 1 or 2 conversations. For nDCG@K, different K values are experimented, which results in similar trend, so only nDCG@5 is reported.

We find that the baselines that rank conversations with simple features (e.g., length or popularity) perform poorly. This implies that generic algorithms that do not consider conversation content or user preference cannot produce reasonable recommendations.

Although some non-baseline systems capture content in one way or another, only ADAPTED HFT and our model exploit latent topic models to better represent content in tweets, and outperform other methods.

Compared to ADAPTED HFT, which only considers latent topics under a collaborative filtering framework, our model extracts both topics and discourse modes as latent variables, and shows superior performance on both datasets. Our discourse variables go beyond topical content to capture social behaviors that affect user engagement, such as

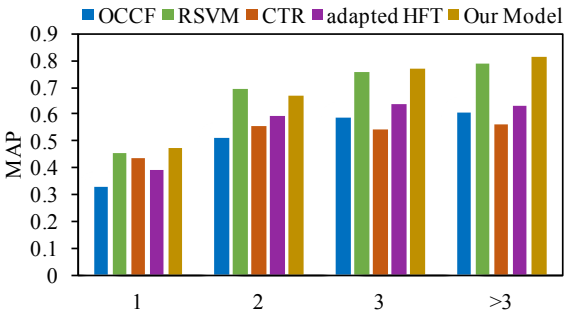


Figure 5: MAP scores of models for users involved in varying number of conversations on TREC dataset. Horizontal axis: degree of data sparsity indicated by the number of conversations a user involved in training data. Vertical axis: MAP scores. For each degree level, from left to right shows the results of OCCF, RSVM, CTR, ADPATED HFT, and our model.

arguments, question-asking, agreement, and other discourse modes.

Training with Varying Conversation History. To test the model performance based different levels of user engagement history, we further experiment with varying the length of conversations for training. Specifically, in addition to using 75% of conversation history, we also extract the first 25% and 50% of history as training. The rest of a conversation is separated equally for development and test. Figure 4 shows the MAP scores for US Election and TREC datasets. The increasing MAP for all methods as the training history increases indicates that generally, conversation history is essential for recommendation. Our model performs consistently better over different lengths of conversation histories.

Results for Varying Degree of Data Sparsity. From Table 1 and Figure 3, we observe that most users in our datasets are involved in only a few conversations. In order to study the effects of data sparsity on recommendation models, we examine in Figure 5 the MAP scores for users engaged in a varying number of conversations, as measured on the TREC dataset. The results on the US Election dataset have similar distributions. As we see, the prediction results become worse for users involved in fewer conversations. This indicates that data sparsity serves as a challenge for all recommendation models. We also observe that our model performs consistently better than other models over different degrees of sparsity. This implies that effectively capturing discourse structure in conversation context is useful to mitigating the effects of data sparsity on conversation recommendation.

Models	*Conv 1 (c_1)*	*Conv 2 (c_2)*
OCCF	0.941	0.922
ADAPTED HFT	0.923	0.954
Our model	0.924	0.961

Table 3: Predicted recommendation scores by different models of U_1 for conversations c_1 and c_2 in Figure 1. U_1 later replies to c_2 but not c_1, where our model predicts scores of 0.961 for c_2 (higher than 0.924 for c_1).

Latent Dim.	*User U_1*	*Conv 1 (c_1)*	*Conv 2 (c_2)*
Topic 1 (Sanders)	0.92 ($\gamma^U_{u_1,1}$)	0.10 ($\gamma^C_{c_1,1}$)	0.63 ($\gamma^C_{c_2,1}$)
Topic 2 (Clinton)	0.14 ($\gamma^U_{u_1,2}$)	0.84 ($\gamma^C_{c_1,2}$)	0.12 ($\gamma^C_{c_2,2}$)
Disc 1 (argument)	0.46 ($\delta^U_{u_1,1}$)	0.28 ($\delta^C_{c_1,1}$)	0.38 ($\delta^C_{c_2,1}$)
Disc 2 (statement)	-0.24 ($\delta^U_{u_1,2}$)	0.98 ($\delta^C_{c_1,2}$)	-0.09 ($\delta^C_{c_2,2}$)

Table 4: Sample latent dimensions of topics ($\gamma^U_{u_1}$ for user, and $\gamma^C_{c_*}$ for conversations) and discourse modes ($\delta^U_{u_1}$ for user, and $\delta^C_{c_*}$ for conversations). User U_1 shows interest in topic 1 (about Sanders), which is also a dominating topic in conversation c_2, but is not interested in topic 2 (about Clinton). U_1 shows a preference for discourse mode 1 (argument) over mode 2 (statement).

5.2 Case Study and Discussion

Here we present a case study based on the sample conversations in Figure 1. Recall that user U_1 is interested in conversations about Sanders, and also prefers more argumentative discourse, and thus returns in conversation c_2 but not c_1.

Table 3 shows the predicted scores for the two conversations from OCCF, ADAPTED HFT, and our model (as in Eq. 2). Both ADAPTED HFT and our model more accurately recommend c_2 over c_1, with our model producing a slightly higher recommendation score for c_2.

Table 4 shows the latent dimension values for the learned topics and discourse modes for this user and these two conversations. Based on human inspection, topic 1 appears to contain words about Sanders, which is the main topic in conversation c_2. Topic 2 is about Clinton, which is a dominating topic in conversation c_1. Our model also picks up user interest in topic 1 (Sanders), and thus assigns $\gamma^U_{u_1,1}$ a high value. For discourse modes, our model also generates a high score for "argument" discourse (labeled via human inspection) for both the user and c_2.

5.3 Further Analysis of Topic and Discourse

Ablation Study. We have shown that joint modeling of topical content and discourse modes produces the superior performance for our model.

Here we provide an ablation study to examine the relative contributions of those two aspects by setting the trade-off parameter λ to 1.0 (topic only) or 0.0 (discourse only). Table 5 shows that topics or discourse individually improve slightly upon the comparison ADAPTED HFT, but only jointly do they improve significantly upon it.

Models	US Election	TREC
ADAPTED HFT	0.698	0.487
Our model (topic only)	0.711	0.491
Our model (discourse only)	0.705	0.483
Our model (full)	**0.762**	**0.591**

Table 5: MAP of different variants of our model. Best results in each column is in **bold**.

Topic Coherence. To examine the quality of topics found by our model, we use the C_V topic coherence score measured via the open-source toolkit Palmetto[10], which has been shown to produce evaluation performance comparable to human judgment (Röder et al., 2015). Our model achieves topic coherence scores of 0.343 and 0.376 on TREC and US Election datasets, compared to 0.338 and 0.371 for the topics from ADAPTED HFT.

Discourse	Top 10 Terms	
	US Election	TREC
Question	? it if ... so all how because when any	? : HASHTAG or too with MENTION and ... what
Reaction	you like any good ! please no ~ lol what	all EMOTICON & !!! right ok u ;) thank haha
Statement	's do think the . should they from and have	i , a what all you be how then ...
Argument	but that all fuck without against out though ! anything	do would up that too even always never anything much
Reference	be i about that MENTION it " you -lrb- ?	MENTION ... ! : what rt it you URL :)

Table 6: Top 10 representative terms for sample discourse modes discovered by our model in two datasets. Names of discourse modes are our interpretations according to the word distributions generated by our model.

Sample Discourse Modes. While our topic word distributions are relatively unsurprising, of greater interest are the discourse mode word distributions. Table 6 shows a sample of discourse modes as labeled by human. Although this is merely a qualitative human judgment at this point, there does appear to be a notable overlap in discourse modes between the two datasets even though they were learned separately.

[10] https://github.com/AKSW/Palmetto/

5.4 First Time Reply Results

From a recommendation perspective, users may be interested in joining new conversations. We thus compare each recommendation system for first time replies. For each user, we only evaluate for conversations where they are newcomers. Table 7 shows that, unsurprisingly, all systems perform poorly on this task, though our model performs slightly better. This suggests that other features, e.g., network structures or other discussion thread features, could usefully be included in future studies that target new conversations.

Models	US Election	TREC
OCCF	0.035	0.033
RSVM	0.023	0.002
CTR	0.029	0.016
ADAPTED HFT	0.054	0.058
Our model	**0.083**	**0.090**

Table 7: MAP of models considering only first time replies. Best results in each column is in **bold**.

6 Conclusion

This paper has presented a framework for microblog conversation recommendation via jointly modeling topics and discourse modes. Experimental results show that our method can outperform competitive approaches that omit user discourse behaviors. Qualitative analysis shows that our joint model yields meaningful topics and discourse representations.

Acknowledgements

This work is partly supported by Innovation and Technology Fund (ITF) Project No. 6904333, General Research Fund (GRF) Project No. 14232816 (12183516), and National Science Foundation Grant IIS-1566382. We thank Shuming Shi, Yan Song, and the three anonymous reviewers for the insightful suggestions on various aspects of this work.

References

Lada A. Adamic and Natalie Glance. 2005. The political blogosphere and the 2004 U.S. election: Divided they blog. In *Proceedings of the 3rd International Workshop on Link Discovery*. ACM, LinkKDD '05, pages 36–43.

Eugene Agichtein, Carlos Castillo, Debora Donato, Aristides Gionis, and Gilad Mishne. 2008. Finding high-quality content in social media. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*. ACM, pages 183–194.

Noor Aldeen Alawad, Aris Anagnostopoulos, Stefano Leonardi, Ida Mele, and Fabrizio Silvestri. 2016. Network-aware recommendations of novel tweets. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, pages 913–916.

Yoav Artzi, Patrick Pantel, and Michael Gamon. 2012. Predicting responses to microblog posts. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 602–606.

Lars Backstrom, Jon Kleinberg, Lillian Lee, and Cristian Danescu-Niculescu-Mizil. 2013. Characterizing and curating conversation threads: expansion, focus, volume, re-entry. In *Proceedings of the sixth ACM International Conference on Web Search and Data Mining*. ACM, pages 13–22.

Eytan Bakshy, Solomon Messing, and Lada A Adamic. 2015. Exposure to ideologically diverse news and opinion on Facebook. *Science* 348(6239):1130–1132.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent Dirichlet Allocation. *Journal of machine Learning research* 3(Jan):993–1022.

Ceren Budak and Rakesh Agrawal. 2013. On participation in group chats on Twitter. In *Proceedings of the 22nd International Conference on World Wide Web*. ACM, pages 165–176.

Kailong Chen, Tianqi Chen, Guoqing Zheng, Ou Jin, Enpeng Yao, and Yong Yu. 2012. Collaborative personalized tweet recommendation. In *Proceedings of the 35th international ACM SIGIR Conference on Research and development in information retrieval*. ACM, pages 661–670.

Justin Cheng, Michael Bernstein, Cristian Danescu-Niculescu-Mizil, and Jure Leskovec. 2017. Anyone can become a troll: Causes of trolling behavior in online discussions. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*. ACM, CSCW '17, pages 1217–1230.

Nigel Crook, Ramón Granell, and Stephen G. Pulman. 2009. Unsupervised classification of dialogue acts using a Dirichlet process mixture model. In *Proceedings of The 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue, SIGDIAL 2009*. pages 341–348.

Yajuan Duan, Long Jiang, Tao Qin, Ming Zhou, and Heung-Yeung Shum. 2010. An empirical study on learning to rank of tweets. In *Proceedings of the 23rd International Conference on Computational Linguistics*. Association for Computational Linguistics, pages 295–303.

Wei Feng and Jianyong Wang. 2013. Retweet or not?: personalized tweet re-ranking. In *Proceedings of the sixth ACM International Conference on Web Search and Data Mining*. ACM, pages 577–586.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for Twitter: Annotation, features, and experiments. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference*. pages 42–47.

Vicenç Gómez, Hilbert J Kappen, and Andreas Kaltenbrunner. 2011. Modeling the structure and evolution of discussion cascades. In *Proceedings of the 22nd ACM Conference on Hypertext and hypermedia*. ACM, pages 181–190.

Tom Griffiths. 2002. Gibbs sampling in the generative model of Latent Dirichlet Allocation .

Yue He and Jinxiu Tan. 2015. Study on sina microblog personalized recommendation based on semantic network. *Expert Systems with Applications* 42(10):4797–4804.

Liangjie Hong, Aziz S Doumith, and Brian D Davison. 2013. Co-factorization machines: modeling user interests and predicting individual decisions in Twitter. In *Proceedings of the sixth ACM International Conference on Web Search and Data Mining*. ACM, pages 557–566.

Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE, pages 263–272.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.* 20(4):422–446.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. pages 133–142.

Shafiq R. Joty, Giuseppe Carenini, and Chin-Yew Lin. 2011. Unsupervised modeling of dialog acts in asynchronous conversations. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI 2011*. pages 1807–1813.

D. Jurafsky, E. Shriberg, and D. Biasca. 1997. Switchboard SWBD-DAMSL shallow-discourse-function annotation coders manual. Technical Report Draft 13, University of Colorado, Institute of Cognitive Science.

Yehuda Koren, Robert M. Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *IEEE Computer* 42(8):30–37.

Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. 2010. What is Twitter, a social network or a news media? In *Proceedings of the 19th International Conference on World wide Web*. ACM, pages 591–600.

David Laniado, Riccardo Tasso, Yana Volkovich, and Andreas Kaltenbrunner. 2011. When the wikipedians talk: Network and tree structure of wikipedia discussion pages. In *ICWSM*.

Chei Sian Lee and Long Ma. 2012. News sharing in social media: The effect of gratifications and prior experience. *Computers in Human Behavior* 28(2):331–339.

Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing* 7(1):76–80.

Annie Louis and Shay B. Cohen. 2015. Conversation trees: A grammar model for topic structure in forums. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1543–1553.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to information retrieval*. Cambridge University Press.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems*. ACM, pages 165–172.

Jorge Nocedal. 1980. Updating quasi-newton matrices with limited storage. *Mathematics of computation* 35(151):773–782.

Brendan O'Connor, Ramnath Balasubramanyan, Bryan R Routledge, and Noah A Smith. 2010. From tweets to polls: Linking text sentiment to public opinion time series. *ICWSM* 11(122-129):1–2.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics*. pages 380–390.

Rong Pan, Yunhong Zhou, Bin Cao, Nathan Nan Liu, Rajan M. Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In *Proceedings of the 8th IEEE International Conference on Data Mining*. pages 502–511.

Ye Pan, Feng Cong, Kailong Chen, and Yong Yu. 2013. Diffusion-aware personalized social update recommendation. In *Proceedings of the 7th ACM Conference on Recommender Systems*. ACM, pages 69–76.

Xiaojun Quan, Chunyu Kit, Yong Ge, and Sinno Jialin Pan. 2015. Short and sparse text topic modeling via self-aggregation. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*. pages 2270–2276.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of Twitter conversations. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics*. pages 172–180.

Michael Röder, Andreas Both, and Alexander Hinneburg. 2015. Exploring the space of topic coherence measures. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. pages 399–408.

Ruslan Salakhutdinov and Andriy Mnih. 2008. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th International Conference on Machine learning*. ACM, pages 880–887.

Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics* 26(3):339–373.

Jan Vosecky, Kenneth Wai-Ting Leung, and Wilfred Ng. 2014. Collaborative personalized Twitter search with topic-language models. In *Proceedings of the 37th international ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, pages 53–62.

Chong Wang and David M Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pages 448–456.

M Woszczyna and A Waibel. 1994. Inferring linguistic structure in spoken language. In *Proceedings of ICSLP*. IC-SLP.

Shaomei Wu, Jake M Hofman, Winter A Mason, and Duncan J Watts. 2011. Who says what to whom on Twitter. In *Proceedings of the 20th International Conference on World Wide Web*. ACM, pages 705–714.

Rui Yan, Mirella Lapata, and Xiaoming Li. 2012. Tweet recommendation with graph co-ranking. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics, pages 516–525.

Yang Yu, Xiaojun Wan, and Xinjie Zhou. 2016. User embedding for scholarly microblog recommendation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. volume 2, pages 449–453.

Before Name-calling: Dynamics and Triggers of Ad Hominem Fallacies in Web Argumentation

Ivan Habernal[†] **Henning Wachsmuth**[‡] **Iryna Gurevych**[†] **Benno Stein**[‡]

[†] Ubiquitous Knowledge Processing Lab (UKP) and Research Training Group AIPHES
Department of Computer Science, Technische Universität Darmstadt, Germany
www.ukp.tu-darmstadt.de www.aiphes.tu-darmstadt.de

[‡] Faculty of Media, Bauhaus-Universität Weimar, Germany
<firstname>.<lastname>@uni-weimar.de

Abstract

Arguing without committing a fallacy is one of the main requirements of an ideal debate. But even when debating rules are strictly enforced and fallacious arguments punished, arguers often lapse into attacking the opponent by an *ad hominem* argument. As existing research lacks solid empirical investigation of the typology of ad hominem arguments as well as their potential causes, this paper fills this gap by (1) performing several large-scale annotation studies, (2) experimenting with various neural architectures and validating our working hypotheses, such as controversy or reasonableness, and (3) providing linguistic insights into triggers of ad hominem using explainable neural network architectures.

1 Introduction

Human reasoning is lazy and biased but it perfectly serves its purpose in the argumentative context (Mercier and Sperber, 2017). When challenged by genuine back-and-forth argumentation, humans do better in both generating and evaluating arguments (Mercier and Sperber, 2011). The dialogical perspective on argumentation has been reflected in argumentation theory prominently by the pragma-dialectic model of argumentation (van Eemeren and Grootendorst, 1992). Not only sketches this theory an ideal normative model of argumentation but also distinguishes the wrong argumentative moves, *fallacies* (van Eemeren and Grootendorst, 1987). Among the plethora of prototypical fallacies, notwithstanding the controversy of most taxonomies (Boudry et al., 2015), *ad hominem* argument is perhaps the most famous one. Arguing *against the person* is considered faulty, yet is prevalent in online and offline discourse.[1]

Although the ad hominem fallacy has been known since Aristotle, surprisingly there are very few empirical works investigating its properties. While Sahlane (2012) analyzed ad hominem and other fallacies in several hundred newspaper editorials, others usually only rely on few examples, as observed by de Wijze (2002). As Macagno (2013) concludes, ad hominem arguments should be considered as multifaceted and complex strategies, involving not a simple argument, but several combined tactics. However, such research, to the best of our knowledge, does not exist. Very little is known not only about the feasibility of ad hominem theories in practical applications (the NLP perspective) but also about the dynamics and triggers of ad hominem (the theoretical counterpart).

This paper investigates the research gap at three levels of increasing discourse complexity: ad hominem in isolation, direct ad hominem without dialogical exchange, and ad hominem in large inter-personal discourse context. We asked the following research questions. First, what qualitative and quantative properties do ad hominem arguments have in Web debates and how does that reflect the common theoretical view (RQ1)? Second, how much of the debate context do we need for recognizing ad hominem by humans and machine learning systems (RQ2)? And finally, what are the actual triggers of ad hominem arguments and can we predict whether the discussion is going to end up with one (RQ3)?

We tackle these questions by leveraging Web-based argumentation data (*Change my View* on Reddit), performing several large-scale annotation studies, and creating a new dataset. We experiment with various neural architectures and ex-

[1] According to 'Godwin's law' known from the internet pop-culture (https://en.wikipedia.org/wiki/Godwin's_law), if a discussion goes on long enough, sooner or later someone will compare someone or something to Adolf Hitler.

Proceedings of NAACL-HLT 2018, pages 386–396
New Orleans, Louisiana, June 1 - 6, 2018.

trapolate the trained models to validate our working hypotheses. Furthermore, we propose a list of potential linguistic and rhetorical triggers of ad hominem based on interpreting parameters of trained neural models.[2] This article thus presents the first NLP work on multi-faceted ad hominem fallacies in genuine dialogical argumentation. We also release the data and the source code to the research community.[3]

2 Theoretical background and related work

The prevalent view on argumentation emphasizes its pragmatic goals, such as persuasion and group-based deliberation (van Eemeren et al., 2014), although numerous works have dealt with argument as product, that is, treating a single argument and its properties in isolation (Toulmin, 1958; Habernal and Gurevych, 2017). Yet the social role of argumentation and its alleged responsibility for the very skill of human reasoning explained from the evolutionary perspective (Mercier and Sperber, 2017) provide convincing reasons to treat argumentation as an inherently dialogical tool.

The observation that some arguments are in fact 'deceptions in disguise' was made already by Aristotle (Aristotle and Kennedy (translator), 1991), for which the term *fallacy* has been adopted. Leaving the controversial typology of fallacies aside (Hamblin, 1970; van Eemeren and Grootendorst, 1987; Boudry et al., 2015), the *ad hominem* argument is addressed in most theories. Ad hominem argumentation relies on the strategy of attacking the opponent and some feature of the opponent's character instead of the counter-arguments (Tindale, 2007). With few exceptions, the following five sub-types of ad hominem are prevalent in the literature: **abusive ad hominem** (a pure attack on the character of the opponent), **tu quoque ad hominem** (essentially analogous to the "He did it first" defense of a three-year-old in a sandbox), **circumstantial ad hominem** (the "practice what you preach" attack and accusation of hypocrisy), **bias ad hominem** (the attacked opponent has a hidden agenda), and **guilt by association** (associating the opponent with somebody with a low credibility) (Schiappa and Nordin, 2013; Macagno, 2013; Walton, 2007; Hansen, 2017; Woods, 2008). We omit examples here as these provided in theoretical works or textbooks are usually artificial, as already criticized by (de Wijze, 2002) or (Boudry et al., 2015).

The topic of fallacies, which might be considered as sub-topic of argumentation quality, has recently been investigated also in the NLP field. Existing works are, however, limited to the monological view (Wachsmuth et al., 2017; Habernal and Gurevych, 2016b,a; Stab and Gurevych, 2017) or they focus primarily on learning fallacy recognition by humans (Habernal et al., 2017, 2018a). Another related NLP sub-field includes abusive language and personal attacks in general. Wulczyn et al. (2017) investigated whether or not Wikipedia talk page comments are personal attacks and annotated 38k instances resulting in a highly skewed distribution (only 0.9% were actual attacks). Regarding the participants' perspective, Jain et al. (2014) examined principal roles in 80 discussions from the *Wikipedia: Article for Deletion* pages (focusing on stubbornness or ignoredness, among others) and found several typical roles, including 'rebels', 'voices', or 'idiots'. In contrast to our data under investigation (Change My View debates), Wikipedia talk pages do not adhere to strict argumentation rules with manual moderation and have a different pragmatic purpose.

Reddit as a source platform has also been used in other relevant works. Saleem et al. (2016) detected hateful speech on Reddit by exploiting particular sub-communities to automatically obtain training data. Wang et al. (2016) experimented with an unsupervised neural model to cluster social roles on sub-reddits dedicated to computer games. Zhang et al. (2017) proposed a set of nine comment-level dialogue act categories and annotated 9k threads with 100k comments and built a CRF classifier for dialogue act labeling. Unlike these works which were not related to argumentation, Tan et al. (2016) examined persuasion strategies on Change My View using word overlap features. In contrast to our work, they focused solely on the successful strategies with delta-awarded posts. Using the same dataset, Musi (2017) recently studied concession in argumentation.

3 Data

Change My View (CMV) is an online 'place to post an opinion you accept [...] in an effort to un-

[2] An attempt to address the plea for thinking about problems, cognitive science, and the details of human language (Manning, 2015).

[3] https://github.com/UKPLab/naacl2018-before-name-calling-habernal-et-al

derstand other perspectives on the issue', in other words an online platform for 'good-faith' argumentation hosted on Reddit.[4] A user posts a **submission** (also called **original post(er); OP**) and other participants provide arguments to change the OP's view, forming a typical tree-form Web discussion. A special feature of CMV is that the OP acknowledges convincing arguments by giving a **delta** point (Δ). Unlike the vast majority of internet discussion forums, CMV enforces obeying strict rules (such as no 'low effort' posts, or accusing of being unwilling to change view) whose violation results into deleting the comment by moderators. These formal requirements of an ideal debate with the notion of violating rules correspond to incorrect moves in critical discussion in the normative pragma-dialectic theory (van Eemeren and Grootendorst, 1987). *Thus, violating the rule of 'not being rude or hostile' is equivalent to committing ad hominem fallacy.* For our experiments, we scraped, in cooperation with Reddit, the complete CMV including the content of the deleted comments so we could fully reconstruct the fallacious discussions, relying on the rule violation labels provided by the moderators. The dataset contains ≈ 2M posts in 32k submissions, forming 780k unique threads.

We will set up the stage for further experiments by providing several quantitative statistics we performed on the dataset. Only 0.2% posts in CMV are ad hominem arguments. This contrasts with a typical online discussion: Coe et al. (2014) found 19.5% of comments under online news articles to be incivil. Most threads contain only a single ad hominem argument (3,396 threads; there are 3,866 ad hominem arguments in total in CMV); only 35 threads contain more than three ad hominem arguments. In 48.6% of threads containing a single ad hominem, the ad hominem argument is the very last comment. This corresponds to the popular belief that if one is out of arguments, they start attacking and the discussion is over. This trend is also shown in Figure 1 which displays the relative position of the first ad hominem argument in a thread. Replying to ad hominem with another ad hominem happens only in 15% of the cases; this speaks for the attempts of CMV participants to keep up with the standards of a rather rational discussion.

Regarding ad hominem authors, about 66% of them start attacking 'out of blue', without any previous interaction in the thread. On the other hand, 11% ad hominem authors write at least one 'normal' argument in the thread (we found one outlier who committed ad hominem after writing 57 normal arguments in the thread). Only in 20% cases, the ad hominem thread is an interplay between the original poster and another participant. It means that there are usually more people involved in an ad hominem thread. Unfortunately, sometimes the OP herself also commits ad hominem (12%).

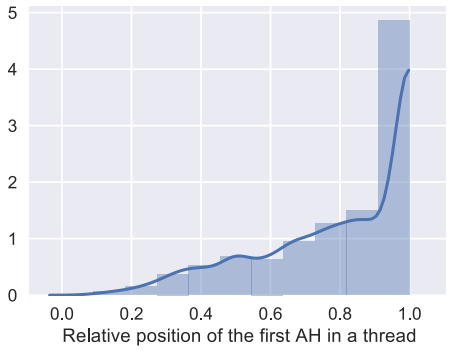


Figure 1: 'No discussion after ad hominem.' Distribution of the number of comments before the first ad hominem is committed proportional to the thread length.

We also investigated the relation between the presence of ad hominem arguments and the submission topic. While most submissions are accompanied by only one or two ad hominem arguments (75% of submissions), there are also extremes with over 50 ad hominem arguments. Manual analysis revealed that these extremes deal with religion, sexuality/gender, U.S. politics (mostly Trump), racism in the U.S., and veganism. We will elaborate on that later in Section 4.2.

4 Experiments

The experimental part is divided into three parts according to the increasing level of discourse complexity. We first experiment with ad hominem in isolation in section 4.1, then with direct ad hominem replies to original posts without dialogical exchange in section 4.2, and finally with ad hominem in a larger inter-personal discourse context in section 4.3.

4.1 Ad hominem without context in CMV

The first experimental set-up examines ad hominem arguments in *Change my view* regardless of its dialogical context.

[4] https://www.reddit.com/r/changemyview/

4.1.1 Data verification

Ad hominem arguments labeled by the CMV moderators come with no warranty. To verify their reliability, we conducted the following annotation studies. First, we needed to estimate parameters of crowdsourcing and its reliability. We sampled 100 random arguments from CMV without context: positive candidates were the reported ad hominem arguments, whereas negative candidates were sampled from comments that either violate other argumentation rules or have a delta label. To ensure the maximal content similarity of these two groups, for each positive instance the semantically closest negative instance was selected.[5] We then experimented with different numbers of Amazon Mechanical Turk workers and various thresholds of the MACE gold label estimator (Hovy et al., 2013); comparing two groups of six workers each and 0.9 threshold yielded almost perfect inter-annotator agreement (0.79 Cohen's κ). We then used this setting (six workers, 0.9 MACE threshold) to annotate another 452 random arguments sampled in the same way as above.

Crowdsourced 'gold' labels were then compared to the original CMV labels (balanced binary task: positive instances (ad hominem) and negative instances) reaching accuracy of 0.878. This means that the ad hominem labels from CMV moderators are quite reliable. Manual error analysis of disagreements revealed 11 missing ad hominem labels. These were not spotted by the moderators but were annotated as such by crowd workers.

4.1.2 Recognizing ad hominem arguments

We sampled a larger balanced set of positive instances (ad hominem) and negative instances using the same methodology as in section 4.1.1, resulting in 7,242 instances, and casted the task of recognition of ad hominem arguments as a binary supervised task. We trained two neural classifiers, namely a 2-stacked bi-directional LSTM network (Graves and Schmidhuber, 2005), and a convolutional network (Kim, 2014), and evaluated them using 10-fold cross validation. Throughout the paper we use pre-trained `word2vec` word embeddings (Mikolov et al., 2013). Detailed hyperparameters are described in the source codes (link provided in section 1). As results in Table 1 show, the task of recognizing ad hominem arguments is feasible and almost achieves the human upper bound performance.

Model	Accuracy
Human upper bound estimate	0.878
2 Stacked Bi-LSTM	0.782
CNN	**0.810**

Table 1: Prediction of ad hominem arguments

4.1.3 Typology of ad hominem

While binary classification of ad hominem as presented above might be sufficient for the purpose of red-flagging arguments, theories provide us with a much finer granularity (recall the typology in section 2). To validate whether this typology is empirically relevant, we executed an annotation experiment to classify ad hominem arguments into the provided five types (plus 'other' if none applies). We sampled 200 ad hominem arguments from threads in which interlocution happens only between two persons and which end up with ad hominem. The Mechanical Turk workers were shown this last ad hominem argument as well as the preceding one. Each instance was annotated by 16 workers to achieve a stable distribution of labels as suggested by Aroyo and Welty (2015). While 41% arguments were categorized as *abusive*, other categories (*tu quoque*, *circumstantial*, and *guilt by association*) were found to be rather ambiguous with very subtle differences. In particular, we observed a very low percentage agreement on these categories and a label distribution spiked around two or more categories. After a manual inspection we concluded that (1) the theoretical typology does not account for longer ad hominem arguments that mix up different attacks and that (2) there are actual phenomena in ad hominem arguments not covered by theoretical categories. These observations reflect those of Macagno (2013, p. 399) about ad hominem moves as multifaceted strategies.

We thus propose a list of phenomena typical to ad hominem arguments in CMV based on our empirical study. For this purpose, we follow up with another annotation experiment on 400 arguments, with seven workers per instance.[6] The goal was

[5] Similarity was computed using a cosine similarity of average embedding vectors multiplied by the argument length difference to minimize length-related artifacts. The sample was balanced with roughly 50% positive and 50% negative instances.

[6] Here we decided on seven workers per item by relying on other span annotation experiments done in a similar setup (Habernal et al., 2018b).

to annotate a text span which made the argument an ad hominem; a single argument could contain several spans. We estimated the gold spans using MACE and performed a manual post-analysis by designing a typology of causes of ad hominem together with their frequency of occurrence. The results and examples are summarized in Table 2.

4.1.4 Results and interpretation

The data verification annotation study (section 4.1.1) has two direct consequences. First, the high κ score (0.79) answers RQ2: for recognizing ad hominem argument, no previous context is necessary. Second, we still found 5% overlooked ad hominem arguments in CMV thus a moderation-facilitating tool might come handy; this can be served by the well-performing CNN model (0.810 accuracy; section 4.1.2).

The existing theoretical typology of ad hominem arguments, as presented for example in most textbooks, provides only a very simplified view. On the one hand, some of the categories which we found in the empirical labeling study (section 4.1.3) do map to their corresponding counterparts (such as the vulgar insults). On the other hand, some ad hominem insults typical to online argumentation (illiteracy insults, condescension) are not present in studies on ad hominem. Hence, we claim that any potential typology of ad hominem arguments should be multinomial rather than categorical, as we found multiple different spans in a single argument.

4.2 Triggers of first level ad hominem

In the following section, we increase the complexity of the studied discourse by taking the original post into account.

4.2.1 Annotation study

We already showed that ad hominem arguments are usually preceded by a discussion between the interlocutors. However, 897 submissions (original posts; OPs) have at least one intermediate ad hominem (in other words, the original post is directly attacked). We were thus interested in what triggers these first-level ad hominem arguments. We hypothesize two causes: (1) the *controversy* of the OP, similarly to some related works on news comments (Coe et al., 2014) and (2) the *reasonableness* of the OP (whether the topic is reasonable to argue about). We model both features on a three-point scale, namely *controversy*: 1 = 'not really controversial', 2 = 'somehow controversial', 3 = 'very controversial' and *reasonableness*: 1 = 'quite stupid', 2 = 'neutral', 3 = 'quite reasonable'.[7]

We sampled two groups of OPs: those which had some ad hominem arguments in any of its threads but no delta (ad hominem group) and those without ad hominem but some deltas (Delta group). In total, 1,800 balanced instances were annotated by five workers and the resulting value was averaged for each item.[8]

Statistical analysis of the annotated 1,800 OPs revealed that ad hominem arguments are associated with more controversial OPs (mean controversy 1.23) while delta-awarded arguments with less controversial OPs (mean controversy 1.06; K-S test;[9] statistics 0.13, P-value: 7.97×10^{-7}). On the other hand, reasonableness does not seem to play such a role. The difference between ad hominem in reasonable OPs (mean 1.20) and delta in reasonable OPs (mean 1.11) is not that statistically strong; (K-S test statistics: 0.07, P-value: 0.02).

4.2.2 Regression model for predicting controversy and reasonableness

We further built a regression model for predicting controversy and reasonableness of the OPs. Along with Bi-LSTM and CNN networks (same models as in 4.1.2) we also developed a neural model that integrates CNN with topic distribution (CNN+LDA). The motivation for a topic-incorporating model was based on our earlier observations presented in section 3. In particular, we trained an LDA topic model (k = 50) (Blei et al., 2003) on the heldout OPs and during training/testing, we merged the estimated topic distribution vector with the output layer after convolution and pooling. We performed 10-fold cross validation on the 1,800 annotated OPs and got reasonable performance for controversy prediction (ρ

[7] Examples of not really controversial: *"I Don't Think Monty Python is Funny"*, very controversial: *"Blacks are generally intellectual inferior to the other major races"*, quite stupid: *"Burritos are better than sandwiches"*, and quite reasonable: *"Nations whose leadership is based upon religion are fundamentally backwards"*.

[8] A pilot crowd sourcing annotation with 5 + 5 workers showed a fair reliability for controversy (Spearman's ρ 0.804) and medium reliability for reasonableness (Spearman's ρ 0.646).

[9] Kolmogorov-Smirnov (K-S) test is a non-parametric test without any assumptions about the underlying probability distribution.

Type	(%)	Example spans
Vulgar insult	31.3	"Your just an asshole", "you dumb fuck", etc.
Illiteracy insult	13.0	"Reading comprehension is your friend", "If you can't grasp the concept, I can't help you"
Condescension	6.5	"little buddy", "sir", "boy", "Again, how old are you?"
Ridiculing and sarcasm	6.5	"Thank you so much for all your pretentious explanations", "Can you also use Google?"
'Idiot'-insults	6.5	"Ever have discussions with narcissistic idiots on the internet? They are so tiring"
Accusation of stupidity	4.3	"You have no capability to understand why", "You're obviously just Nobody with enough brains to operate a computer could possibly believe something this stupid"
Lack of argumentation skills	4.3	"You're making the claims, it's your job to prove it. Don't you know how debating works?", "You're trash at debating."
Accusation of trolling	3.9	"You're just a dishonest troll", "You're using troll tactics"
Accusation of ignorance	3.5	"Please dont waste peoples time pretending to know what you're talking about", "Do you even know what you're saying?"
"You didn't read what I wrote"	3.0	"Read what I posted before acting like a pompous ass", "Did you even read this?"
"What you say is idiotic"	2.6	"To say that people intrinsically understand portion size is idiotic.", "Your second paragraph is fairly idiotic"
Accusation of lying	2.6	"Possible lie any harder?", "You are just a liar."
"You don't face the facts and ignore the obvious"	1.7	"Willful ignorance is not something I can combat", "How can you explain that? You can't because it will hurt your feelings to face reality"
Accusation of ad hominem or other fallacies	1.7	"You started with a fallacy and then deflected.", "You still refuse to acknowledge that you used a strawman argument against me"
Other	8.3	"Wow. Someone sounds like a bit of an anti-semite", "You're too dishonest to actually quote the verse because you know it's bullshit"

Table 2: What makes an argument ad hominem: results of the empirical study of labeling spans in 400 ad hominem arguments.

Controversy (Spearman's ρ)	
Human upper bounds	0.804
Bi-LSTM	0.539
CNN	0.559
CNN-LDA	**0.569**
Reasonableness (Spearman's ρ)	
Human upper bounds	0.646
Bi-LSTM	0.332
CNN	0.320
CNN-LDA	**0.385**

Table 3: Results of predicting controversy and reasonableness of the original post.

0.569) and medium performance for reasonableness prediction (ρ 0.385), respectively; both using the CNN+LDA model (see Table 3).

We then used the trained model and extrapolated on all held-out OPs (1,267 ad hominem and 10,861 delta OPs, respectively). The analysis again showed that ad hominem arguments tend to be found under more controversial OPs whereas delta arguments in the less controversial ones (K-S test statistics: 0.14, P-value: 1×10^{-18}). For reasonableness, the rather low performance of the predictor does not allow us draw any conclusions on the extrapolated data.

4.2.3 Results and interpretation

Controversy of the original post is immediately heating up the debate participants and correlates with a higher number of direct ad hominem responses. This corresponds to observations made in comments in newswire where 'weightier' topics tended to stir incivility (Coe et al., 2014). On the other hand, 'stupidity' (or 'reasonableness') does not seem to play any significant role. The CNN+LDA model for predicting controversy (ρ 0.569) might come handy for signaling potentially 'heated' discussions.

4.3 Before calling names

In this section, we focus on the dialogical aspect of CMV debates and dynamics of ad hominem fallacies. Although ad hominem arguments appear in many forms (Section 4.1.3), we treat all ad hominem arguments equal in the following experiments.

4.3.1 Data sampling

So far we explored what makes an ad hominem argument and whether debated topic influences the

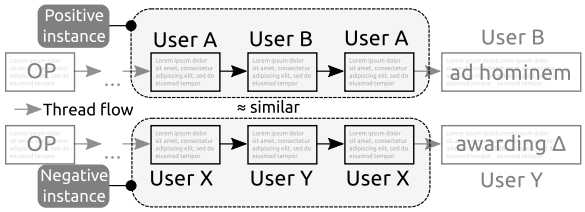


Figure 2: Sampling instances for learning triggers of ad hominem.

number of intermediate attacks. However, possible causes of the argumentative dynamics that ends up with an ad hominem argument remain an open question, which has been addressed in neither argumentation theory nor in cognitive psychology, to the best of our knowledge. We thus cast an explanation of triggers and dynamics of ad hominem discussions as a supervised machine learning problem and draw theoretical insights by a retrospective interpretation of the learned models.

We sample positive instances by taking three contextual arguments preceding the ad hominem argument from threads which are an interplay between two persons. Negative samples are drawn similarly from threads in which the argument is awarded with Δ as shown in Figure 2.[10] Each instance consists of the three concatenated arguments delimited by a special OOV token. This resulted in 2,582 balanced training instances.

4.3.2 Neural models

The alleged lack of interpretability of neural networks has motivated several lines of approaches, such as layer-wise relevance propagation (Arras et al., 2017) or representation erasure (Li et al., 2016), both on sentiment analysis. As our task at hand deals with multi-party discourse that presumably involves temporal relations important for the learned representation, we opted for a state-of-the-art self-attentive LSTM model. In particular, we re-implemented the Structured Self-Attentive Embedding Neural Network (SSAE-NN) (Lin et al., 2017) which learns an embedding matrix representation of the input using attention weights. To make the attention even more interpretable, we replaced the final non-linear MLP layers with a single linear classifier (softmax). By summing over one dimension of the attention embedding matrix, each word from the input sequence gets associated with a single attention weight that gives us insights into the classifier's 'features' (still indirectly, as the true representation is a matrix; see the original paper).[11] The learning objective is to recognize whether the thread ends up in an ad hominem argument or a delta point. We trained the model in 10-fold cross-validation and although our goal is not to achieve the best performance but rather to gain insight, we also tested a CNN model (accuracy 0.7095) which performed slightly worse than the SSAE-NN model (accuracy 0.7208).

4.3.3 Results and interpretation

During testing the model, we projected attention weights to the original texts as heat maps and manually analyzed 191 true positives (ad hominem threads recognized correctly), as well as 77 false positives (ad hominem threads misclassified as delta) and 84 false negatives (delta as ad hominem), in total about 120k tokens. The full output is available in the supplementary materials, we use IDs as a reference in the following text.

In the following analysis, we solely relied on the weights of words or phrases learned by the attention model, see an example in Figure 3. Based on our observations, we summarize several linguistic and argumentative phenomena with examples most likely responsible for ad hominem threads in Table 4.

The identified phenomena have few interesting properties in common. First, they all are topic-independent rhetorical devices (except for the loaded keywords at the bottom). Second, many of them deal with meta-level argumentation, i.e., arguing about argumentation (such as missing support or fallacy accusations). Third, most of them do not contain profanity (in contrast to the actual ad hominem arguments of which a third are vulgar insults; cf. Table 2). And finally, all of them should be easy to avoid.

Misleading 'features' False positives revealed properties that misled the network to classify delta threads as ad hominem threads.

- These include **topic words** (such as *racism, blacks, slave, abortion*) which reflects the implicit bias in the data.
- Actual interest mixed with indifference in

[10] To ensure as much content similarity as possible, we used the same similarity sampling as in section 4.1.1.

[11] We also experimented with regularizing the attention matrix as the authors proposed, but it resulted in worse performance.

587_ah_t1_cm7djx3

(OOV_comment_begin) If only you would n't rely on [fallacious] (http : OOV) [arguments] (http : OOV) to make your point. So no , I do n't realize how stupid and naive I am. All I 've realized is that you are n't actually prepared to have an actual discussion .

(OOV_comment_begin) What god do you believe in ? And it 's not a fallacy when it 's very comparable to the most popular gods .

(OOV_comment_begin) You 're making an assumption on what I believe , then attacking your assumption of what my belief is without me even telling you anything. OOV It is a OOV It 's the comparison itself that is OOV If they were n't comparable at all , then it 'd be impossible to commit the OOV You can compare apples to oranges , but the moment you use your fingernails to peel an apple you look like an idiot .

Figure 3: An example of reconstructed word weight heat map extracted from the attention matrix for a thread which ends up in ad hominem; three previous arguments are shown (see Figure 2 for sampling details).

Phenomena	**Examples**
Introducing vulgar intensifiers or interrogatives	**388(-1)** *"Where the fuck is your idea to ..."*, **712(-2)** *"no shortage of fucking gun"*, **1277(-1)** *"This is fucking CMV"*, **428(-2)** *"I'm fucking trans!"*, **2018(-2)** *"an arrogant fuck"*, **1277(-2)** *"What the fuck are you smoking?"*
Direct imperatives	**1003(-3)** *"You should get more mad about it"*, **857(-2)** *"You need to do a lot better than that."*, **233(-2)** *"So now delete your post"*, **749(-1)** *"google this fact as well"*, **1276(-1)** *"Just look back at the reasons why ..."*
Accusing of believing in or using propaganda	**522(-1)** *"It's right wing propaganda?"*, **1003(-1)** *"If you're not outraged, you're not paying attention to our propaganda that says the opposite of literally thousands of published research articles"*
Accusation of fallacies or bad argumentation practice	**238(-3)** *"your snide remarks and poor argumentation skills"*, **1117(-2)** *"you're circle jerking A vs. B"*, **263(-3)** *"You're grasping at straws"*, **78(-3)** *"You sure like changing words and statements to make your argument appear more cogent, don't you?"*, **210(-1)** *"Your arguments range from ... to ..."*, **1085(-3)** *"It's only a fallacy"*, **144(-1)** *"You haven't presented any evidence or argument that disagrees with anything I've said."*, **587(-3)** *"If only you wouldn't rely on fallacious arguments"*
Reinterpreting opponent's positions	**982(-1)** *"The fact that you obviously think ... reveals ..."*, **982(-2)** *"What makes you think I see myself ... ?"*, **1060(-3)** *"That kind of thinking is ..."*, **760(-1)** *"If I'm understanding you correctly"*, **405(-1)** *"... deluded yourself into believing factually incorrect things"* **587(-1)** *"You're making an assumption on what I believe, then attacking your assumption of what my belief is without me even telling you anything."*
Accusation of not reading the other party's arguments	**586(-1)** *"... me without even reading my ..."*, **240(-1)** *"You are just reading it wrong."*, **310(-1)** *"Oh, you're not actually reading my ..."*
Pointing at missing or unsupported evidence and facts	**1238(-2)** *"unsupported bullshit as before"*, **1121(-3)** *"you can't chose your facts"*, **931(-1)** *"If that's your only argument ..."*, **486(-2)** *"unsubstantiated statement"*, **486(-1)** *"unsupported claims"*, **71(-2)** *"factually correct"*, **915(-1)** *"But for the sake of argument, your points are pitifully .."*, **388(-3)** *"Please provide statistics ... It's silly to debate statistics without actual numbers."*
UPPERCASE	**1238(-3)** *"NO ONE CLAIMED THAT ... ARE NOT ... AGAINST ..."*
Sarcasm	**78(-2)** *"But I'm sure you know best"*, **310(-1)** *"Have a nice day."*, **1276(-1)** *"Good luck with that"*
Mentions of trolling	**701(-2)** *"Then you are giving trolls the victory then?"*
Loaded keywords	*Nazi, rape, racist*

Table 4: Phenomena resulting into ad hominem learned by the SSAE-NN model. The first number is the instance ID (available in the supplementary material), the minus number in parentheses is the position of the argument before the ad hominem.

sarcasm is also problematic (**185(-2)** *"That's a very interesting ..."*).

- Another problematic phenomena is also **expressed disagreement** (**678(-2)** *"overheated rhetoric"*, **203(-2)** *"But I suppose this argument is ..."*, **230(-2)** *"But I don't think it's quite ..."*, **938(-1)** *"I disagree too, however ..."*).

False negatives were caused basically by presence of many 'informative' **content words** (**980** *unemployment, quarterly publication, inflation data*, **474** *actual publications, this experiment, biological ailments, medical doctorate*, **1214** *graduate degree, education, health insurance*) and **misinterpreted sarcasm** (**285(-1)** *"Also this is a cute analogy"*).

5 Conclusion

In this article, we investigated ad hominem argumentation on three levels of discourse complexity. We looked into qualitative and quantative properties of ad hominem arguments, crowdsourced labeled data, experimented with models for prediction (0.810 accuracy; 4.1.2), and proposed an updated typology of ad hominem properties (4.1.3). We then looked into the dynamics of argumentation to examine the relation between the quality of the original post and immediate ad hominem arguments (4.2). Finally, we exploited the learned representation of Self-Attentive Embedding Neural Network to search for features triggering ad hominem in one-to-one discussions. We found several categories of rhetorical devices as well as misleading features (4.3.3).

There are several points that deserve further investigation. First, we have ignored meta-information of the debate participants, such as their overall activity (i.e., whether they are spammers or trolls). Second, the proposed typology of ad hominem causes has not yet been post-verified empirically. Third, we expect that personality traits of the participants (BIG5) may also play a significant role in the argumentative exchange. We leave these points for future work.

We believe that our findings will help gain better understanding of, and hopefully keep restraining from, ad hominem fallacies in good-faith discussions.

Acknowledgments

This work has been supported by the ArguAna Project GU 798/20-1 (DFG), and by the DFG-funded research training group "Adaptive Preparation of Information form Heterogeneous Sources" (AIPHES, GRK 1994/1).

References

Aristotle and George Kennedy (translator). 1991. *On Rhetoric: A Theory of Civil Discourse*. Oxford University Press, USA.

Lora Aroyo and Chris Welty. 2015. Truth is a lie: Crowd truth and the seven myths of human annotation. *AI Magazine* 36(1):15–24. https://doi.org/10.1609/aimag.v36i1.2564.

Leila Arras, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2017. Explaining Recurrent Neural Network Predictions in Sentiment Analysis. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. Association for Computational Linguistics, Copenhagen, Denmark, pages 159–168. http://www.aclweb.org/anthology/W17-5221.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research* 3:993–1022. http://dl.acm.org/citation.cfm?id=944919.944937.

Maarten Boudry, Fabio Paglieri, and Massimo Pigliucci. 2015. The Fake, the Flimsy, and the Fallacious: Demarcating Arguments in Real Life. *Argumentation* 29(4):431–456. https://doi.org/10.1007/s10503-015-9359-1.

Kevin Coe, Kate Kenski, and Stephen A. Rains. 2014. Online and Uncivil? Patterns and Determinants of Incivility in Newspaper Website Comments. *Journal of Communication* 64(4):658–679. https://doi.org/10.1111/jcom.12104.

Stephen de Wijze. 2002. Complexity, Relevance and Character: Problems with Teaching the "Ad Hominem" Fallacy. *Educational Philosophy and Theory* 35(1):31–56. https://doi.org/10.1111/1469-5812.00004.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks* 18(5):602–610. https://doi.org/10.1016/j.neunet.2005.06.042.

Ivan Habernal and Iryna Gurevych. 2016a. What makes a convincing argument? Empirical analysis and detecting attributes of convincingness in Web argumentation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1214–1223. http://aclweb.org/anthology/D16-1129.

Ivan Habernal and Iryna Gurevych. 2016b. Which argument is more convincing? Analyzing and predicting convincingness of Web arguments using bidirectional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1589–1599. http://www.aclweb.org/anthology/P16-1150.

Ivan Habernal and Iryna Gurevych. 2017. Argumentation Mining in User-Generated Web Discourse. *Computational Linguistics* 43(1):125–179. https://doi.org/10.1162/COLI_a_00276.

Ivan Habernal, Raffael Hannemann, Christian Pollak, Christopher Klamm, Patrick Pauli, and Iryna Gurevych. 2017. Argotario: Computational Argumentation Meets Serious Games. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Copenhagen, Denmark, pages 7–12. http://www.aclweb.org/anthology/D17-2002.

Ivan Habernal, Patrick Pauli, and Iryna Gurevych. 2018a. Adapting Serious Game for Fallacious Argumentation to German: Pitfalls, Insights, and Best Practices. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. European Language Resources Association (ELRA), Miyazaki, Japan, page (to appear).

Ivan Habernal, Henning Wachsmuth, Iryna Gurevych, and Benno Stein. 2018b. The Argument Reasoning Comprehension Task: Identification and Reconstruction of Implicit Warrants. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, New Orleans, LA, page (to appear). https://arxiv.org/abs/1708.01425.

Charles L. Hamblin. 1970. *Fallacies*. Methuen, London, UK.

Hans Hansen. 2017. Fallacies. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*, Metaphysics Research Lab, Stanford University. http://plato.stanford.edu/entries/fallacies/.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with MACE. In *Proceedings of NAACL-HLT 2013*. Association for Computational Linguistics, Atlanta, Georgia, pages 1120–1130. http://www.aclweb.org/anthology/N13-1132.

Siddharth Jain, Archna Bhatia, Angelique Rein, and Eduard Hovy. 2014. A Corpus of Participant Roles in Contentious Discussions. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. European Language Resources Association (ELRA), Reykjavik, Iceland, pages 1751–1756.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1746–1751. http://www.aclweb.org/anthology/D14-1181.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. Understanding Neural Networks through Representation Erasure. In *arXiv preprint*. http://arxiv.org/abs/1612.08220.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A Structured Self-attentive Sentence Embedding. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*. Toulon, France, pages 1–15. http://arxiv.org/abs/1703.03130.

Fabrizio Macagno. 2013. Strategies of character attack. *Argumentation* 27(4):369–401. https://doi.org/10.1007/s10503-013-9291-1.

Christopher D. Manning. 2015. Computational Linguistics and Deep Learning. *Computational Linguistics* 41(4):701–707. https://doi.org/10.1162/COLI_a_00239.

Hugo Mercier and Dan Sperber. 2011. Why do humans reason? Arguments for an argumentative theory. *The Behavioral and Brain Sciences* 34(2):57–74; discussion 74–111. https://doi.org/10.1017/S0140525X10000968.

Hugo Mercier and Dan Sperber. 2017. *The Enigma of Reason*. Harvard University Press, Cambridge, MA, USA.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., pages 3111–3119.

Elena Musi. 2017. How did you change my view? A corpus-based study of concessions' argumentative role. *Discourse Studies* page (in press). https://doi.org/10.1177/1461445617734955.

Ahmed Sahlane. 2012. Argumentation and fallacy in the justification of the 2003 War on Iraq. *Argumentation* 26(4):459–488. https://doi.org/10.1007/s10503-012-9265-8.

Haji Mohammad Saleem, Kelly P Dillon, Susan Benesch, and Derek Ruths. 2016. A Web of Hate: Tackling Hateful Speech in Online Social Spaces. In Guy De Pauw, Ben Verhoeven, Bart Desmet, and Els Lefever, editors, *Proceedings of the First Workshop on Text Analytics for Cybersecurity and Online Safety*. European Language Resources Association (ELRA), Portorož, Slovenia, pages 1–9.

Edward Schiappa and John P. Nordin. 2013. *Argumentation: Keeping Faith with Reason*. Pearson UK, 1st edition.

Christian Stab and Iryna Gurevych. 2017. Recognizing Insufficiently Supported Arguments in Argumentative Essays. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*. Association for Computational Linguistics, pages 980–990. http://www.aclweb.org/anthology/E17-1092.

Chenhao Tan, Vlad Niculae, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2016. Winning Arguments: Interaction Dynamics and Persuasion Strategies in Good-faith Online Discussions. In *Proceedings of the 25th International Conference on World Wide Web*. ACM, Montreal, CA, page (to appear). http://arxiv.org/abs/1602.01103.

Christopher W. Tindale. 2007. *Fallacies and Argument Appraisal*. Cambridge University Press, New York, NY, USA, critical reasoning and argumentation edition.

Stephen E. Toulmin. 1958. *The Uses of Argument*. Cambridge University Press.

Frans H. van Eemeren, Bart Garssen, Erik C. W. Krabbe, A. Francisca Snoeck Henkemans, Bart Verheij, and Jean H. M. Wagemans. 2014. *Handbook of Argumentation Theory*. Springer, Berlin/Heidelberg.

Frans H. van Eemeren and Rob Grootendorst. 1987. Fallacies in pragma-dialectical perspective. *Argumentation* 1(3):283–301. https://doi.org/10.1007/BF00136779.

Frans H. van Eemeren and Rob Grootendorst. 1992. *Argumentation, communication, and fallacies: a pragma-dialectical perspective*. Lawrence Erlbaum Associates, Inc.

Henning Wachsmuth, Nona Naderi, Ivan Habernal, Yufang Hou, Graeme Hirst, Iryna Gurevych, and Benno Stein. 2017. Argumentation Quality Assessment: Theory vs. Practice. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 250–255. http://aclweb.org/anthology/P17-2039.

Douglas Walton. 2007. *Media Argumentation: Dialect, Persuasion and Rhetoric*. Cambridge University Press.

Alex Wang, William L. Hamilton, and Jure Leskovec. 2016. Learning Linguistic Descriptors of User Roles in Online Communities. In *Proceedings of 2016 EMNLP Workshop on Natural Language Processing and Computational Social Science*. Association for Computational Linguistics, Austin, Texas, pages 76–85. http://aclweb.org/anthology/W16-5610.

John Woods. 2008. Lightening up on the Ad Hominem. *Informal Logic* 27(1):109. https://doi.org/10.22329/il.v27i1.467.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex Machina: Personal Attacks Seen at Scale. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, Perth, Australia, pages 1391–1399. https://doi.org/10.1145/3038912.3052591.

Amy Zhang, Bryan Culbertson, and Praveen Paritosh. 2017. Characterizing Online Discussion Using Coarse Discourse Sequences. In *Proceedings of the Eleventh International AAAI Conference on Web and Social Media (ICWSM 2017)*. AAAI Press, Montreal, Canada, pages 357–366.

Scene Graph Parsing as Dependency Parsing

Yu-Siang Wang
National Taiwan University
b03202047@ntu.edu.tw

Chenxi Liu(✉)
Johns Hopkins University
cxliu@jhu.edu

Xiaohui Zeng
Hong Kong University of Science and Technology
xzengaf@connect.ust.hk

Alan Yuille
Johns Hopkins University
alan.yuille@jhu.edu

Abstract

In this paper, we study the problem of parsing structured knowledge graphs from textual descriptions. In particular, we consider the scene graph representation (Johnson et al., 2015) that considers objects together with their attributes and relations: this representation has been proved useful across a variety of vision and language applications. We begin by introducing an alternative but equivalent edge-centric view of scene graphs that connect to dependency parses. Together with a careful redesign of label and action space, we combine the two-stage pipeline used in prior work (generic dependency parsing followed by simple post-processing) into one, enabling end-to-end training. The scene graphs generated by our learned neural dependency parser achieve an F-score similarity of 49.67% to ground truth graphs on our evaluation set, surpassing best previous approaches by 5%. We further demonstrate the effectiveness of our learned parser on image retrieval applications.[1]

1 Introduction

Recent years have witnessed the rise of interest in many tasks at the intersection of computer vision and natural language processing, including semantic image retrieval (Johnson et al., 2015; Vendrov et al., 2015), image captioning (Mao et al., 2014; Karpathy and Li, 2015; Donahue et al., 2015; Liu et al., 2017b), visual question answering (Antol et al., 2015; Zhu et al., 2016; Andreas et al., 2016), and referring expressions (Hu et al., 2016; Mao et al., 2016; Liu et al., 2017a). The pursuit for these tasks is in line with people's desire for high level understanding of visual content, in particular, using textual descriptions or questions to help understand or express images and scenes.

What is shared among all these tasks is the need for a *common representation* to establish connection between the two different modalities. The majority of recent works handle the vision side with convolutional neural networks, and the language side with recurrent neural networks (Hochreiter and Schmidhuber, 1997; Cho et al., 2014) or word embeddings (Mikolov et al., 2013; Pennington et al., 2014). In either case, neural networks map original sources into a semantically meaningful (Donahue et al., 2014; Mikolov et al., 2013) vector representation that can be aligned through end-to-end training (Frome et al., 2013). This suggests that the vector embedding space is an appropriate choice as the common representation connecting different modalities (see e.g. Kaiser et al. (2017)).

While the dense vector representation yields impressive performance, it has an unfortunate limitation of being less intuitive and hard to interpret. Scene graphs (Johnson et al., 2015), on the other hand, proposed a type of directed graph to encode information in terms of objects, attributes of objects, and relationships between objects (see Figure 1 for visualization). This is a more structured and explainable way of expressing the knowledge from either modality, and is able to serve as an alternative form of common representation. In fact, the value of scene graph representation has already been proven in a wide range of visual tasks, including semantic image retrieval (Johnson et al., 2015), caption quality evaluation (Anderson et al., 2016), etc. In this paper, we focus on scene graph generation from textual descriptions.

Previous attempts at this problem (Schuster et al., 2015; Anderson et al., 2016) follow the same spirit. They first use a dependency parser to obtain the dependency relationship for all words in a sentence, and then use either a rule-based or a learned classifier as post-processing to generate the scene graph. However, the rule-based classifier cannot

[1]Code is available at https://github.com/Yusics/bist-parser/tree/sgparser

Proceedings of NAACL-HLT 2018, pages 397–407
New Orleans, Louisiana, June 1 - 6, 2018.

Figure 1: Each image in the Visual Genome (Krishna et al., 2017) dataset contains tens of region descriptions and the region scene graphs associated with them. In this paper, we study how to generate high quality scene graphs (two such examples are shown in the figure) from textual descriptions, without using image information.

learn from data, and the learned classifier is rather simple with hand-engineered features. In addition, the dependency parser was trained on linguistics data to produce complete dependency trees, some parts of which may be redundant and hence confuse the scene graph generation process.

Therefore, our model abandons the two-stage pipeline, and uses a single, customized dependency parser instead. The customization is necessary for two reasons. First is the difference in label space. Standard dependency parsing has tens of edge labels to represent rich relationships between words in a sentence, but in scene graphs we are only interested in three types, namely objects, attributes, and relations. Second is whether every word needs a head. In some sense, the scene graph represents the "skeleton" of the sentence, which suggests that empty words are unlikely to be included in the scene graph. We argue that in scene graph generation, it is unnecessary to require a parent word for every single word.

We build our model on top of a neural dependency parser implementation (Kiperwasser and Goldberg, 2016) that is among the state-of-the-art. We show that our carefully customized dependency parser is able to generate high quality scene graphs by learning from data. Specifically, we use the Visual Genome dataset (Krishna et al., 2017), which provides rich amounts of region description - region graph pairs. We first align nodes in region graphs with words in the region descriptions using simple rules, and then use this alignment to train our customized dependency parser. We evaluate our parser by computing the F-score between the parsed scene graphs and ground truth scene graphs. We also apply our approach to image retrieval to show its effectiveness.

2 Related Works

2.1 Scene Graphs

The scene graph representation was proposed in Johnson et al. (2015) as a way to represent the rich, structured knowledge within an image. The nodes in a scene graph represent either an object, an attribute for an object, or a relationship between two objects. The edges depict the connection and association between two nodes. This representation is later adopted in the Visual Genome dataset (Krishna et al., 2017), where a large number of scene graphs are annotated through crowd-sourcing.

The scene graph representation has been proved useful in various problems including semantic image retrieval (Johnson et al., 2015), visual question answering (Teney et al., 2016), 3D scene synthesis (Chang et al., 2014), and visual relationship detection (Lu et al., 2016). Excluding Johnson et al. (2015) which used ground truth, scene graphs are obtained either from images (Dai et al., 2017; Xu et al., 2017; Li et al., 2017) or from textual descriptions (Schuster et al., 2015; Anderson et al., 2016). In this paper we focus on the latter.

In particular, parsed scene graphs are used in Schuster et al. (2015) for image retrieval. We show that with our more accurate scene graph parser, performance on this task can be further improved.

2.2 Parsing to Graph Representations

The goal of dependency parsing (Kübler et al., 2009) is to assign a parent word to every word in a sentence, and every such connection is associated with a label. Dependency parsing is particularly suitable for scene graph generation because it directly models the relationship between individual

words without introducing extra nonterminals. In fact, all previous work (Schuster et al., 2015; Anderson et al., 2016) on scene graph generation run dependency parsing on the textual description as a first step, followed by either heuristic rules or simple classifiers. Instead of running two separate stages, our work proposed to use a single dependency parser that is end-to-end. In other words, our customized dependency parser generates the scene graph in an online fashion as it reads the textual description once from left to right.

In recent years, dependency parsing with neural network features (Chen and Manning, 2014; Dyer et al., 2015; Cross and Huang, 2016; Kiperwasser and Goldberg, 2016; Dozat and Manning, 2016; Shi et al., 2017) has shown impressive performance. In particular, Kiperwasser and Goldberg (2016) used bidirectional LSTMs to generate features for individual words, which are then used to predict parsing actions. We base our model on Kiperwasser and Goldberg (2016) for both its simplicity and good performance.

Apart from dependency parsing, Abstract Meaning Representation (AMR) parsing (Flanigan et al., 2014; Werling et al., 2015; Wang et al., 2015; Konstas et al., 2017) may also benefit scene graph generation. However, as first pointed out in Anderson et al. (2016), the use of dependency trees still appears to be a common theme in the literature, and we leave the exploration of AMR parsing for scene graph generation as future work.

More broadly, our task also relates to entity and relation extraction, e.g. Katiyar and Cardie (2017), but there object attributes are not handled. Neural module networks (Andreas et al., 2016) also use dependency parses, but they translate questions into a series of actions, whereas we parse descriptions into their graph form. Finally, Krishnamurthy and Kollar (2013) connected parsing and grounding by training the parser in a weakly supervised fashion.

3 Task Description

In this section, we begin by reviewing the scene graph representation, and show how its nodes and edges relate to the words and arcs in dependency parsing. We then describe simple yet reliable rules to align nodes in scene graphs with words in textual descriptions, such that customized dependency parsing, described in the next section, may be trained and applied.

3.1 Scene Graph Definition

There are three types of nodes in a scene graph: object, attribute, and relation. Let $\mathcal{O}$ be the set of object classes, $\mathcal{A}$ be the set of attribute types, and $\mathcal{R}$ be the set of relation types. Given a sentence s, our goal in this paper is to parse s into a scene graph:

$$G(s) = \langle O(s), A(s), R(s) \rangle \tag{1}$$

where $O(s) = \{o_1(s), \ldots, o_m(s)\}, o_i(s) \in \mathcal{O}$ is the set of object instances mentioned in s, $A(s) \subseteq O(s) \times \mathcal{A}$ is the set of attributes associated with object instances, and $R(s) \subseteq O(s) \times \mathcal{R} \times O(s)$ is the set of relations between object instances.

$G(s)$ is a graph because we can first create an object node for every element in $O(s)$; then for every (o, a) pair in $A(s)$, we create an attribute node and add an unlabeled edge $o \rightarrow a$; finally for every (o_1, r, o_2) triplet in $R(s)$, we create a relation node and add two unlabeled edges $o_1 \rightarrow r$ and $r \rightarrow o_2$. The resulting directed graph exactly encodes information in $G(s)$. We call this the **node-centric** graph representation of a scene graph.

We realize that a scene graph can be equivalently represented by no longer distinguishing between the three types of nodes, yet assigning labels to the edges instead. Concretely, this means there is now only one type of node, but we assign a `ATTR` label for every $o \rightarrow a$ edge, a `SUBJ` label for every $o_1 \rightarrow r$ edge, and a `OBJT` label for every $r \rightarrow o_2$ edge. We call this the **edge-centric** graph representation of a scene graph.

We can now establish a connection between scene graphs and dependency trees. Here we only consider scene graphs that are acyclic[2]. The edge-centric view of a scene graph is very similar to a dependency tree: they are both directed acyclic graphs where the edges/arcs have labels. The difference is that in a scene graph, the nodes are the objects/attributes/relations and the edges have label space {`ATTR`, `SUBJ`, `OBJT`}, whereas in a dependency tree, the nodes are individual words in a sentence and the edges have a much larger label space.

3.2 Sentence-Graph Alignment

We have shown the connection between nodes in scene graphs and words in dependency parsing. With alignment between nodes in scene

[2]In Visual Genome, only 4.8% region graphs have cyclic structures.

graphs and words in the textual description, scene graph generation and dependency parsing becomes equivalent: we can construct the generated scene graph from the set of labeled edges returned by the dependency parser. Unfortunately, such alignment is not provided between the region graphs and region descriptions in the Visual Genome (Krishna et al., 2017) dataset. Here we describe how we use simple yet reliable rules to do sentence-graph (word-node) alignment.

There are two strategies that we could use in deciding whether to align a scene graph node d (whose label space is $\mathcal{O} \cup \mathcal{A} \cup \mathcal{R}$) with a word/phrase w in the sentence:

- Word-by-word match (WBW): $d \leftrightarrow w$ only when d's label and w match word-for-word.
- Synonym match (SYN)[3]: $d \leftrightarrow w$ when the wordnet synonyms of d's label contain w.

Obviously WBW is a more conservative strategy than SYN.

We propose to use two cycles and each cycle further consists of three steps, where we try to align objects, attributes, relations in that order. The pseudocode for the first cycle is in Algorithm 1. The second cycle repeats line 4-15 immediately afterwards, except that in line 6 we also allow SYN. Intuitively, in the first cycle we use a conservative strategy to find "safe" objects, and then scan for their attributes and relations. In the second cycle we relax and allow synonyms in aligning object nodes, also followed by the alignment of attribute and relation nodes.

The ablation study of the alignment procedure is reported in the experimental section.

4 Customized Dependency Parsing

In the previous section, we have established the connection between scene graph generation and dependency parsing, which assigns a parent word for every word in a sentence, as well as a label for this directed arc. We start by describing our base dependency parsing model, which is neural network based and performs among the state-of-the-art. We then show why and how we do customization, such that scene graph generation is achieved with a single, end-to-end model.

[3]This strategy is also used in (Denkowski and Lavie, 2014) and (Anderson et al., 2016).

Algorithm 1: First cycle of the alignment procedure.

1 **Input**: Sentence s; Scene graph $G(s)$
2 Initialize aligned nodes N as empty set
3 Initialize aligned words W as empty set
4 **for** o in object nodes of $G(s) \setminus N$ **do**
5 **for** w in $s \setminus W$ **do**
6 **if** $o \leftrightarrow w$ according to WBW **then**
7 Add (o, w); $N = N \cup \{o\}$; $W = W \cup \{w\}$
8 **for** a in attribute nodes of $G(s) \setminus N$ **do**
9 **for** w in $s \setminus W$ **do**
10 **if** $a \leftrightarrow w$ according to WBW or SYN **and** a's object node is in N **then**
11 Add (a, w); $N = N \cup \{a\}$; $W = W \cup \{w\}$
12 **for** r in relation nodes of $G(s) \setminus N$ **do**
13 **for** w in $s \setminus W$ **do**
14 **if** $r \leftrightarrow w$ according to WBW or SYN **and** r's subject and object nodes are both in N **then**
15 Add (r, w); $N = N \cup \{r\}$; $W = W \cup \{w\}$

4.1 Neural Dependency Parsing Base Model

We base our model on the transition-based parser of Kiperwasser and Goldberg (2016). Here we describe its key components: the arc-hybrid system that defines the transition actions, the neural architecture for feature extractor and scoring function, and the loss function.

The Arc-Hybrid System In the arc-hybrid system, a configuration consists of a stack σ, a buffer β, and a set T of dependency arcs. Given a sentence $s = w_1, \ldots, w_n$, the system is initialized with an empty stack σ, an empty arc set T, and $\beta = 1, \ldots, n, \texttt{ROOT}$, where ROOT is a special index. The system terminates when σ is empty and β contains only ROOT. The dependency tree is given by the arc set T upon termination.

The arc-hybrid system allows three transition actions, SHIFT, LEFT$_l$, RIGHT$_l$, described in Table 1. The SHIFT transition moves the first element of the buffer to the stack. The LEFT(l) transition yields an arc from the first element of the buffer to the top element of the stack, and then removes the top element from the stack. The

Stack σ_t	Buffer β_t	Arc set T_t	Action	Stack σ_{t+1}	Buffer β_{t+1}	Arc set T_{t+1}
σ	$b_0\|\beta$	T	SHIFT	$\sigma\|b_0$	β	T
$\sigma\|s_1\|s_0$	$b_0\|\beta$	T	LEFT(l)	$\sigma\|s_1$	$b_0\|\beta$	$T \cup \{(b_0, s_0, l)\}$
$\sigma\|s_1\|s_0$	β	T	RIGHT(l)	$\sigma\|s_1$	β	$T \cup \{(s_1, s_0, l)\}$
$\sigma\|s_0$	β	T	REDUCE	σ	β	T

Table 1: Transition actions under the arc-hybrid system. The first three actions are from dependency parsing; the last one is introduced for scene graph parsing.

RIGHT(l) transition yields an arc from the second top element of the stack to the top element of the stack, and then also removes the top element from the stack.

The following paragraphs describe how to select the correct transition action (and label l) in each step in order to generate a correct dependency tree.

BiLSTM Feature Extractor Let the word embeddings of a sentence s be $\mathbf{w}_1, \ldots, \mathbf{w}_n$. An LSTM cell is a parameterized function that takes as input $\mathbf{w}_t$, and updates its hidden states:

$$\text{LSTM cell} : (\mathbf{w}_t, \mathbf{h}_{t-1}) \rightarrow \mathbf{h}_t \quad (2)$$

As a result, an LSTM network, which simply applies the LSTM cell t times, is a parameterized function mapping a sequence of input vectors $\mathbf{w}_{1:t}$ to a sequence of output vectors $\mathbf{h}_{1:t}$. In our notation, we drop the intermediate vectors $\mathbf{h}_{1:t-1}$ and let $\text{LSTM}(\mathbf{w}_{1:t})$ represent $\mathbf{h}_t$.

A bidirectional LSTM, or BiLSTM for short, consists of two LSTMs: LSTM_F which reads the input sequence in the original order, and LSTM_B which reads it in reverse. Then

$$\text{BILSTM}(\mathbf{w}_{1:n}, i) = \text{LSTM}_F(\mathbf{w}_{1:i}) \circ \text{LSTM}_B(\mathbf{w}_{n:i}) \quad (3)$$

where $\circ$ denotes concatenation. Intuitively, the forward LSTM encodes information from the left side of the i-th word and the backward LSTM encodes information to its right, such that the vector $\mathbf{v}_i = \text{BILSTM}(\mathbf{w}_{1:n}, i)$ has the full sentence as context.

When predicting the transition action, the feature function $\phi(c)$ that summarizes the current configuration $c = (\sigma, \beta, T)$ is simply the concatenated BiLSTM vectors of the top three elements in the stack and the first element in the buffer:

$$\phi(c) = \mathbf{v}_{s_2} \circ \mathbf{v}_{s_1} \circ \mathbf{v}_{s_0} \circ \mathbf{v}_{b_0} \quad (4)$$

MLP Scoring Function The score of transition action y under the current configuration c is determined by a multi-layer perceptron with one hidden layer:

$$f(c, y) = MLP(\phi(c))[y] \quad (5)$$

where

$$MLP(x) = W_2 \cdot \tanh(W_1 \cdot x + b_1) + b_2 \quad (6)$$

Hinge Loss Function The training objective is to raise the scores of correct transitions above scores of incorrect ones. Therefore, at each step, we use a hinge loss defined as:

$$\mathcal{L} = \max(0, 1 - \max_{y^+ \in Y^+} f(c, y^+) + \max_{y^- \in Y \setminus Y^+} f(c, y^-)) \quad (7)$$

where Y is the set of possible transitions and Y^+ is the set of correct transitions at the current step. In each training step, the parser scores all possible transitions using Eqn. 5, incurs a loss using Eqn. 7, selects a following transition, and updates the configuration. Losses at individual steps are summed throughout the parsing of a sentence, and then parameters are updated using backpropagation.

In test time, we simply choose the transition action that yields the highest score at each step.

4.2 Customization

In order to generate scene graphs with dependency parsing, modification is necessary for at least two reasons. First, we need to redefine the label space of arcs so as to reflect the edge-centric representation of a scene graph. Second, not every word in the sentence will be (part of) a node in the scene graph (see Figure 2 for an example). In other words, some words in the sentence may not have a parent word, which violates the dependency parsing setting. We tackle these two challenges by redesigning the edge labels and expanding the set of transition actions.

Redesigning Edge Labels We define a total of five edge labels, so as to faithfully bridge the edge-centric view of scene graphs with dependency parsing models:

- CONT: This label is created for nodes whose label is a phrase. For example, the phrase "in front of" is a single relation node in the scene graph. By introducing the CONT label, we expect the parsing result to be either

$$\text{in} \xrightarrow{\text{CONT}} \text{front} \xrightarrow{\text{CONT}} \text{of} \tag{8}$$

  or

$$\text{in} \xleftarrow{\text{CONT}} \text{front} \xleftarrow{\text{CONT}} \text{of} \tag{9}$$

  where the direction of the arcs (left or right) is predefined by hand.

  The leftmost word under the right arc rule or the rightmost word under the left arc rule is called the *head* of the phrase. A single-word node does not need this CONT label, and the head is itself.

- ATTR: The arc label from the head of an object node to the head of an attribute node.

- SUBJ: The arc label from the head of an object node (subject) to the head of a relation node.

- OBJT: The arc label from the head of a relation node to the head of an object node (object).

- BEGN: The arc label from the ROOT index to all heads of object nodes without a parent.

Expanding Transition Actions With the three transition actions SHIFT, LEFT(l), RIGHT(l), we only drop an element (from the top of the stack) after it has already been associated with an arc. This design ensures that an arc is associated with every word. However, in our setting for scene graph generation, there may be no arc for some of the words, especially empty words.

Our solution is to augment the action set with a REDUCE action, that pops the stack *without* adding to the arc set (see Table 1). This action is often used in other transition-based dependency parsing systems (e.g. arc-eager (Nivre, 2004)). More recently, Hershcovich et al. (2017) and Buys and Blunsom (2017) also included this action when parsing sentences to graph structures.

Parser	F-score
Stanford (Schuster et al., 2015)	0.3549
SPICE (Anderson et al., 2016)	0.4469
Ours (left arc rule)	**0.4967**
Ours (right arc rule)	0.4952
Ours (all SYN)	0.4877
Ours (no SYN)	0.4538
Oracle	0.6985

Table 2: The F-scores (i.e. SPICE metric) between scene graphs parsed from region descriptions and ground truth region graphs on the intersection of Visual Genome (Krishna et al., 2017) and MS COCO (Lin et al., 2014) validation set.

We still minimize the loss function defined in Eqn. 7, except that now $|Y|$ increases from 3 to 4. During training, we impose the oracle to select the REDUCE action when it is in Y^+. In terms of loss function, we increment by 1 the loss incurred by the other 3 transition actions if REDUCE incurs zero loss.

5 Experiments

5.1 Implementation Details

We train and evaluate our scene graph parsing model on (a subset of) the Visual Genome (Krishna et al., 2017) dataset. Each image in Visual Genome contains a number of regions, and each region is annotated with both a region description and a region scene graph. Our training set is the intersection of Visual Genome and MS COCO (Lin et al., 2014) train2014 set, which contains a total of 34027 images/ 1070145 regions. We evaluate on the intersection of Visual Genome and MS COCO val2014 set, which contains a total of 17471 images/ 547795 regions.

In our experiments, the number of hidden units in BiLSTM is 256; the number of layers in BiLSTM is 2; the word embedding dimension is 200; the number of hidden units in MLP is 100. We use fixed learning rate 0.001 and Adam optimizer (Kingma and Ba, 2014) with epsilon 0.01. Training usually converges within 4 epochs.

We will release our code and trained model upon acceptance.

5.2 Quality of Parsed Scene Graphs

We use a slightly modified version of SPICE score (Anderson et al., 2016) to evaluate the quality of

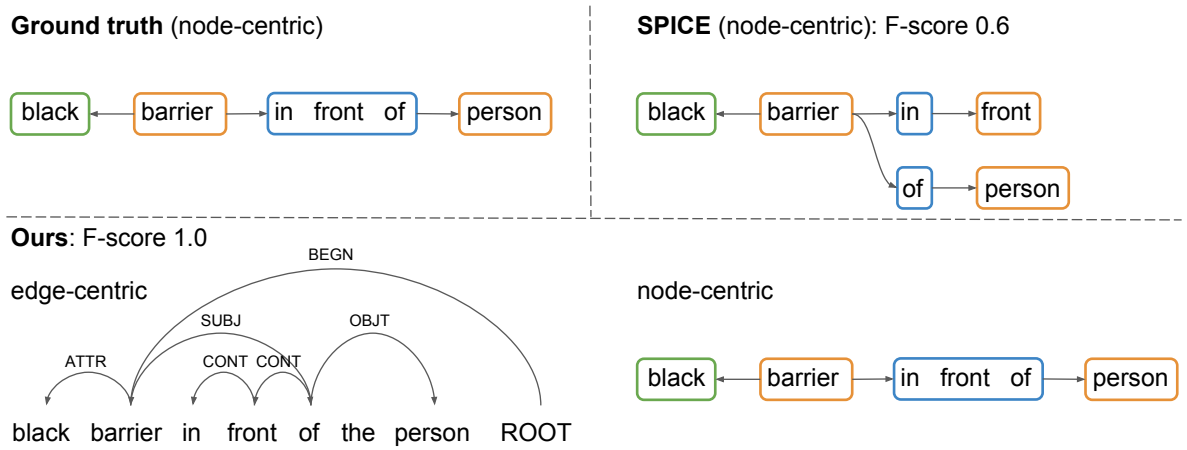


Figure 2: Scene graph parsing result of the sentence "black barrier in front of the person". In the node-centric graphs, orange represents object node, green represents attribute node, blue represents relation node.

	Stack	Buffer	Action
0		black barrier in front of the person ROOT	SHIFT
1	black	barrier in front of the person ROOT	LEFT(ATTR)
2		barrier in front of the person ROOT	SHIFT
3	barrier	in front of the person ROOT	SHIFT
4	barrier in	front of the person ROOT	LEFT(CONT)
5	barrier	front of the person ROOT	SHIFT
6	barrier front	of the person ROOT	LEFT(CONT)
7	barrier	of the person ROOT	SHIFT
8	barrier of	the person ROOT	SHIFT
9	barrier of the	person ROOT	REDUCE
10	barrier of	person ROOT	SHIFT
11	barrier of person	ROOT	RIGHT(OBJT)
12	barrier of	ROOT	RIGHT(SUBJ)
13	barrier	ROOT	LEFT(BEGN)
14		ROOT	

Figure 3: Intermediate actions taken by the trained dependency parser when parsing the sentence "black barrier in front of the person".

scene graph parsing. Specifically, for every region, we parse its description using a parser (e.g. the one used in SPICE or our customized dependency parser), and then calculate the F-score between the parsed graph and the ground truth region graph (see Section 3.2 of Anderson et al. (2016) for more details). Note that when SPICE calculates the F-score, a node in one graph could be matched to several nodes in the other, which is problematic. We fix this and enforce one-to-one matching when calculating the F-score. Finally, we report the average F-score across all regions.

Table 2 summarizes our results. We see that our customized dependency parsing model achieves an average F-score of 49.67%, which significantly outperforms the parser used in SPICE by 5 percent. This result shows that our customized dependency parser is very effective at learning from data, and generates more accurate scene graphs than the best previous approach.

Ablation Studies First, we study how the sentence-graph alignment procedure affects the final performance. Recall that our procedure involves two cycles, each with three steps. Of the six steps, synonym match (SYN) is only not used in the first step. We tried two more settings, where SYN is either used in all six steps or none of the six steps. We can see from Table 2 that the final

	Development set			Test set		
	R@5	**R@10**	**Med. rank**	**R@5**	**R@10**	**Med. rank**
(Schuster et al., 2015)	33.82%	45.58%	6	34.96%	45.68%	**5**
Ours	**36.69%**	**49.41%**	**4**	**36.70%**	**49.37%**	**5**

Table 3: Image retrieval results. We follow the same experiment setup as Schuster et al. (2015), except using a different scoring function when ranking images. Our parser consistently outperforms the Stanford Scene Graph Parser across evaluation metrics.

F-score drops in both cases, hence supporting the procedure that we chose.

Second, we study whether changing the direction of CONT arcs from pointing left to pointing right will make much difference. Table 2 shows that the two choices give very similar performance, suggesting that our dependency parser is robust to this design choice.

Finally, we report the oracle score, which is the similarity between the aligned graphs that we use during training and the ground truth graphs. The F-score is relatively high at 69.85%. This shows that improving the parser (about 20% margin) and improving the sentence-graph alignment (about 30% margin) are both promising directions for future research.

Qualitative Examples We provide one parsing example in Figure 2 and Figure 3. This is a sentence that is relatively simple, and the underlying scene graph includes two object nodes, one attribute node, and one compound word relation node. In parsing this sentence, all four actions listed in Table 1 are used (see Figure 3) to produce the edge-centric scene graph (bottom left of Figure 2), which is then trivially converted to the node-centric scene graph (bottom right of Figure 2).

5.3 Application in Image Retrieval

We test if the advantage of our parser can be propagated to computer vision tasks, such as image retrieval. We directly compare our parser with the Stanford Scene Graph Parser (Schuster et al., 2015) on the development set and test set of the image retrieval dataset used in Schuster et al. (2015) (not Visual Genome).

For every region in an image, there is a human-annotated region description and region scene graph. The queries are the region descriptions. If the region graph corresponding to the query is a subgraph of the complete graph of another image, then that image is added to the ground truth set for this query. All these are strictly following Schuster et al. (2015). However, since we did not obtain nor reproduce the CRF model used in Johnson et al. (2015) and Schuster et al. (2015), we used F-score similarity instead of the likelihood of the maximum a posteriori CRF solution when ranking the images based on the region descriptions. Therefore the numbers we report in Table 3 are not directly comparable with those reported in Schuster et al. (2015).

Our parser delivers better retrieval performance across all three evaluation metrics: recall@5, recall@10, and median rank. We also notice that the numbers in our retrieval setting are higher than those (even with oracle) in Schuster et al. (2015)'s retrieval setting. This strongly suggests that generating accurate scene graphs from images is a very promising research direction in image retrieval, and grounding parsed scene graphs to bounding box proposals without considering visual attributes/relationships (Johnson et al., 2015) is suboptimal.

6 Conclusion

In this paper, we offer a new perspective and solution to the task of parsing scene graphs from textual descriptions. We begin by moving the labels/types from the nodes to the edges and introducing the edge-centric view of scene graphs. We further show that the gap between edge-centric scene graphs and dependency parses can be filled with a careful redesign of label and action space. This motivates us to train a single, customized, end-to-end neural dependency parser for this task, as opposed to prior approaches that used generic dependency parsing followed by heuristics or simple classifier. We directly train our parser on a subset of Visual Genome (Krishna et al., 2017), without transferring any knowledge from Penn Treebank (Marcus et al., 1993) as previous works did.

The quality of our trained parser is validated in terms of both SPICE similarity to the ground truth graphs and recall rate/median rank when performing image retrieval.

We hope our paper can lead to more thoughts on the creative uses and extensions of existing NLP tools to tasks and datasets in other domains. In the future, we plan to tackle more computer vision tasks with this improved scene graph parsing technique in hand, such as image region grounding. We also plan to investigate parsing scene graph with cyclic structures, as well as whether/how the image information can help boost parsing quality.

Acknowledgments

The majority of this work was done when YSW and XZ were visiting Johns Hopkins University. We thank Peter Anderson, Sebastian Schuster, Ranjay Krishna, Tsung-Yi Lin for comments and help regarding the experiments. We also thank Tianze Shi, Dingquan Wang, Chu-Cheng Lin for discussion and feedback on the draft. This work was sponsored by the National Science Foundation Center for Brains, Minds, and Machines NSF CCF-1231216. CL also acknowledges an award from Snap Inc.

References

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. SPICE: semantic propositional image caption evaluation. In *ECCV*. Springer, volume 9909 of *Lecture Notes in Computer Science*, pages 382–398.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Neural module networks. In *CVPR*. IEEE Computer Society, pages 39–48.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: visual question answering. In *ICCV*. IEEE Computer Society, pages 2425–2433.

Jan Buys and Phil Blunsom. 2017. Robust incremental neural semantic graph parsing. In *ACL*. Association for Computational Linguistics, pages 1215–1226.

Angel X. Chang, Manolis Savva, and Christopher D. Manning. 2014. Learning spatial knowledge for text to 3d scene generation. In *EMNLP*. ACL, pages 2028–2038.

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

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*. ACL, pages 1724–1734.

James Cross and Liang Huang. 2016. Incremental parsing with minimal features using bi-directional LSTM. In *ACL (2)*. The Association for Computer Linguistics.

Bo Dai, Yuqi Zhang, and Dahua Lin. 2017. Detecting visual relationships with deep relational networks. In *CVPR*. IEEE Computer Society, pages 3298–3308.

Michael J. Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *WMT@ACL*. The Association for Computer Linguistics, pages 376–380.

Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Trevor Darrell, and Kate Saenko. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*. IEEE Computer Society, pages 2625–2634.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2014. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*. JMLR.org, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 647–655.

Timothy Dozat and Christopher D. Manning. 2016. Deep biaffine attention for neural dependency parsing. *CoRR* abs/1611.01734.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *ACL (1)*. The Association for Computer Linguistics, pages 334–343.

Jeffrey Flanigan, Sam Thomson, Jaime G. Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *ACL (1)*. The Association for Computer Linguistics, pages 1426–1436.

Andrea Frome, Gregory S. Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. 2013. Devise: A deep visual-semantic embedding model. In *NIPS*. pages 2121–2129.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2017. A transition-based directed acyclic graph parser for UCCA. In *ACL*. Association for Computational Linguistics, pages 1127–1138.

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

Ronghang Hu, Huazhe Xu, Marcus Rohrbach, Jiashi Feng, Kate Saenko, and Trevor Darrell. 2016. Natural language object retrieval. In *CVPR*. IEEE Computer Society, pages 4555–4564.

Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Fei-Fei Li. 2015. Image retrieval using scene graphs. In *CVPR*. IEEE Computer Society, pages 3668–3678.

Lukasz Kaiser, Aidan N. Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. 2017. One model to learn them all. *CoRR* abs/1706.05137.

Andrej Karpathy and Fei-Fei Li. 2015. Deep visual-semantic alignments for generating image descriptions. In *CVPR*. IEEE Computer Society, pages 3128–3137.

Arzoo Katiyar and Claire Cardie. 2017. Going out on a limb: Joint extraction of entity mentions and relations without dependency trees. In *ACL*. Association for Computational Linguistics, pages 917–928.

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

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL* 4:313–327.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: sequence-to-sequence models for parsing and generation. In *ACL (1)*. Association for Computational Linguistics, pages 146–157.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision* 123(1):32–73.

Jayant Krishnamurthy and Thomas Kollar. 2013. Jointly learning to parse and perceive: Connecting natural language to the physical world. *TACL* 1:193–206.

Sandra Kübler, Ryan T. McDonald, and Joakim Nivre. 2009. *Dependency Parsing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Yikang Li, Wanli Ouyang, Bolei Zhou, Kun Wang, and Xiaogang Wang. 2017. Scene graph generation from objects, phrases and caption regions. *CoRR* abs/1707.09700.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: common objects in context. In *ECCV*. Springer, volume 8693 of *Lecture Notes in Computer Science*, pages 740–755.

Chenxi Liu, Zhe Lin, Xiaohui Shen, Jimei Yang, Xin Lu, and Alan L. Yuille. 2017a. Recurrent multimodal interaction for referring image segmentation. In *ICCV*. IEEE Computer Society, pages 1280–1289.

Chenxi Liu, Junhua Mao, Fei Sha, and Alan L. Yuille. 2017b. Attention correctness in neural image captioning. In *AAAI*. AAAI Press, pages 4176–4182.

Cewu Lu, Ranjay Krishna, Michael S. Bernstein, and Fei-Fei Li. 2016. Visual relationship detection with language priors. In *ECCV*. Springer, volume 9905 of *Lecture Notes in Computer Science*, pages 852–869.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L. Yuille, and Kevin Murphy. 2016. Generation and comprehension of unambiguous object descriptions. In *CVPR*. IEEE Computer Society, pages 11–20.

Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, and Alan L. Yuille. 2014. Deep captioning with multimodal recurrent neural networks (m-rnn). *CoRR* abs/1412.6632.

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

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*. Association for Computational Linguistics, pages 50–57.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. ACL, pages 1532–1543.

Sebastian Schuster, Ranjay Krishna, Angel Chang, Li Fei-Fei, and Christopher D Manning. 2015. Generating semantically precise scene graphs from textual descriptions for improved image retrieval. In *Proceedings of the fourth workshop on vision and language*. volume 2.

Tianze Shi, Liang Huang, and Lillian Lee. 2017. Fast(er) exact decoding and global training for transition-based dependency parsing via a minimal feature set. In *EMNLP*. Association for Computational Linguistics, pages 12–23.

Damien Teney, Lingqiao Liu, and Anton van den Hengel. 2016. Graph-structured representations for visual question answering. *CoRR* abs/1609.05600.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2015. Order-embeddings of images and language. *CoRR* abs/1511.06361.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015. A transition-based algorithm for AMR parsing. In *HLT-NAACL*. The Association for Computational Linguistics, pages 366–375.

Keenon Werling, Gabor Angeli, and Christopher D. Manning. 2015. Robust subgraph generation improves abstract meaning representation parsing. In *ACL (1)*. The Association for Computer Linguistics, pages 982–991.

Danfei Xu, Yuke Zhu, Christopher B. Choy, and Li Fei-Fei. 2017. Scene graph generation by iterative message passing. In *CVPR*. IEEE Computer Society, pages 3097–3106.

Yuke Zhu, Oliver Groth, Michael S. Bernstein, and Li Fei-Fei. 2016. Visual7w: Grounded question answering in images. In *CVPR*. IEEE Computer Society, pages 4995–5004.

Learning Visually Grounded Sentence Representations

Douwe Kiela
Facebook AI Research
dkiela@fb.com

Alexis Conneau
Facebook AI Research
aconneau@fb.com

Allan Jabri[1]
UC Berkeley
ajabri@berkeley.edu

Maximilian Nickel
Facebook AI Research
maxn@fb.com

Abstract

We investigate grounded sentence representations, where we train a sentence encoder to predict the image features of a given caption—i.e., we try to "imagine" how a sentence would be depicted visually—and use the resultant features as sentence representations. We examine the quality of the learned representations on a variety of standard sentence representation quality benchmarks, showing improved performance for grounded models over non-grounded ones. In addition, we thoroughly analyze the extent to which grounding contributes to improved performance, and show that the system also learns improved word embeddings.

1 Introduction

Following the word embedding upheaval of the past few years, one of NLP's next big challenges has become the hunt for universal sentence representations: generic representations of sentence meaning that can be "plugged into" any kind of system or pipeline. Examples include Paragraph2Vec (Le and Mikolov, 2014), C-Phrase (Pham et al., 2015), SkipThought (Kiros et al., 2015) and FastSent (Hill et al., 2016a). These representations tend to be learned from large corpora in an unsupervised setting, much like word embeddings, and effectively "transferred" to the task at hand.

Purely text-based semantic models, which represent word meaning as a distribution over other words (Harris, 1954; Turney and Pantel, 2010; Clark, 2015), suffer from the grounding problem (Harnad, 1990). It has been shown that grounding leads to improved performance on a variety of word-level tasks (Baroni, 2016; Kiela, 2017). Unsupervised sentence representation models are often doubly exposed to the grounding problem, especially if they represent sentence meanings as a distribution over other sentences, as in SkipThought (Kiros et al., 2015).

Here, we examine whether grounding also leads to improved sentence representations. In short, the grounding problem is characterized by the lack of an association between symbols and external information. We address this problem by aligning text with paired visual data and hypothesize that sentence representations can be enriched with external information—i.e., grounded—by forcing them to capture visual semantics. We investigate the performance of these representations and the effect of grounding on a variety of semantic benchmarks.

There has been much recent interest in generating actual images from text (Goodfellow et al., 2014; van den Oord et al., 2016; Mansimov et al., 2016). Our method takes a slightly different approach: instead of predicting actual images, we train a deep recurrent neural network to predict the *latent feature representation* of images. That is, we are specifically interested in the semantic content of visual representations and how useful that information is for learning sentence representations. One can think of this as trying to imagine, or form a "mental picture", of a sentence's meaning (Chrupała et al., 2015). Much like a sentence's meaning in classical semantics is given by its model-theoretic ground truth (Tarski, 1944), our ground truth is provided by images.

Grounding is likely to be more useful for concrete words and sentences: a sentence such as "democracy is a political system" does not yield any coherent mental picture. In order to accommodate the fact that much of language is abstract, we take sentence representations obtained using text-only data (which are better for representing abstract meaning) and combine them with the grounded representations that our system learns (which are good for representing concrete meaning), leading to multi-modal sentence representations.

[1]Work done while at Facebook AI Research.

Proceedings of NAACL-HLT 2018, pages 408–418
New Orleans, Louisiana, June 1 - 6, 2018.

In what follows, we introduce a system for grounding sentence representations by learning to predict visual content. Although it is not the primary aim of this work, it is important to first examine how well this system achieves what it is trained to do, by evaluating on the COCO5K image and caption retrieval task. We then analyze the performance of grounded representations on a variety of sentence-level semantic transfer tasks, showing that grounding increases performance over text-only representations. We then investigate an important open question in multi-modal semantics: to what extent are improvements in semantic performance due to grounding, rather than to having more data or data from a different distribution? In the remainder, we analyze the role that concreteness plays in representation quality and show that our system learns grounded word embedding projections that outperform non-grounded ones. To the best of our knowledge, this is the first work to comprehensively study grounding for distributed sentence representations on such a wide set of semantic benchmark tasks.

2 Related work

Sentence representations Although there appears to be a consensus with regard to the methodology for learning word representations, this is much more of an open problem for sentence representations. Recent work has ranged from trying to learn to compose word embeddings (Le and Mikolov, 2014; Pham et al., 2015; Wieting et al., 2016; Arora et al., 2017), to neural architectures for predicting the previous and next sentences (Kiros et al., 2015) or learning representations via large-scale supervised tasks (Conneau et al., 2017). In particular, SkipThought (Kiros et al., 2015) led to an increased interest in learning sentence representations. Hill et al. (2016a) compare a wide selection of unsupervised and supervised methods, including a basic caption prediction system that is similar to ours. That study finds that "different learning methods are preferable for different intended applications", i.e., that the matter of optimal universal sentence representations is as of yet far from decided.

InferSent (Conneau et al., 2017) recently showed that supervised sentence representations can be of very high quality. Here, we learn grounded sentence representations in a supervised setting, combine them with standard unsupervised sentence representations, and show how grounding can help for a variety of sentence-level tasks.

Multi-modal semantics Language grounding in semantics has been motivated by evidence that human meaning representations are grounded in perceptual experience (Jones et al., 1991; Perfetti, 1998; Andrews et al., 2009; Riordan and Jones, 2011). That is, despite ample evidence of humans representing meaning with respect to an external environment and sensorimotor experience (Barsalou, 2008; Louwerse, 2008), standard semantic models rely solely on textual data. This gives rise to an infinite regress in text-only semantic representations, i.e., words are defined in terms of other words, *ad infinitum*.

The field of multi-modal semantics, which aims to address this issue by enriching textual representations with information from other modalities, has mostly been concerned with word representations (Bruni et al., 2014; Baroni, 2016; Kiela, 2017, and references therein). Learning multi-modal representations that ground text-only representations has been shown to improve performance on a variety of core NLP tasks. This work is most closely related to that of Chrupała et al. (2015), who also aim to ground language by relating images to captions: here, we additionally address abstract sentence meaning; have a different architecture, loss function and fusion strategy; and explicitly focus on grounded universal sentence representations.

Bridging vision and language There is a large body of work that involves jointly embedding images and text, at the word level (Frome et al., 2013; Joulin et al., 2016), the phrase level (Karpathy et al., 2014; Li et al., 2016), and the sentence level (Karpathy and Fei-Fei, 2015; Klein et al., 2015; Kiros et al., 2015; Chen and Zitnick, 2015; Reed et al., 2016). Our model similarly learns to map sentence representations to be consistent with a visual semantic space, and we focus on studying how these grounded text representations transfer to NLP tasks.

Moreover, there has been a lot of work in recent years on the task of image caption generation (Bernardi et al., 2016; Vinyals et al., 2015; Mao et al., 2015; Fang et al., 2015). Here, we do the opposite: we predict the correct image (features) from the caption, rather than the caption from the image (features). Similar ideas were recently successfully applied to multi-modal machine translation

(Elliott and Kádár, 2017; Gella et al., 2017; Lee et al., 2017). Recently, Das et al. (2017) trained dialogue agents to communicate about images, trying to predict image features as well.

3 Approach

In the following, let $\mathcal{D} = \{(I_k, C_k)\}_{k=1}^{N}$ be a dataset where each image I_k is associated with one or more captions $C_k = \{C_1, \ldots, C_{|C|_k}\}$. A prominent example of such a dataset is COCO (Lin et al., 2014), which consists of images with up to 5 corresponding captions for each image. The objective of our approach is to encode a given sentence, i.e., a caption C, and learn to ground it in the corresponding image I. To encode the sentence, we train a bidirectional LSTM (BiLSTM) on the caption, where the input is a sequence of projected word embeddings. We combine the final left-to-right and right-to-left hidden states of the LSTM and take the element-wise maximum to obtain a sentence encoding. We then examine three distinct methods for grounding the sentence encoding.

In the first method, we try to predict the image features (Cap2Img). That is, we learn to map the caption to the same space as the image features that represent the correct image. We call this strong perceptual grounding, where we take the visual input directly into account.

An alternative method is to exploit the fact that one image in COCO has multiple captions (Cap2Cap), and to learn to predict which other captions are valid descriptions of the same image. This approach is strictly speaking not perceptually grounded, but exploits the fact that there is an implicit association between the captions and the shared underlying image, and so could be considered a weaker version of grounding.

Finally, we experiment with a model that optimizes both these objectives jointly: that is, we predict both images and alternative captions for the same image (Cap2Both). Thus, Cap2Both incorporates both strong perceptual and weak implicit grounding. Please see Figure 1 for an illustration of the various models. In what follows, we discuss them in more technical detail.

3.1 Bidirectional LSTM

To learn sentence representations, we employ a bidirectional LSTM architecture. In particular, let $x = (x_1, \ldots, x_T)$ be an input sequence where each word is represented via an embedding $\mathbf{x}_t \in \mathbb{R}^n$. Using a standard LSTM (Hochreiter and Schmidhuber, 1997), the hidden state at time t, denoted $\mathbf{h}_t \in \mathbb{R}^m$, is computed via

$$\mathbf{h}_{t+1}, \mathbf{c}_{t+1} = \text{LSTM}(\mathbf{x}_t, \mathbf{h}_t, \mathbf{c}_t \mid \Theta)$$

where $\mathbf{c}_t$ denotes the cell state of the LSTM and where Θ denotes its parameters.

To exploit contextual information in both input directions, we process input sentences using a bidirectional LSTM, that reads an input sequence in both normal and reverse order. In particular, for an input sequence x of length T, we compute the hidden state at time t, $\mathbf{h}_t \in \mathbb{R}^{2m}$ via

$$\mathbf{h}^f_{t+1} = \text{LSTM}(\mathbf{x}_t, \mathbf{h}^f_t, \mathbf{c}^f_t \mid \Theta^f)$$
$$\mathbf{h}^b_{t+1} = \text{LSTM}(\mathbf{x}_{T-t}, \mathbf{h}^b_t, \mathbf{c}^b_t \mid \Theta^b)$$

Here, the two LSTMs process x in a forward and a backward order, respectively. We subsequently use $\max : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ to combine them into their element-wise maximum, yielding the representation of a caption after it has been processed with the BiLSTM:

$$\mathbf{h}_T = \max(\mathbf{h}^f_t, \mathbf{h}^b_t)$$

We use GloVe vectors (Pennington et al., 2014) for our word embeddings. The embeddings are kept fixed during training, which allows a trained sentence encoder to transfer to tasks (and a vocabulary) that it has not yet seen, provided GloVe embeddings are available. Since GloVe representations are not tuned to represent grounded information, we learn a global transformation of GloVe space to grounded word space. Specifically, let $\overline{\mathbf{x}} \in \mathbb{R}^n$ be the original GloVe embeddings. We then learn a linear map $U \in \mathbb{R}^{n \times n}$ such that $\mathbf{x} = U\overline{\mathbf{x}}$ and use $\mathbf{x}$ as input to the BiLSTM. The linear map U and the BiLSTM are trained jointly.

3.2 Cap2Img

Let $\mathbf{v} \in \mathbb{R}^I$ be the latent representation of an image (e.g.the final layer of a ResNet). To ground captions in the images that they describe, we map $\mathbf{h}_T$ into the latent space of image representations such that their similarity is maximized. In other words, we aim to predict the latent features of an image from its caption. The mapping of caption to image space is performed via a series of projections

$$\mathbf{p}_0 = \mathbf{h}_T$$
$$\mathbf{p}_{\ell+1} = \psi(P_\ell \mathbf{p}_\ell)$$

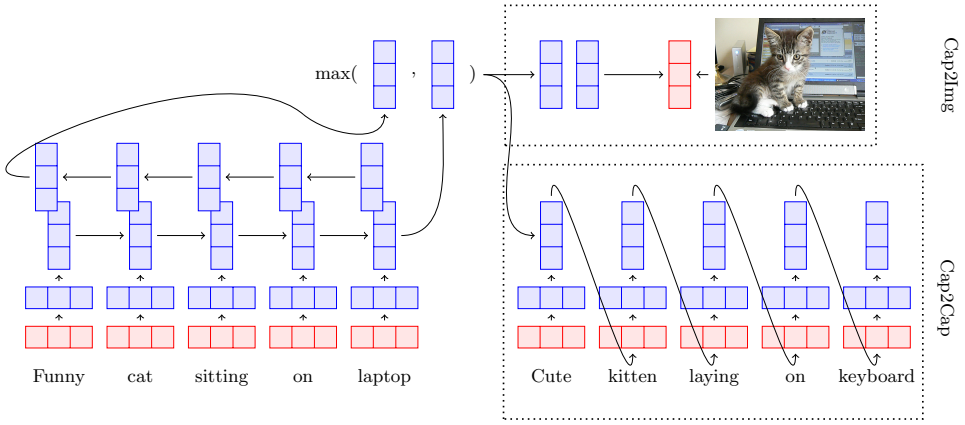


Figure 1: Model architecture: predicting either an image (Cap2Img), an alternative caption (Cap2Cap), or both at the same time (Cap2Both).

where ψ denotes a non-linearity such as ReLUs or tanh.

By jointly training the BiLSTM with these latent projections, we can then ground the language model in its visual counterpart. In particular, let $\Theta = \Theta_{\text{BiLSTM}} \cup \{P_\ell\}_{\ell=1}^{L}$ be the parameters of the BiLSTM as well as the projection layers. We then minimize the following ranking loss:

$$\mathcal{L}_{\text{C2I}}(\Theta) = \sum_{(I,C)\in\mathcal{D}} f_{\text{rank}}(I, C) + f_{\text{rank}}(C, I) \quad (1)$$

where

$$f_{\text{rank}}(a, b) = \sum_{b'\in\mathcal{N}_a} [\gamma - \text{sim}(a, b) + \text{sim}(a, b')]_+$$

where $[x]_+ = \max(0, x)$ denotes the threshold function at zero and γ defines the margin. Furthermore, $\mathcal{N}_a$ denotes the set of negative samples for an image or caption and $\text{sim}(\cdot, \cdot)$ denotes a similarity measure between vectors. In the following, we employ the cosine similarity, i.e.,

$$\text{sim}(a, b) = \frac{\langle \mathbf{a}, \mathbf{b} \rangle}{\|\mathbf{a}\|\|\mathbf{b}\|}.$$

Although this loss is not smooth at zero, it can be trained end-to-end using subgradient methods. Compared to e.g. an l_2 regression loss, Equation (1) is less susceptible to error incurred by subspaces of the visual representation that are irrelevant to the high level visual semantics. Empirically, we found it to be more robust to overfitting.

3.3 Cap2Cap

Let $x = (x_1, \ldots, x_T)$, $y = (y_1, \ldots, y_S)$ be a caption pair that describes the same image. To learn weakly grounded representations, we employ a standard sequence-to-sequence model (Sutskever et al., 2014), whose task is to predict y from x. As in the Cap2Cap model, let $\mathbf{h}_T$ be the representation of the input sentence after it has been processed with a BiLSTM. We then model the joint probability of y given x as

$$p(y \mid x) = \prod_{s=1}^{S} p(y_s \mid \mathbf{h}_T, y_1, \ldots, y_{s-1}, \Theta).$$

To model the conditional probability of y_s we use the usual multiclass classification approach over the vocabulary of the corpus $\mathcal{V}$ such that

$$p(y_s = k \mid \mathbf{h}_T, y_1, \ldots, y_{s-1}, \Theta) = \frac{e^{\langle \mathbf{v}_k, \mathbf{y}_s \rangle}}{\sum_{j=1}^{|\mathcal{V}|} e^{\langle \mathbf{v}_j, \mathbf{y}_s \rangle}}.$$

Here, $\mathbf{y}_s = \psi(W_V \mathbf{g}_s + \mathbf{b})$ and $\mathbf{g}_s$ is hidden state of the decoder LSTM at time s.

To learn the model parameters, we minimize the negative log-likelihood over all caption pairs, i.e.,

$$\mathcal{L}_{\text{C2C}}(\theta) = -\sum_{x,y\in\mathcal{D}} \sum_{s=1}^{|y|} \log p(y_s \mid \mathbf{h}_T, y_1, \ldots, y_{s-1}, \Theta).$$

3.4 Cap2Both

Finally, we also integrate both concepts of grounding into a joint model, where we optimize the following loss function:

$$\mathcal{L}_{\text{C2B}}(\Theta) = \mathcal{L}_{C2I}(\Theta) + \mathcal{L}_{C2C}(\Theta).$$

3.5 Grounded universal representations

On their own, features from this system are likely to suffer from the fact that training on COCO introduces biases: aside from the inherent dataset bias in COCO itself, the system will only have coverage for concrete concepts. COCO is also a much smaller dataset than e.g. the Toronto Books Corpus often used in purely text-based methods (Kiros et al., 2015). As such, grounded representations are potentially less "universal" than text-based alternatives, which also cover abstract concepts.

There is evidence that meaning is dually coded in the human brain: while abstract concepts are processed in linguistic areas, concrete concepts are processed in both linguistic and visual areas (Paivio, 1990). Anderson et al. (2017) recently corroborated this hypothesis using semantic representations and fMRI studies. In our case, we want to be able to accommodate concrete sentence meanings, for which our vision-centric system is likely to help; as well as abstract sentence meanings, where trying to "imagine" what "democracy is a political system" might look like will probably only introduce noise.

Hence, we optionally complement our systems' representations with more abstract universal sentence representations trained on language-only data (specifically, the Toronto Books Corpus). Although it would be interesting to examine multitask scenarios where these representations are jointly learned, we leave this for future work. Here, instead, we combine grounded and language-only representations using simple concatenation, i.e., $r_{gs} = r_{grounded} \mid\mid r_{ling-only}$. Concatenation has been proven to be a strong and straightforward mid-level multi-modal fusion method, previously explored in multi-modal semantics for word representations (Bruni et al., 2014; Kiela and Bottou, 2014). We call the combined system GroundSent (GS), and distinguish between sentences perceptually grounded in images (GroundSent-Img), weakly grounded in captions (GroundSent-Cap) or grounded in both (GroundSent-Both).

3.6 Implementation details

We use 300-dimensional GloVe (Pennington et al., 2014) embeddings, trained on WebCrawl, for the initial word representations and optimize using Adam (Kingma and Ba, 2015). We use ELU (Clevert et al., 2016) for the non-linearity in projection layers, set dropout to 0.5 and use a dimensionality of 1024 for the LSTM. The network was initialized with orthogonal matrices for the recurrent layers (Saxe et al., 2014) and He initialization (He et al., 2015) for all other layers. The learning rate and margin were tuned on the validation set using grid search.

4 Data, evaluation and comparison

We use the same COCO splits as Karpathy and Fei-Fei (2015) for training (113,287 images), validation (5000 images) and testing (5000 images). Image features for COCO were obtained by transferring the final layer from a ResNet-101 (He et al., 2016) trained on ImageNet (ILSVRC 2015).

4.1 Transfer tasks

We are specifically interested in how well (grounded) universal sentence representations transfer to different tasks. To evaluate this, we perform experiments for a variety of tasks. In all cases, we compare against layer-normalized Skip-Thought vectors, a well-known high-performing sentence encoding method (Ba et al., 2016). To ensure that we use the exact same evaluations, with identical hyperparameters and settings, we evaluate all systems with the same evaluation pipeline, namely SentEval (Conneau and Kiela, 2018)[2]. Following previous work in the field, the idea is to take universal sentence representations and to learn a simple classifier on top for each of the transfer tasks—the higher the quality of the sentence representation, the better the performance on these transfer tasks should be.

4.1.1 Semantic classification

We evaluate on the following well-known and widely used evaluations: movie review sentiment (MR) (Pang and Lee, 2005), product reviews (CR) (Hu and Liu, 2004), subjectivity classification (SUBJ) (Pang and Lee, 2004), opinion polarity (MPQA) (Wiebe et al., 2005), paraphrase identification (MSRP) (Dolan et al., 2004) and sentiment classification (SST, binary version) (Socher et al., 2013). Accuracy is measured in all cases, except for MRPC, which measures accuracy and the F1-score.

[2]See https://github.com/facebookresearch/SentEval. The aim of SentEval is to encompass a comprehensive set of benchmarks that has been loosely established in the research community as the standard for evaluating sentence representations.

	COCO5K									
	Caption Retrieval					Image Retrieval				
Model	**R@1**	**R@5**	**R@10**	**MEDR**	**MR**	**R@1**	**R@5**	**R@10**	**MEDR**	**MR**
DVSA	11.8	32.5	45.4	12.2	NA	8.9	24.9	36.3	19.5	NA
FV	17.3	39.0	50.2	10.0	46.4	10.8	28.3	40.1	17.0	49.3
OE	23.3	NA	65.0	5.0	24.4	18.0	NA	57.6	7.0	35.9
Cap2Both	19.4	45.0	59.4	7.0	26.5	11.7	32.6	46.4	12.0	41.7
Cap2Img	27.1	55.6	70.0	4.0	19.2	17.1	43.0	57.3	8.0	36.6

Table 1: Retrieval (higher is better) results on COCO, plus median rank (MEDR) and mean rank (MR) (lower is better). Note that while this work underwent review, better methods have been published, most notably VSE++ (Faghri et al., 2017).

Model	**MR**	**CR**	**SUBJ**	**MPQA**	**MRPC**	**SST**	**SNLI**	**SICK**
ST-LN	78.1	80.1	92.7	88.0	69.6/81.2	82.9	73.8	78.5
GroundSent-Cap	79.9	81.4	93.1	88.9	72.9/82.2	85.0	75.5	79.7
GroundSent-Img	79.1	80.8	93.1	89.0	71.9/81.4	86.1	76.1	82.2
GroundSent-Both	79.6	81.7	93.4	89.4	72.7/82.5	84.8	76.1	81.6

Table 2: Accuracy results on sentence classification and entailment tasks.

4.1.2 Entailment

Recent years have seen an increased interest in entailment classification as an appropriate evaluation of sentence representation quality. We evaluate representations on two well-known entailment, or natural language inference, datasets: the large-scale SNLI dataset (Bowman et al., 2015) and the SICK dataset (Marelli et al., 2014).

4.2 Implementational details

We implement a simple logistic regression on top of the sentence representation. In the cases of SNLI and SICK, as is the standard for these datasets, the representations for the individual sentences u and v are combined by using $\langle \mathbf{u}, \mathbf{v}, \mathbf{u} * \mathbf{v}, |\mathbf{u} - \mathbf{v}| \rangle$ as the input features. We tune the seed and an l_2 penalty on the validation sets for each, and train using Adam (Kingma and Ba, 2015), with a learning rate of 0.001 and a batch size of 32.

5 Results

Although it is not the primary aim of this work to learn a state-of-the-art image and caption retrieval system, it is important to first establish the capability of our system to do what it is trained to do. Table 1 shows the results on the COCO5K caption and image retrieval tasks for the two models that predict image features. We compare our system against several well-known approaches, namely Deep Visual-Semantic Alignments (DVSA) (Karpathy and Fei-Fei, 2015), Fisher Vectors (FV) (Klein et al., 2015) and Order Embeddings (OE) (Vendrov et al., 2015). As the results show, Cap2Img performs very well on this task, outperforming the compared models on caption retrieval and being very close to order embeddings on image retrieval[3]. The fact that the system outperforms Order Embeddings on caption retrieval suggests that it has a better sentence encoder. Cap2Both does not work as well on this task as the image-only case, probably because interference from the language signal makes the problem harder to optimize. The results indicate that the system has learned to predict image features from captions, and captions from images, at a level exceeding or close to the state-of-the-art on this task.

5.1 Transfer task performance

Having established that we can learn high-quality grounded sentence encodings, the core question we now wish to examine is how well grounded sentence representations transfer. In this section, we combine our grounded features with the

[3]In fact, we found that we can achieve better performance on this task by reducing the dimensionality of the encoder. A lower dimensionality in the encoder also reduces the transferability of the features, unfortunately, so we leave a more thorough investigation of this phenomenon for future work.

Model	MR	CR	SUBJ	MPQA	MRPC	SST	SNLI	SICK
STb-1024	70.3	68.0	87.5	85.5	69.7/80.6	78.3	67.3	76.6
STb-2048	73.1	**75.7**	88.3	86.5	71.6/**81.7**	79.0	71.0	78.8
2×STb-1024	71.4	74.7	88.2	86.6	71.3/80.7	75.8	69.4	78.3
Cap2Cap	71.4	74.7	86.7	**86.7**	70.3/79.8	76.1	68.5	78.2
Cap2Img	72.1	75.5	86.9	86.0	**72.3**/81.1	77.7	71.4	81.2
Cap2Both	71.6	74.4	86.5	85.5	71.4/79.5	78.5	71.3	**81.7**
GroundSent-Cap	73.1	73.0	**88.6**	86.6	70.8/81.2	79.4	70.7	79.1
GroundSent-Img	72.5	74.9	88.4	85.7	71.3/81.2	79.4	70.5	79.7
GroundSent-Both	**73.3**	75.2	87.5	86.6	69.9/79.9	**80.3**	**72.0**	78.1

Table 3: Thorough investigation of the contribution of grounding, ensuring equal number of components and identical architectures, on the variety of sentence-level semantic benchmark tasks. STb=SkipThought-like model with bidirectional LSTM+max. 2×STb-1024=ensemble of 2 different STb models with different initializations. GroundSent is STb-1024+Cap2Cap/Img/Both. We find that performance improvements are sometimes due to having more parameters, but in most cases due to grounding.

high-quality layer-normalized SkipThought representations of Ba et al. (2016), leading to multi-modal sentence representations as described in Section 3.5. That is, we concatenate Cap2Cap, Cap2Img or Cap2Both and Skip-Thought with Layer Normalization (ST-LN) representations, yielding GroundSent-Cap, GroundSent-Img and GroundSent-Both representations, respectively. We report performance of ST-LN using SentEval, which led to slightly different numbers than what is reported in their paper[4].

Table 2 shows the results for the semantic classification and entailment tasks. Note that all systems use the exact same evaluation pipeline, which makes them directly comparable. We can see that in all cases, grounding increases the performance. The question of which type of grounding works best is more difficult: generally, grounding with Cap2Cap and Cap2Both appears to do slightly better on most tasks, but on e.g. SST, Cap2Img works better. The entailment task results (SNLI and SICK in Table 2) show a similar picture: in all cases grounding improves performance.

It is important to note that, in this work, we are not necessarily concerned with replacing the state-of-the-art on these tasks: there are systems that perform better. We are primarily interested in whether grounding helps relative to text-only baselines. We find that it does.

[4]This is probably due to different seeds, optimization methods and other minor implementational details that differ between the original work and SentEval.

5.2 The contribution of grounding

An important open question is whether the increase in performance in multi-modal semantic models is due to qualitatively different information from *grounding*, or simply due to the fact that we have *more parameters* or *data from a different distribution*. In order to examine this, we implement a SkipThought-like model that also uses a bidirectional LSTM with element-wise max on the final hidden layer (henceforth referred to as STb). This model is architecturally identical to the sentence encoder used before: it can be thought of as Cap2Cap, but where the objective is not to predict an alternative caption, but to predict the previous and next sentence in the Toronto Books Corpus, just like SkipThought (Kiros et al., 2015).

We train a 1024-dimensional and 2048-dimensional STb model (for one full iteration, with all other hyperparameters identical to Cap2Cap) to compare against: if grounding improves results because it introduces qualitatively different information, rather than just from having more parameters (i.e., a higher embedding dimensionality), we should expect the multi-modal GroundSent models to perform better not only than STb-1024, but also than STb-2048, which has the same number of parameters (recall that GroundSent models are combinations of grounded and linguistic-only representations). In addition, we compare against an "ensemble" of two different STb-1024 models (i.e., a concatenation of two separately trained STb-1024), to check that we are not (just) observing an ensemble effect.

Dataset	Concreteness
MR	2.3737 ± 0.965
CR	2.4714 ± 1.025
SUBJ	2.4510 ± 1.007
MPQA	2.3158 ± 0.834
MRPC	2.5086 ± 0.987
SST	2.7471 ± 1.142
SNLI	3.1867 ± 1.309
SICK	3.1282 ± 1.372

Table 4: Mean and variance of dataset concreteness, over all words in the datasets.

As Table 3 shows, a more nuanced picture emerges in this comparison: grounding helps more for some datasets than for others. Grounded models outperform the STb-1024 model (which uses much more data—the Toronto Books Corpus is much larger than COCO) in all cases, often already without concatenating the textual modality. The ensemble of two STb-1024 models performs better than the individual one, and so does the higher-dimensional one. In the cases of CR and MRPC (F1), it appears that improved performance is due to having more data or ensemble effects. For the other datasets, grounding clearly yields better results. These results indicate that grounding does indeed capture qualitatively different information, yielding better universal sentence representations.

6 Discussion

There are a few other important questions to investigate. The average abstractness or concreteness of the evaluation datasets may have a large impact on performance. In addition, word embeddings from the learned projection from GloVe input embeddings, which now provides a generic word-embedding grounding method even for words that are not present in the image-caption training data, can be examined.

6.1 Concreteness

As we have seen, performance across datasets and models can vary substantially. A dataset's concreteness plays an important role in the relative merit of applying grounding: a dataset consisting mostly of abstract words is less likely to benefit from grounding than one that uses mostly concrete words. In order to examine this effect, we calculate the average concreteness of the evaluation datasets used in this study. Table 4 shows the average human-annotated concreteness ratings for all words (where available) in each dataset. The ratings were obtained by Brysbaert et al. (2014) in a large-scale study, yielding scores for 40,000 English words.

We observe that the two entailment datasets are more concrete, which is due to the fact that the premises are derived from caption datasets (Flickr30K in the case of SNLI; Flickr8K and video captions in the case of SICK). This explains why grounding can clearly be seen to help in these cases. For the semantic classification tasks, the more concrete datasets are MRPC and SST. The picture is less clear for the first, but in SST we see that the grounded representations definitely do work better. Concreteness values make it easier to analyze performance, but are apparently not always direct indicators of improvements with grounding.

Model	MEN	SimLex	RW	W353
GloVe	0.805	0.408	0.451	0.738
Cap2Both	0.819	0.467	0.487	0.712
Cap2Img	0.845	0.515	0.523	0.753

Table 5: Spearman ρ_s correlation on four standard semantic similarity evaluation benchmarks.

6.2 Grounded word embeddings

Our models contain a projection layer that maps the GloVe word embeddings that they receive as inputs to a different embedding space. There has been a lot of interest in grounded word representations in recent years, so it is interesting to examine what kind of word representations our models learn. We omit Cap2Cap for reasons of space (it performs similarly to Cap2Both). As shown in Table 5, the grounded word projections that our network learns yield higher-quality word embeddings on four standard lexical semantic similarity benchmarks: MEN (Bruni et al., 2014), SimLex-999 (Hill et al., 2016b), Rare Words (Luong et al., 2013) and WordSim-353 (Finkelstein et al., 2001).

7 Conclusion

We have investigated grounding for universal sentence representations. We achieved good performance on caption and image retrieval tasks on the large-scale COCO dataset. We subsequently showed how the sentence encodings that the sys-

tem learns can be transferred to various NLP tasks, and that grounded universal sentence representations lead to improved performance. We analyzed the source of improvements from grounding, and showed that the increased performance appears to be due to the introduction of qualitatively different information (i.e., grounding), rather than simply having more parameters or applying ensemble methods. Lastly, we showed that our systems learned high-quality grounded word embeddings that outperform non-grounded ones on standard semantic similarity benchmarks. It could well be that our methods are even more suited for more concrete tasks, such as visual question answering, visual storytelling, or image-grounded dialogue—an avenue worth exploring in future work. In addition, it would be interesting to explore multi-task learning for sentence representations where one of the tasks involves grounding.

Acknowledgments

We thank the anonymous reviewers for their helpful comments and suggestions. Part of Fig. 1 is licensed from dougwoods/CC-BY-2.0/flickr.com/photos/deerwooduk/682390157.

References

Andrew J. Anderson, Douwe Kiela, Stephen Clark, and Massimo Poesio. 2017. Visually grounded and textual semantic models differentially decode brain activity associated with concrete and abstract nouns. *Transactions of the Association for Computational Linguistics* 5.

Mark Andrews, Gabriella Vigliocco, and David Vinson. 2009. Integrating experiential and distributional data to learn semantic representations. *Psychological review* 116(3):463.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *International Conference on Learning Representations (ICLR)*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* .

Marco Baroni. 2016. Grounding distributional semantics in the visual world. *Language and Linguistics Compass* 10(1):3–13.

Lawrence W. Barsalou. 2008. Grounded cognition. *Annual Review of Psychology* 59(1):617–645.

Raffaella Bernardi, Ruket Cakici, Desmond Elliott, Aykut Erdem, Erkut Erdem, Nazli Ikizler-Cinbis, Frank Keller, Adrian Muscat, and Barbara Plank. 2016. Automatic description generation from images: A survey of models, datasets, and evaluation measures. *Journal of Artificial Intelligence Research (JAIR)* pages 409–442.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of EMNLP*.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artifical Intelligence Research* 49:1–47.

Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. 2014. Concreteness ratings for 40 thousand generally known english word lemmas. *Behavior research methods* 46(3):904–911.

Xinlei Chen and Lawrence C Zitnick. 2015. Mind's eye: A recurrent visual representation for image caption generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 2422–2431.

Grzegorz Chrupała, Ákos Kádár, and Afra Alishahi. 2015. Learning language through pictures. In *Proceedings of ACL*.

Stephen Clark. 2015. Vector Space Models of Lexical Meaning. In Shalom Lappin and Chris Fox, editors, *Handbook of Contemporary Semantic Theory*, Wiley-Blackwell, Oxford, chapter 16.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2016. Fast and accurate deep network learning by exponential linear units (ELUs). In *International Conference on Learning Representations (ICLR)*.

Alexis Conneau and Douwe Kiela. 2018. Senteval: An evaluation toolkit for universal sentence representations. In *Proceedings of LREC*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of EMNLP*. Copenhagen, Denmark.

Abhishek Das, Satwik Kottur, José M. F. Moura, Stefan Lee, and Dhruv Batra. 2017. Learning cooperative visual dialog agents with deep reinforcement learning. In *Proceedings of CVPR*.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of ACL*. page 350.

Desmond Elliott and Ákos Kádár. 2017. Imagination improves multimodal translation. *arXiv preprint arXiv:1705.04350* .

Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. 2017. Vse++: Improved visual-semantic embeddings. *arXiv preprint arXiv:1707.05612* .

H. Fang, S. Gupta, F.N. Iandola, R. Srivastava, L. Deng, P. Dollar, J. Gao, X. He, M. Mitchell, J.C. Platt, C.L. Zitnick, and G. Zweig. 2015. From captions to visual concepts and back. In *CVPR*.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web (WWW)*. ACM, pages 406–414.

A. Frome, G. Corrado, J. Shlens, S. Bengio, J. Dean, and T. Mikolov. 2013. Devise: A deep visual-semantic embedding model. In *NIPS*.

Spandana Gella, Rico Sennrich, Frank Keller, and Mirella Lapata. 2017. Image pivoting for learning multilingual multimodal representations .

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of NIPS*.

Stevan Harnad. 1990. The symbol grounding problem. *Physica D* 42:335–346.

Z. Harris. 1954. Distributional Structure. *Word* 10(23):146—162.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision (CVPR)*. pages 1026–1034.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. pages 770–778.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016a. Learning distributed representations of sentences from unlabelled data. In *Proceedings of NAACL*.

Felix Hill, Roi Reichart, and Anna Korhonen. 2016b. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics* .

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

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of SIGKDD*. pages 168–177.

Susan S Jones, Linda B Smith, and Barbara Landau. 1991. Object properties and knowledge in early lexical learning. *Child development* 62(3):499–516.

A. Joulin, L.J.P. van der Maaten, A. Jabri, and N. Vasilache. 2016. Learning visual features from large weakly supervised data. In *ECCV*.

A. Karpathy, A. Joulin, and L. Fei-Fei. 2014. Deep fragment embeddings for bidirectional image sentence mapping. In *Proceedings of NIPS*.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. pages 3128–3137.

Douwe Kiela. 2017. Deep embodiment: grounding semantics in perceptual modalities (PhD thesis). Technical Report UCAM-CL-TR-899, University of Cambridge, Computer Laboratory.

Douwe Kiela and Léon Bottou. 2014. Learning Image Embeddings using Convolutional Neural Networks for Improved Multi-Modal Semantics. In *Proceedings of EMNLP*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In *Proceedings of NIPS*.

Benjamin Klein, Guy Lev, Gil Sadeh, and Lior Wolf. 2015. Associating neural word embeddings with deep image representations using fisher vectors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. pages 4437–4446.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of ICML*.

Jason Lee, Kyunghyun Cho, Jason Weston, and Douwe Kiela. 2017. Emergent translation in multi-agent communication. *CoRR* abs/1710.06922.

A. Li, A. Jabri, A. Joulin, and L.J.P. van der Maaten. 2016. Learning visual n-grams from web data. In *arxiv*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*. Springer, pages 740–755.

Max M. Louwerse. 2008. Symbol interdependency in symbolic and embodied cognition. *Topics in Cognitive Science* 59(1):617–645.

Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of CoNLL*. pages 104–113.

Elman Mansimov, Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. 2016. Generating images from captions with attention. In *International Conference on Learning Representations (ICLR)*.

J. Mao, W. Xu, Y. Yang, J. Wang, and A.L. Yuille. 2015. Deep captioning with multimodal recurrent neural networks. In *Proceedings of ICLR*.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Reffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of LREC*.

Allan Paivio. 1990. *Mental representations: A dual coding approach*. Oxford University Press.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of ACL*. page 271.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of ACL*. pages 115–124.

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

Charles A Perfetti. 1998. The limits of co-occurrence: Tools and theories in language research. *Discourse Processes* 25(2&3):363—377.

Nghia The Pham, German Kruszewski, Angeliki Lazaridou, and Marco Baroni. 2015. Jointly optimizing word representations for lexical and sentential tasks with the c-phrase model. In *Proceedings of ACL*.

S. Reed, Z. Akata, H. Lee, and B. Schiele. 2016. Learning deep representations of fine-grained visual descriptions. In *Proceedings of CVPR*.

Brian Riordan and Michael N Jones. 2011. Redundancy in perceptual and linguistic experience: Comparing feature-based and distributional models of semantic representation. *Topics in Cognitive Science* 3(2):303–345.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. 2014. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *International Conference on Learning Representations (ICLR)*.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, Christopher Potts, et al. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*. pages 1631—1642.

I. Sutskever, O. Vinyals, and QV. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of NIPS*.

Alfred Tarski. 1944. The semantic conception of truth: and the foundations of semantics. *Philosophy and phenomenological research* 4(3):341–376.

Peter D. Turney and Patrick Pantel. 2010. From Frequency to Meaning: vector space models of semantics. *Journal of Artifical Intelligence Research* 37(1):141–188.

Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, koray kavukcuoglu, Oriol Vinyals, and Alex Graves. 2016. Conditional image generation with pixelcnn decoders. In *Proceedings of NIPS*.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2015. Order-embeddings of images and language. In *International Conference on Learning Representations (ICLR)*.

O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of CVPR*.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation* 39(2):165–210.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards universal paraphrastic sentence embeddings. In *International Conference on Learning Representations (ICLR)*.

Comparatives, Quantifiers, Proportions: A Multi-Task Model for the Learning of Quantities from Vision

Sandro Pezzelle[*], **Ionut-Teodor Sorodoc**[*†], **Raffaella Bernardi**[*‡]
[*]CIMeC - Center for Mind/Brain Sciences, University of Trento
[†]Universitat Pompeu Fabra Barcelona
[‡]DISI - Department of Information Engineering and Computer Science, University of Trento
[*]sandro.pezzelle@unitn.it, [†]ionutteodor.sorodoc@upf.edu
[‡]raffaella.bernardi@unitn.it

Abstract

The present work investigates whether different quantification mechanisms (set comparison, vague quantification, and proportional estimation) can be jointly learned from visual scenes by a multi-task computational model. The motivation is that, in humans, these processes underlie the same cognitive, non-symbolic ability, which allows an automatic estimation and comparison of set magnitudes. We show that when information about lower-complexity tasks is available, the higher-level proportional task becomes more accurate than when performed in isolation. Moreover, the multi-task model is able to generalize to unseen combinations of target/non-target objects. Consistently with behavioral evidence showing the interference of absolute number in the proportional task, the multi-task model no longer works when asked to provide the number of target objects in the scene.

1 Introduction

Understanding and producing sentences like 'There are *more* cars than parking lots', '*Most* of the supporters wear blue t-shirts', '*Twenty percent* of the trees have been planted last year', or '*Seven* students passed the exam', is a fundamental competence which allows speakers to communicate information about quantities. Crucially, the type of information conveyed by these expressions, as well as their underlying cognitive mechanisms, are not equivalent, as suggested by evidence from linguistics, language acquisition, and cognition.

First, comparatives ('more', 'less'), quantifiers ('some', 'most', 'all'), and proportions ('20%', 'two thirds') express a comparison or relation *between sets* (e.g., between the set of cars and the set of parking lots). Such relational information is rather coarse when expressed by comparatives and vague quantifiers, more precise when denoted by proportions. In contrast, numbers ('one', 'six', 'twenty-two') denote the exact, absolute cardinality of the items belonging to *one set* (e.g., the set of students who passed the exam).

Second, during language acquisition, these expressions are neither learned at the same time nor governed by the same rules. Recent evidence showed that children can understand comparatives at around 3.3 years (Odic et al., 2013; Bryant, 2017), with quantifiers being learned a few months later, at around 3.4-3.6 years (Hurewitz et al., 2006; Minai, 2006; Halberda et al., 2008). Crucially, knowing the meaning of numbers, an ability that starts not before the age of 3.5 years (Le Corre and Carey, 2007), is not required to understand and use these expressions. As for proportions, they are acquired significantly later, being fully mastered only at the age of 9 or 10 (Hartnett and Gelman, 1998; Moss and Case, 1999; Sophian, 2000).

Third, converging evidence from cognition and neuroscience supports the hypothesis that some important components of these expressions of quantity are grounded on a preverbal, non-symbolic system representing magnitudes (Piazza, 2010). This system, often referred to as Approximate Number System (ANS), is invariant to the sensory modality and almost universal in the animal domain, and consists in the ability of holistically extracting and comparing approximate numerosities (Piazza and Eger, 2016). In humans, it is present since the youngest age, with 6-month-old infants being able to automatically compare sets and combine them by means of proto-arithmetical operations (Xu and Spelke, 2000; McCrink and Wynn, 2004). Since it obeys Weber's law, according to which highly differing sets (e.g. 2:8) are easier to discriminate than highly similar sets (e.g. 7:8), ANS has been recently claimed to be a *ratio-based* mechanism (Sidney et al., 2017; Matthews et al., 2016). In support of this, behavioral findings indicate that, in non-symbolic

Proceedings of NAACL-HLT 2018, pages 419–430
New Orleans, Louisiana, June 1 - 6, 2018.

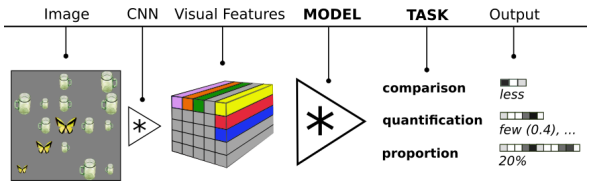


Figure 1: Toy representation of the quantification tasks and corresponding outputs explored in the paper. Note that quantification always refers to animals (target set).

contexts (e.g. visual scenes), proportional values are extracted holistically, i.e. without relying on the pre-computed cardinalities of the sets (Fabbri et al., 2012; Yang et al., 2015). Indeed, people are fairly accurate in providing the proportion of targets in a scene, even in high-speed settings (Healey et al., 1996; Treisman, 2006). Similarly, in briefly-presented scenes, the interpretation of quantifiers is shown to be best described by proportional information (Pezzelle et al., under review).

Altogether, this suggests that performing (1) set comparison, (2) vague quantification, and (3) proportional estimation, which all rely on information regarding relations among sets, underlies increasingly-complex steps of the same mechanism. Notably, such complexity would range from 'more/less' judgements to proportional estimation, as suggested by the increasing precision of ANS through years (Halberda and Feigenson, 2008), the reported boundary role of 'half' in early proportional reasoning (Spinillo and Bryant, 1991), and the different age of acquisition of the corresponding linguistic expressions. Finally, the ratio-based operation underlying these task would be different from (and possibly conflicting with) that of estimating the absolute numerosity of one set. Indeed, absolute numbers are found to interfere with the access to proportions (Fabbri et al., 2012).

Inspired by this converging evidence, the present work proposes a computational framework to explore various quantification tasks in the visual domain (see Figure 1). In particular, we investigate whether ratio-based quantification tasks can be modeled by a single, multi-task learning neural network. Given a synthetic scene depicting animals (in our setting, the 'target' objects) and artifacts ('non-target'), our model is designed to jointly perform all the tasks by means of an architecture that reflects their increasing complexity.[1] To perform proportional estimation (the most complex), the model builds on the representations learned to perform vague quantification and, in turn, set comparison (the least complex). We show that the multi-task model achieves both higher accuracy and higher generalization power compared to the one-task models. In contrast, we prove that introducing the absolute number task in the loop is not beneficial and indeed hurts the performance.

Our main contribution lies in the novel application and evaluation of a multi-task learning architecture on the task of jointly modeling 3 different quantification operations. On the one hand, our results confirm the interdependency of the mechanisms underlying the tasks of set comparison, vague quantification, and proportional estimation. On the other, we provide further evidence on the effectiveness of these computational architectures.

2 Related Work

2.1 Quantities in Language & Vision

In recent years, the task of extracting quantity information from visual scenes has been tackled via Visual Question Answering (VQA). Given a real image and a natural language question, a VQA computational model is asked to understand the image, the linguistic query, and their interaction to provide the correct answer. So-called *count* questions, i.e. 'How many *Xs* have the property *Y*?', are very frequent and have been shown to be particularly challenging for any model (Antol et al., 2015; Malinowski et al., 2015; Ren et al., 2015; Fukui et al., 2016). The difficulty of the task has been further confirmed by the similarly poor performance achieved even on the 'diagnostic' datasets, which include synthetic visual scenes depicting geometric shapes (Johnson et al., 2017; Suhr et al., 2017).

Using Convolutional Neural Networks (CNN), a number of works in Computer Vision (CV) have proposed specific architectures for counting digits (Seguí et al., 2015), people in the crowd (Zhang et al., 2015a), and penguins (Arteta et al., 2016). With a more cognitive flavor, Chattopadhyay et al. (2017) employed a 'divide-and-conquer' strategy to split the image into subparts and count the objects in each subpart by mimicking the 'subitizing' mechanism (i.e. numerosities up to 3-4 can be rapidly and accurately appreciated). Inspired by

[1]The dataset and the code can be downloaded from github.com/sandropezzelle/multitask-quant

the same cognitive ability is Zhang et al. (2015b), who trained a CNN to detect and count the salient objects in the image. Except Suhr et al. (2017), who evaluated models against various types of quantity expressions (including existential quantifiers), these works were just focused on the absolute number.

More akin to our work is Stoianov and Zorzi (2012), who showed that hierarchical generative models learn ANS as a statistical property of (synthetic) images. Their networks were tested on the task of set comparison ('more/less') and obtained 93% accuracy. A few studies specifically focused on the learning of quantifiers. Sorodoc et al. (2016) proposed a model to assign the correct quantifier to synthetic scenes of colored dots, whereas Sorodoc et al. (2018) operationalized the same task in a VQA fashion, using real images and object-property queries (e.g. 'How many *dogs* are *black*?'). Overall, the results of these studies showed that vague quantification can be learned by neural networks, though the performance is much lower when using real images and complex queries. Finally, Pezzelle et al. (2017) investigated the difference between the learning of cardinals and quantifiers from visual scenes, showing that they require two distinct computational operations. To our knowledge, this is the first attempt to jointly investigate the whole range of quantification mechanisms. Moreover, we are the first to exploit a multi-task learning paradigm for exploring the interactions between set comparison, vague quantification, and proportions.

2.2 Multi-Task Learning

Multi-Task Learning (MTL) has been shown to be very effective for a wide range of applications in machine learning (for an overview, see Ruder (2017)). The core idea is that different and yet related tasks can be jointly learned by a multi-purpose model rather than by separate and highly fine-tuned models. Since they share representations between related (or 'auxiliary') tasks, multi-task models are more robust and generalize better than single-task models. Successful applications of MTL have been proposed in CV to improve object classification (Girshick, 2015), face detection and rotation (Zhang et al., 2014; Yim et al., 2015), and to jointly perform a number of tasks as object detection, semantic segmentation, etc. (Misra et al., 2016; Li and Hoiem, 2016). Though, recently, a few studies applied MTL techniques to either count or estimate the number of objects in a scene (Sun et al., 2017; Sindagi and Patel, 2017), to our knowledge none of them were devoted to the learning of various quantification mechanisms.

In the field of natural language processing (NLP), MTL turned out to be beneficial for machine translation (Luong et al., 2016) and for a range of tasks such as chunking, tagging, semantic role labelling, etc. (Collobert et al., 2011; Søgaard and Goldberg, 2016; Bingel and Søgaard, 2017). In particular, Søgaard and Goldberg (2016) showed the benefits of keeping low-level tasks at the lower layers of the network, a setting which enables higher-level tasks to make a better use of the shared representations. Since this finding was also in line with previous evidence suggesting a natural order among different tasks (Shen and Sarkar, 2005), further work proposed MTL models in which several increasingly-complex tasks are hierarchically ordered (Hashimoto et al., 2017). The intuition behind this architecture, referred to as 'joint many-task model' in the source paper (Hashimoto et al., 2017), as well as its technical implementation, constitute the building blocks of the model proposed in the present study.

3 Tasks and Dataset

3.1 Tasks

Given a visual scene depicting a number of animals (targets) and artifacts (non-targets), we explore the following tasks, represented in Figure 1:

(a) set comparison (hence, **setComp**), i.e. judging whether the targets are 'more', 'same', 'less' than non-targets;

(b) vague quantification (hence, **vagueQ**), i.e. predicting the probability to use each of the 9 quantifiers ('none', 'almost none', 'few', 'the smaller part', 'some', 'many', 'most', 'almost all', 'all') to refer to the target set;

(c) proportional estimation (hence, **propTarg**), i.e. predicting the proportion of targets choosing among 17 ratios, ranging from 0 to 100%.

Tasks (a) and (c) are operationalized as classification problems and evaluated through accuracy. That is, only one answer out of 3 and 17, respectively, is considered as correct. Given the vague status of quantifiers, whose meanings are 'fuzzy' and overlapping, task (b) is evaluated by means

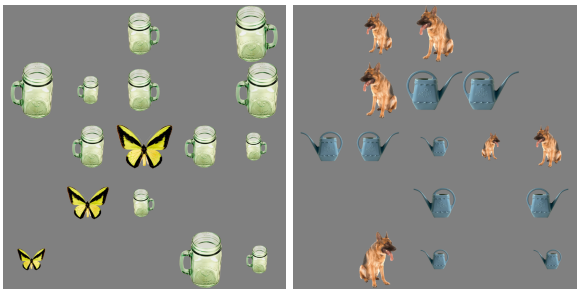


Figure 2: Two scenes included in our dataset. The letfmost one depicts a ratio 1:4 (3 animals, 12 artifacts, 15 total items), the rightmost one a ratio 2:3 (6, 9, 15).

of Pearson's correlation (r) between the predicted and the ground-truth probability vector (cf. § 3.2), for each datapoint.[2] The overall r is obtained by averaging these scores. It is worth mentioning that we could either evaluate (b) in terms of a classification task or operationalize (a) and (c) in terms of a correlation with human responses. The former evaluation is straightforward and can be easily carried out by picking the quantifier with the highest probability. The latter, in contrast, implies relying on behavioral data assessing the degree of overlap between ground-truth classes and speakers' choice. Though interesting, such evaluation is less crucial given the discrete, non-overlapping nature of the classes in tasks (a) and (c).

The tasks are explored by means of a MTL network that jointly performs the three quantification operations (see § 4.2). The intuition is that solving the lower-level tasks would be beneficial for tackling the higher-level ones. In particular, providing a proportional estimation ('80%') after performing vagueQ ('most') and setComp ('more') should lead to a higher accuracy in the highest-level task, which represents a further step in complexity compared to the previous ones. Moreover, lower-level tasks might be boosted in accuracy by the higher-level ones, since the latter include all the operations that are needed to carry out the former. In addition to the MTL model, we test a number of 'one-task' networks specifically designed to solve one task at a time (see § 4.1).

3.2 Dataset

We built a large dataset of synthetic visual scenes depicting a variable number of animals and artifacts on the top of a neutral, grey background

[2]We also experimented with Mean Average Error and dot product and found the same patterns of results (not reported).

	train	val	test	**total**
no. datapoints	11.9K	1.7K	3.4K	**17K**
% datapoints	70%	10%	20%	**100%**

Table 1: Number and partitioning of the datapoints.

(see Figure 2). In doing so, we employed the same methodology and materials used in Pezzelle et al. (under review), where the use of quantifiers in *grounded* contexts was explored by asking participants to select the most suitable quantifier for a given scene. Since the category of animals was always treated as the 'target', and that of artifacts as the 'non-target', we will henceforth use this terminology throughout the paper. The scenes were automatically generated by an in-house script using the following pipeline: (a) Two natural images, one depicting a target object (e.g. a butterfly) and one depicting a non-target (e.g. a mug) were randomly picked up from a sample of the dataset by Kiani et al. (2007). The sample was obtained by Pezzelle et al. (under review), who manually selected pictures depicting whole items (not just parts) and whose color, orientation and shape were not deceptive. In total, 100 unique instances of animals and 145 unique instances of artifacts were included; (b) The proportion of targets in the scene (e.g. 20%) was chosen by selecting one among 17 pre-defined *ratios* between targets:non-targets (e.g. 1:4, 'four non-targets to one target'). Out of 17 ratios, 8 were positive (targets $>$ 50%), 8 negative (targets $<$ 50%), and 1 equal (targets = 50%); (c) The absolute number of targets/non-targets was chosen to equally represent the various combinations available for a given ratio (e.g., for ratio 1:4: 1-4, 2-8, 3-12, 4-16), with the constraint of having a number of total objects in the scene (targets+non-targets) ranging from 3 to 20. In total, 97 combinations were represented in the dataset, with an average of 5.7 combinations/ratio (min 2, max 18); (d) To inject some variability, the instances of target/non-target objects were randomly resized according to one of three possible sizes (i.e. medium, big, and small) and flipped on the vertical axis before being randomly inserted onto a 5*5-cell virtual grid. As reported in Table 1, 17K scenes balanced per ratio (1K scenes/ratio) were generated and further split into train (70%), validation (10%), and test (20%).

Ground-truth classes for the tasks of setComp and propTarg were automatically assigned to each scene while generating the data. For vagueQ,

we took the probability distributions obtained on a dataset of 340 scenes by Pezzelle et al. (under review) and we applied them to our datapoints, which were built in the exact same way. These probability distributions had been collected by asking participants to select, from a list of 9 quantifiers (reported in § 3.1), the most suitable one to describe the target objects in a visual scene presented for 1 second. In particular, they were computed against the proportion of targets in the scene, which in that study was shown to be the overall best predictor for quantifiers. To illustrate, given a scene containing 20% of targets (cf. leftmost panel in Figure 2), the probability of choosing 'few' (ranging from 0 to 1) is 0.38, 'almost none' 0.27, 'the smaller part' 0.25, etc. It is worth mentioning that, for scenes containing either 100% or 0% targets the probability of choosing 'all' and 'none', respectively, is around 1. In all other cases, the distribution of probabilities is fuzzier and reflects the largely overlapping use of quantifiers, as in the example above. On average, the probability of the most-chosen quantifier across ratios is 0.53. Though this number cannot be seen as a genuine inter-annotator agreement score, it suggests that, on average, there is one quantifier which is preferred over the others.

4 Models

In this section, we describe the various models implemented to perform the tasks. For each model, several settings and parameters were evaluated by means of a thorough ablation analysis. Based on a number of factors like performance, speed, and stability of the networks, we opted for using ReLU nonlinear activation at all hidden layers and the simple and effective Stochastic Gradient Descent (SGD) as optimizer (lr = 0.01). We run each model for 100 epochs and saved weights and parameters of the epoch with the lowest validation loss. The best model was then used to obtain the predictions in the test set. All models were implemented using Keras.[3]

4.1 One-Task Models

We implemented separate models to tackle one task at a time. For each task, in particular, both a network using 'frozen' (i.e. pretrained) visual features and one computing the visual features in an 'end-to-end' fashion were tested.

[3] `https://keras.io/`

One-Task-Frozen These models are simple, 2-layer (ReLU) Multi-Layer Perceptron (MLP) networks that take as input a 2048-d frozen representation of the scene and output a vector containing softmax probability values. The frozen representation of the scene had been previously extracted using the state-of-art Inception v3 CNN (Szegedy et al., 2016) pretrained on ImageNet (Deng et al., 2009). In particular, the network is fed with the average of the features computed by the last Convolutional layer, which has size 25*2048.

One-Task-End2end These models are MLP networks that take as input the 203*203-pixel image and compute the visual features by means of the embedded Inception v3 module, which outputs 25*2048-d vectors (the grey and colored box in Figure 1). Subsequently, the 25 feature vectors are reduced twice via ReLU hidden layers, then concatenated, reduced (ReLU), and fed into a softmax layer to obtain the probability values.

4.2 Multi-Task Model

The `multi-task-prop` model performs 3 tasks at the *same time* with an architecture that reproduces in its *order* the conjectured complexity (see Figure 3 and its caption for technical details). The model has a core structure, represented by layers 1-5 in the figure, which is *shared* across tasks and trained with multiple outputs. In particular, (a) layers 1, 2, and 3 are trained using information regarding the output of all 3 tasks. That is, these layers are updated three times by as many back-propagation passes: One on the top of setComp output, the second on the top of vagueQ output, the third on the top of propTarg output; (b) layers 4 and 5 are affected by information regarding the output of vagueQ and propTarg, and thus updated twice; (c) layers 6 and 7 are updated once, on the top of the output of propTarg. Importantly, the three lower layers in Figure 3 (concatenation, ReLU, softmax) are not shared between the tasks, but specialized to output each a specific prediction. As can be noted, the order of the tasks reflects their complexity, since the last task in the pipeline has 2 more layers than the preceding one and 4 more than the first one.

5 Results

Table 2 reports the performance of each model in the various tasks (note that the lowest row and the rightmost column report results described

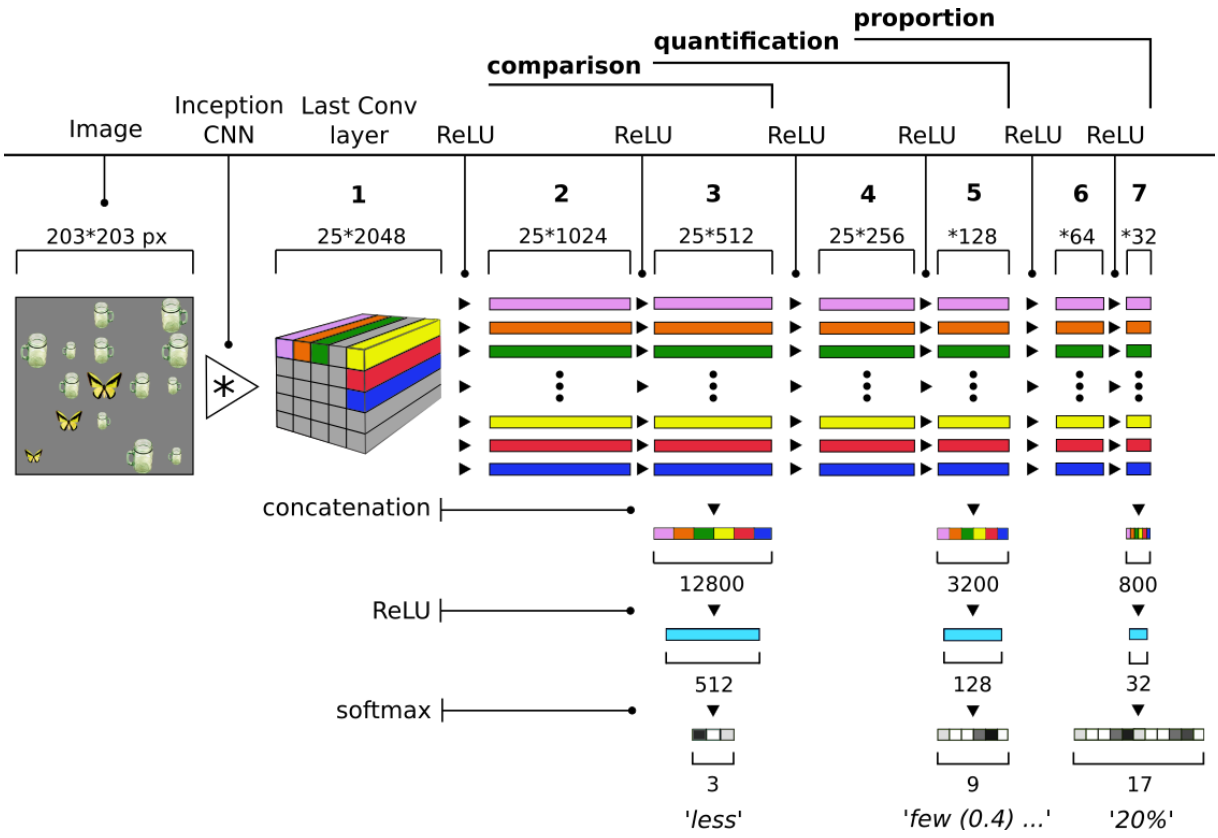


Figure 3: Architecture of the `multi-task-prop` model jointly performing (a) set comparison, (b) vague quantification, and (c) proportional estimation. Given a 203*203-pixel image as input, the model extracts a 25*2048 representation from the last Convolutional layer of the Inception v3. Subsequently, the vectors are reduced twice via ReLU hidden layers to 1024 and 512 dimensions. The 512-d vectors are concatenated and reduced, then a softmax layer is applied to output a 3-d vector with probability distributions for task (a). The same structure (i.e., 2 hidden layers, concatenation, reduction, and softmax) is repeated for tasks (b) and (c). All the tasks are trained with cross-entropy. To evaluate tasks (a) and (c), in testing, we extract the highest-probability class and compute **accuracy**, whereas task (b) is evaluated via **Pearson's correlation** against the 9-d ground-truth probability vector.

in § 6.1). In setComp, all the models are neatly above chance/majority level (0.47). In particular, the `one-task-end2end` model achieves a remarkable 0.90 acc., which is more than 10% better compared to the simple `one-task-frozen` model (0.78). The same pattern of results can be observed for vagueQ, where the Pearson's correlation (r) between the ground-truth and the predicted probability vector is around 0.96, that is more than 30% over the simpler model (0.62). This gap increases even more in propTarg, where the accuracy of the frozen model is more than 40 points below the one achieved by the `one-task-end2end` model (0.21 against 0.66). These results firmly indicate that, on the one hand, the frozen representation of the visual scene encodes little information about the proportion of targets (likely due to the the different task for which they were pretrained, i.e. object classification). On the other hand, computing the visual features in an end-to-end fashion leads to a significant improvement, suggesting that the network learns to pay attention to features that are helpful for specific tasks.

The most interesting results, however, are those achieved by the multi-task model, which turns out to be the best in all the tasks. As reported in Table 2, sharing the weights between the various tasks is especially beneficial for propTarg, where the accuracy reaches 0.92, that is, more than 25 points over the end-to-end, one-task model. An almost perfect performance of the model in this task can be observed in Figure 4, which reports the confusion matrix with the errors made by the model. As can be seen, the few errors are between 'touching' classes, e.g. between ratio 3:4 (43% of targets) and ratio 2:3 (40%). Since these classes

model	setComp	vagueQ	propTarg	nTarg
	accuracy	*Pearson r*	*accuracy*	*accuracy*
chance/majority	0.470	0.320	0.058	0.132
one-task-frozen	0.783	0.622	0.210	0.312
one-task-end2end	0.902	0.964	0.659	**0.966**
multi-task-prop	**0.995**	**0.982**	**0.918**	–
multi-task-number	0.854	0.807	–	0.478

Table 2: Performance of the models in the tasks of set comparison (setComp), vague quantification (vagueQ), proportional estimation (propTarg), and absolute number of targets (nTarg). Values in **bold** are the highest.

differ by a very small percentage, we gain indirect evidence that the model is learning some kind of proportional information rather than trivial associations between scenes and orthogonal classes.

To further explore this point, one way is to inspect the last layer of the proportional task (i.e. the 32-d turquoise vector in Figure 3). If the vectors contain information regarding the proportion of targets, we should expect scenes depicting the same proportion to have a similar representation. Also, scenes with similar proportions (e.g. 40% and 43%) would be closer to each other than are scenes with different proportions (e.g. 25% and 75%). Figure 5 depicts the results of a two-dimensional PCA analysis performed on the vectors of the last layer of the proportional task (the 32-d vectors).[4] As can be noted, scenes depicting the same proportion clearly cluster together, thus indicating that using these representations in a retrieval task would lead to a very high precision. Crucially, the clusters are perfectly ordered with respect to proportion. Starting from the purple cluster on the left side (90%) and proceeding clockwise, we find 83% (green), 80% (turquoise), 75% (brown), and so on, until reaching 10% (light blue). Proportions 0% (blue) and 100% (yellow) are neatly separated from the other clusters, being at the extremes of the 'clock'.

An improvement in the results can be also observed for setComp and vaqueQ, where the model achieves 0.99 acc. and 0.98 *r*, respectively. Figure 6 reports, for each quantifier, the probability values predicted by the model against the ground-truth ones. As can be seen, the red lines (model) approximate very closely the green ones (humans). In the following section, we perform further experiments to provide a deeper evaluation of the results.

6 In-Depth Evaluation

6.1 Absolute Numbers in the Loop

As discussed in § 1, the cognitive operation underlying setComp, vagueQ, and propTarg is different compared to that of estimating the absolute number of objects included in one set. To investigate whether such dissociation emerges at the computational level, we tested a modified version of our proposed multi-task model where propTarg task

[4] We used `https://projector.tensorflow.org/`

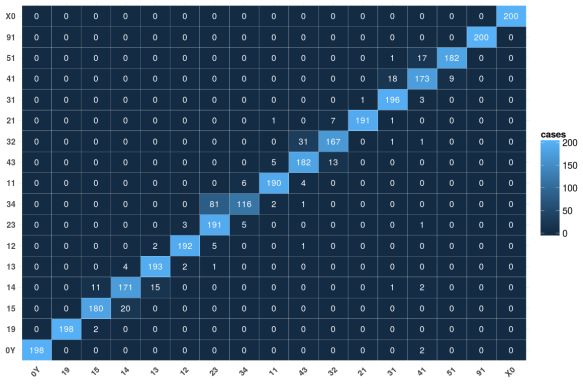


Figure 4: PropTarg. Heatmap reporting the errors made by the `multi-task-prop` model. Note that labels refer to *ratios*, i.e. 14 stands for ratio 1:4 (20% targets).

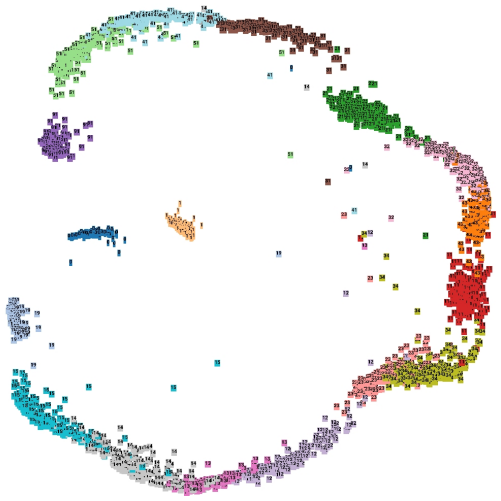


Figure 5: PCA visualization of the last layer (before softmax) of the proportional task in the MTL model.

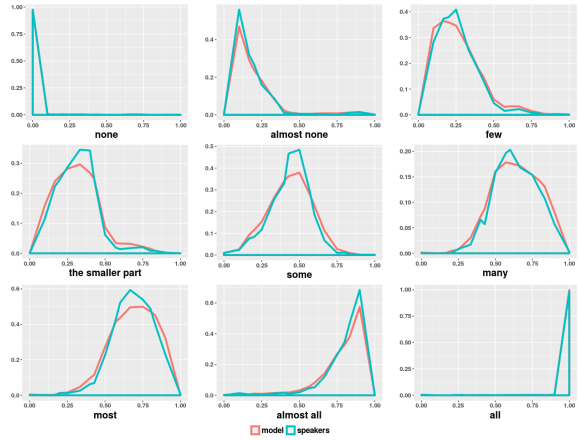


Figure 6: VagueQ. Probability values predicted by the `multi-task-prop` model against ground-truth probability distributions for each quantifier.

has been replaced with nTarg, namely the task of predicting the absolute number of targets. One-task models were also tested to evaluate the difficulty of the task when performed in isolation. Since the number of targets in the scenes ranges from 0 to 20, nTarg is evaluated as a 21-class classification task (majority class 0.13).

As reported in Table 2, the accuracy achieved by the `one-task-end2end` model is extremely high, i.e. around 0.97. This suggests that, when learned in isolation, the task is fairly easy, but only if the features are computed *within* the model. In fact, using frozen features results in a quite low accuracy, namely 0.31. This pattern of results is even more interesting if compared against the results of the `multi-task-number` model. When included in the multi-task pipeline, in fact, nTarg has a huge, 50-point accuracy drop (0.48). Moreover, both setComp and vagueQ turn out to be significantly *hurt* by the highest-level task, and experience a drop of around 14 and 17 points compared to the `one-task-end2end` model, respectively. These findings seem to corroborate the incompatibility of the operations needed for solving the tasks.

6.2 Reversing the Architecture

Previous work exploring MTL suggested that defining a hierarchy of increasingly-complex tasks is beneficial for jointly learning related tasks (see § 2.2). In the present work, the order of the tasks was inspired by cognitive and linguistic abilities (see § 1). Though cognitively implausible, it might still be the case that the model is able to learn even when reversing the order of the tasks, i.e. from the conjectured highest-level to the lowest-level one. To shed light on this issue, we tested the `multi-task-prop` model after reversing its architecture. That is, propTarg is now the first task, followed by vagueQ, and setComp.

model	setComp	vagueQ	propTarg
	accuracy	*Pearson r*	*accuracy*
chance/majority	0.470	0.320	0.058
one-task-frozen	0.763	0.548	0.068
one-task-end2end	0.793	0.922	0.059
multi-task-prop	**0.943**	**0.960**	**0.539**

Table 3: Unseen dataset. Performance of the models in each task. Values in **bold** are the highest.

In contrast with the pattern of results obtained by the original pipeline, no benefits are observed for this version of MTL model compared to one-task networks. In particular, both vagueQ (0.32 *r*) and propTarg (0.08 acc.) performance are around chance level, with setComp reaching just 0.65 acc., i.e. 25 point lower than the `one-task-end2end` model. The pipeline of increasing complexity motivated theoretically is thus confirmed at the computational level.

6.3 Does MTL Generalize?

As discussed in § 2.2, MTL is usually claimed to allow a higher generalization power. To investigate whether our proposed `multi-task-prop` model genuinely learns to quantify from visual scenes, and not just associations between patterns and classes, we tested it with unseen combinations of targets/non-targets. The motivation is that, even in the most challenging propTarg task, the model might learn to match a given combination, e.g. 3:12, to a given proportion, i.e. 20%. If this is the case, the model would solve the task by learning "just" to assign a class to each of the 97 possible combinations included in the dataset. If it learns a more abstract representation of the proportion of targets depicted in the scene, in contrast, it should be able to generalize to unseen combinations.

We built an additional dataset using the exact same pipeline described in § 3.2. This time, however, we randomly selected one combination per ratio (17 combinations in total) to be used only for validation and testing. The remaining 80 combinations were used for training. A balanced number of datapoints for each combination were generated in val/test, whereas datapoints in training set

were balanced with respect to ratios, by randomly selecting scenes among the remaining combinations. The *unseen* dataset included around 14K datapoints (80% train, 10% val, 10% test). Table 3 reports the results of the models on the unseen dataset. Starting from setComp, we note a similar and fairly high accuracy achieved by the two one-task models (0.76 and 0.79, respectively). In vagueQ, in contrast, the `one-task-end2end` model neatly outperforms the simpler model (0.92 vs. 0.55 *r*). Finally, in propTarg both models are at chance level, with an accuracy that is lower than 0.07. Overall, this pattern of results suggests that propTarg is an extremely hard task for the separate models, which are not able to generalize to unseen combinations. The `multi-task-prop` model, in contrast, shows a fairly high generalization power. In particular, it achieves 0.54 acc. in propTarg, that is, almost 10 times chance level. The overall good performance in predicting the correct proportion can be appreciated in Figure 7, where the errors are represented by means of a heatmap. The error analysis reveals that end-of-the-scale proportions (0% and 100%) are the easiest, followed by proportions 75% (3:1), 67% (2:1), 50% (1:1), and 60% (3:2). More in general, negative ratios (targets $< 50\%$) are mispredicted to a much greater extent than are positive ones. Moreover, the model shows a bias toward some proportions, that the model seems 'to see everywhere'. However, the fact that the errors are found among the adjacent ratios (similar proportions) seems to be a convincing evidence that the model learns representations encoding genuine proportional information. Finally, it is worth mentioning that in setComp and vagueQ the model achieves very high results, 0.94 acc. and 0.96 *r*, respectively.

7 Discussion

In the present study, we investigated whether *ratio-based* quantification mechanisms, expressed in language by comparatives, quantifiers, and proportions, can be computationally modeled in vision exploiting MTL. We proved that sharing a common core turned out to boost the performance in all the tasks, supporting evidence from linguistics, language acquisition, and cognition. Moreover, we showed (a) the increasing complexity of the tasks, (b) the interference of absolute number, and (c) the high generalization power of MTL. These results lead to many additional questions.

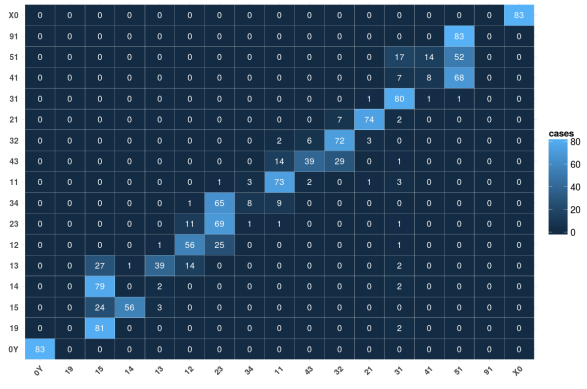


Figure 7: PropTarg. Heatmap with the errors made by the `multi-task-prop` model in the unseen dataset.

For instance, can these methods be successfully applied to datasets of real scenes? We firmly believe this to be the case, though the results might be affected by the natural biases contained in those images. Also, is this pipeline of increasing complexity specific to vision (non-symbolic level), or is it shared across modalities, *in primis* language? Since linguistic expressions of quantity are grounded on a non-symbolic system, we might expect that a model trained on one modality can be applied to another, at least to some extent. Even further, jointly learning representations from both modalities might represent an even more natural, human-like way to learn and refer to quantities. Further work is needed to explore all these issues.

Acknowledgments

We kindly acknowledge Gemma Boleda and the AMORE team (UPF), Raquel Fernández and the Dialogue Modelling Group (UvA) for the feedback, advice and support. We are also grateful to Aurélie Herbelot, Stephan Lee, Manuela Piazza, Sebastian Ruder, and the anonymous reviewers for their valuable comments. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 715154). We gratefully acknowledge the support of NVIDIA Corporation with the donation of GPUs used for this research. This paper reflects the authors' view only, and the EU is not responsible for any use that may be made of the information it contains.

References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual Question Answering. In *Proceedings of the IEEE International Conference on Computer Vision*. pages 2425–2433.

Carlos Arteta, Victor Lempitsky, and Andrew Zisserman. 2016. Counting in the wild. In *European Conference on Computer Vision*. Springer, pages 483–498.

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. *EACL 2017* page 164.

Peter Bryant. 2017. *Perception and understanding in young children: An experimental approach*, volume 4. Routledge.

Prithvijit Chattopadhyay, Ramakrishna Vedantam, Ramprasaath R. Selvaraju, Dhruv Batra, and Devi Parikh. 2017. Counting everyday objects in everyday scenes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(Aug):2493–2537.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, pages 248–255.

Sara Fabbri, Sara Caviola, Joey Tang, Marco Zorzi, and Brian Butterworth. 2012. The role of numerosity in processing nonsymbolic proportions. *The Quarterly Journal of Experimental Psychology* 65(12):2435–2446.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding. In *Conference on Empirical Methods in Natural Language Processing*. ACL, pages 457–468.

Ross Girshick. 2015. Fast R-CNN. In *Proceedings of the IEEE international conference on computer vision*. pages 1440–1448.

Justin Halberda and Lisa Feigenson. 2008. Developmental change in the acuity of the "Number Sense": The Approximate Number System in 3-, 4-, 5-, and 6-year-olds and adults. *Developmental psychology* 44(5):1457.

Justin Halberda, Len Taing, and Jeffrey Lidz. 2008. The development of 'most' comprehension and its potential dependence on counting ability in preschoolers. *Language Learning and Development* 4(2):99–121.

Patrice Hartnett and Rochel Gelman. 1998. Early understandings of numbers: Paths or barriers to the construction of new understandings? *Learning and instruction* 8(4):341–374.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A Joint Many-Task Model: Growing a Neural Network for Multiple NLP Tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Copenhagen, Denmark, pages 446–456.

Christopher G Healey, Kellogg S Booth, and James T Enns. 1996. High-speed visual estimation using preattentive processing. *ACM Transactions on Computer-Human Interaction (TOCHI)* 3(2):107–135.

Felicia Hurewitz, Anna Papafragou, Lila Gleitman, and Rochel Gelman. 2006. Asymmetries in the acquisition of numbers and quantifiers. *Language learning and development* 2(2):77–96.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pages 1988–1997.

Roozbeh Kiani, Hossein Esteky, Koorosh Mirpour, and Keiji Tanaka. 2007. Object category structure in response patterns of neuronal population in monkey inferior temporal cortex. *Journal of neurophysiology* 97(6):4296–4309.

Mathieu Le Corre and Susan Carey. 2007. One, two, three, four, nothing more: An investigation of the conceptual sources of the verbal counting principles. *Cognition* 105(2):395–438.

Zhizhong Li and Derek Hoiem. 2016. Learning without forgetting. In *European Conference on Computer Vision*. Springer, pages 614–629.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *International Conference on Learning Representations (ICLR)*. San Juan, Puerto Rico.

Mateusz Malinowski, Marcus Rohrbach, and Mario Fritz. 2015. Ask your neurons: A neural-based approach to answering questions about images. In *Proceedings of the IEEE international conference on computer vision*. pages 1–9.

Percival G Matthews, Mark Rose Lewis, and Edward M Hubbard. 2016. Individual differences in nonsymbolic ratio processing predict symbolic math performance. *Psychological science* 27(2):191–202.

Koleen McCrink and Karen Wynn. 2004. Large-number addition and subtraction by 9-month-old infants. *Psychological Science* 15(11):776–781.

Utako Minai. 2006. *Everyone knows, therefore every child knows: An investigation of logico-semantic competence in child language*. Ph.D. thesis, University of Maryland.

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

Joan Moss and Robbie Case. 1999. Developing children's understanding of the rational numbers: A new model and an experimental curriculum. *Journal for research in mathematics education* pages 122–147.

Darko Odic, Paul Pietroski, Tim Hunter, Jeffrey Lidz, and Justin Halberda. 2013. Young children's understanding of "more" and discrimination of number and surface area. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 39(2):451.

Sandro Pezzelle, Raffaella Bernardi, and Manuela Piazza. under review. Probing the mental scale of quantifiers. *Cognition* .

Sandro Pezzelle, Marco Marelli, and Raffaella Bernardi. 2017. Be precise or fuzzy: Learning the meaning of cardinals and quantifiers from vision. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, Valencia, Spain, pages 337–342.

Manuela Piazza. 2010. Neurocognitive start-up tools for symbolic number representations. *Trends in cognitive sciences* 14(12):542–551.

Manuela Piazza and Evelyn Eger. 2016. Neural foundations and functional specificity of number representations. *Neuropsychologia* 83:257–273.

Mengye Ren, Ryan Kiros, and Richard Zemel. 2015. Exploring models and data for image question answering. In *Advances in neural information processing systems*. pages 2953–2961.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098* .

Santi Seguí, Oriol Pujol, and Jordi Vitria. 2015. Learning to count with deep object features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. pages 90–96.

Hong Shen and Anoop Sarkar. 2005. Voting between multiple data representations for text chunking. In *Conference of the Canadian Society for Computational Studies of Intelligence*. Springer, pages 389–400.

Pooja G Sidney, Clarissa A Thompson, Percival G Matthews, and Edward M Hubbard. 2017. From continuous magnitudes to symbolic numbers: The centrality of ratio. *Behavioral and Brain Sciences* 40.

Vishwanath A Sindagi and Vishal M Patel. 2017. CNN-Based cascaded multi-task learning of high-level prior and density estimation for crowd counting. In *Advanced Video and Signal Based Surveillance (AVSS), 2017 14th IEEE International Conference on*. IEEE, pages 1–6.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. volume 2, pages 231–235.

Catherine Sophian. 2000. Perceptions of proportionality in young children: matching spatial ratios. *Cognition* 75(2):145 – 170.

Ionut Sorodoc, Angeliki Lazaridou, Gemma Boleda, Aurélie Herbelot, Sandro Pezzelle, and Raffaella Bernardi. 2016. "Look, some green circles!": Learning to quantify from images. In *Proceedings of the 5th Workshop on Vision and Language*. pages 75–79.

Ionut Sorodoc, Sandro Pezzelle, Aurélie Herbelot, Mariella Dimiccoli, and Raffaella Bernardi. 2018. Learning quantification from images: A structured neural architecture. *Natural Language Engineering* page 1–30.

Alina G Spinillo and Peter Bryant. 1991. Children's proportional judgments: The importance of "half". *Child Development* 62(3):427–440.

Ivilin Stoianov and Marco Zorzi. 2012. Emergence of a 'visual number sense' in hierarchical generative models. *Nature neuroscience* 15(2):194–196.

Alane Suhr, Mike Lewis, James Yeh, and Yoav Artzi. 2017. A corpus of natural language for visual reasoning. In *55th Annual Meeting of the Association for Computational Linguistics, ACL*.

Maojin Sun, Yan Wang, Teng Li, Jing Lv, and Jun Wu. 2017. Vehicle counting in crowded scenes with multi-channel and multi-task convolutional neural networks. *Journal of Visual Communication and Image Representation* 49:412–419.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the Inception Architecture for Computer Vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 2818–2826.

Anne Treisman. 2006. How the deployment of attention determines what we see. *Visual Cognition* 14(4-8):411–443. PMID: 17387378.

Fei Xu and Elizabeth S Spelke. 2000. Large number discrimination in 6-month-old infants. *Cognition* 74(1):B1–B11.

Ying Yang, Qingfen Hu, Di Wu, and Shuqi Yang. 2015. Children's and adults' automatic processing of proportion in a Stroop-like task. *International Journal of Behavioral Development* 39(2):97–104.

Junho Yim, Heechul Jung, ByungIn Yoo, Changkyu Choi, Dusik Park, and Junmo Kim. 2015. Rotating your face using multi-task deep neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 676–684.

Cong Zhang, Hongsheng Li, Xiaogang Wang, and Xiaokang Yang. 2015a. Cross-scene crowd counting via deep convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 833–841.

Jianming Zhang, Shugao Ma, Mehrnoosh Sameki, Stan Sclaroff, Margrit Betke, Zhe Lin, Xiaohui Shen, Brian Price, and Radomir Mech. 2015b. Salient object subitizing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 4045–4054.

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

Being Negative but Constructively: Lessons Learnt from Creating Better Visual Question Answering Datasets

Wei-Lun Chao*, **Hexiang Hu***, **Fei Sha**
University of Southern California
Los Angeles, California, USA
weilunchao760414@gmail.com, hexiang.frank.hu@gmail.com, feisha@usc.edu

Abstract

Visual question answering (Visual QA) has attracted a lot of attention lately, seen essentially as a form of (visual) Turing test that artificial intelligence should strive to achieve. In this paper, we study a crucial component of this task: how can we design good datasets for the task? We focus on the design of multiple-choice based datasets where the learner has to select the right answer from a set of candidate ones including the target (i.e. the correct one) and the decoys (i.e. the incorrect ones). Through careful analysis of the results attained by state-of-the-art learning models and human annotators on existing datasets, we show that the design of the decoy answers has a significant impact on how and what the learning models learn from the datasets. In particular, the resulting learner can ignore the visual information, the question, or both while still doing well on the task. Inspired by this, we propose automatic procedures to remedy such design deficiencies. We apply the procedures to re-construct decoy answers for two popular Visual QA datasets as well as to create a new Visual QA dataset from the Visual Genome project, resulting in the largest dataset for this task. Extensive empirical studies show that the design deficiencies have been alleviated in the remedied datasets and the performance on them is likely a more faithful indicator of the difference among learning models. The datasets are released and publicly available via http://www.teds.usc.edu/website_vqa/.

1 Introduction

Multimodal information processing tasks such as image captioning (Farhadi et al., 2010; Ordonez et al., 2011; Xu et al., 2015) and visual question answering (Visual QA) (Antol et al., 2015) have gained a lot of attention recently. A number of significant advances in learning algorithms have been made, along with the development of nearly two dozens of datasets in this very active research domain. Among those datasets, popular ones include MSCOCO (Lin et al., 2014; Chen et al., 2015), Visual Genome (Krishna et al., 2017), VQA (Antol et al., 2015), and several others. The overarching objective is that a learning machine needs to go beyond understanding different modalities of information separately (such as image recognition alone) and to learn how to correlate them in order to perform well on those tasks.

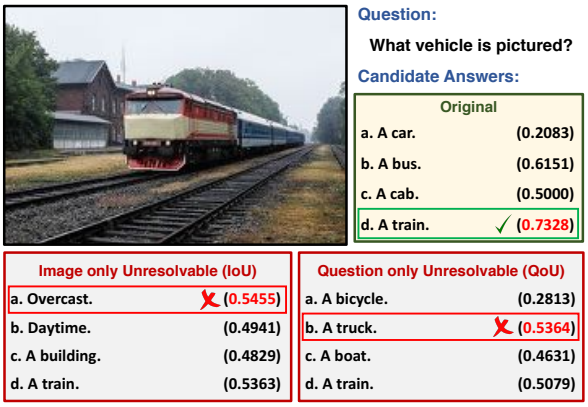


Figure 1: An illustration of how the shortcuts in the Visual7W dataset (Zhu et al., 2016) should be remedied. In the original dataset, the correct answer "A train" is easily selected by a machine as it is far often used as the correct answer than the other decoy (negative) answers. (The numbers in the brackets are probability scores computed using eq. (2)). Our two procedures — QoU and IoU (cf. Sect. 4) — create alternative decoys such that both the correct answer and the decoys are highly likely by examining either the image or the question **alone**. In these cases, machines make mistakes unless they consider all information **together**. Thus, the alternative decoys suggested our procedures are better designed to gauge how well a learning algorithm can understand all information equally well.

*Equal contributions

Proceedings of NAACL-HLT 2018, pages 431–441
New Orleans, Louisiana, June 1 - 6, 2018.

To evaluate the progress on those complex and more AI-like tasks is however a challenging topic. For tasks involving language generation, developing an automatic evaluation metric is itself an open problem (Anderson et al., 2016; Kilickaya et al., 2017; Liu et al., 2016; Kafle and Kanan, 2017b). Thus, many efforts have concentrated on tasks such as *multiple-choice* Visual QA (Antol et al., 2015; Zhu et al., 2016; Jabri et al., 2016) or selecting the best caption (Hodosh et al., 2013; Hodosh and Hockenmaier, 2016; Ding et al., 2016; Lin and Parikh, 2016), where the selection accuracy is a natural evaluation metric.

In this paper, we study how to design high-quality multiple choices for the Visual QA task. In this task, the machine (or the human annotator) is presented with an image, a question and a list of candidate answers. The goal is to select the correct answer through a consistent understanding of the image, the question and each of the candidate answers. As in any multiple-choice based tests (such as GRE), designing what should be presented as negative answers — we refer them as *decoys* — is as important as deciding the questions to ask. We all have had the experience of exploiting the elimination strategy: *This question is easy — none of the three answers could be right so the remaining one must be correct!*

While a clever strategy for taking exams, such "shortcuts" prevent us from studying faithfully how different learning algorithms comprehend the meanings in images and languages (e.g., the quality of the embeddings of both images and languages in a semantic space). It has been noted that machines can achieve very high accuracies of selecting the correct answer without the visual input (i.e., the image), the question, or both (Jabri et al., 2016; Antol et al., 2015). Clearly, the learning algorithms have overfit on incidental statistics in the datasets. For instance, if the decoy answers have rarely been used as the correct answers (to any questions), then the machine can rule out a decoy answer with a binary classifier that determines whether the answers are in the set of the correct answers — note that this classifier does not need to examine the image and it just needs to memorize the list of the correct answers in the training dataset. See Fig. 1 for an example, and Sect. 3 for more and detailed analysis.

We focus on minimizing the impacts of exploiting such shortcuts. We suggest a set of principles for creating decoy answers. In light of the amount of human efforts in curating existing datasets for the Visual QA task, we propose two procedures that revise those datasets such that the decoy answers are better designed. In contrast to some earlier works, the procedures are fully automatic and do not incur additional human annotator efforts. We apply the procedures to revise both Visual7W (Zhu et al., 2016) and VQA (Antol et al., 2015). Additionally, we create new multiple-choice based datasets from COCOQA (Ren et al., 2015) and the recently released VQA2 (Goyal et al., 2017) and Visual Genome datasets (Krishna et al., 2017). The one based on Visual Genome becomes the largest multiple-choice dataset for the Visual QA task, with more than one million image-question-candidate answers triplets.

We conduct extensive empirical and human studies to demonstrate the effectiveness of our procedures in creating high-quality datasets for the Visual QA task. In particular, we show that machines need to use all three information (image, questions and answers) to perform well — any missing information induces a large drop in performance. Furthermore, we show that humans dominate machines in the task. However, given the revised datasets are likely reflecting the true gap between the human and the machine understanding of multimodal information, we expect that advances in learning algorithms likely focus more on the task itself instead of overfitting to the idiosyncrasies in the datasets.

The rest of the paper is organized as follows. In Sect. 2, we describe related work. In Sect. 3, we analyze and discuss the design deficiencies in existing datasets. In Sect. 4, we describe our automatic procedures for remedying those deficiencies. In Sect. 5 we conduct experiments and analysis. We conclude the paper in Sect. 6.

2 Related Work

Wu et al. (2017) and Kafle and Kanan (2017b) provide recent overviews of the status quo of the Visual QA task. There are about two dozens of datasets for the task. Most of them use real-world images, while some are based on synthetic ones. Usually, for each image, multiple questions and their corresponding answers are generated. This can be achieved either by human annotators, or with an automatic procedure that uses captions or question templates and detailed image annota-

tions. We concentrate on 3 datasets: VQA (Antol et al., 2015), Visual7W (Zhu et al., 2016), and Visual Genome (Krishna et al., 2017). All of them use images from MSCOCO (Lin et al., 2014).

Besides the pairs of questions and correct answers, VQA, Visual7W, and visual Madlibs (Yu et al., 2015) provide decoy answers for each pair so that the task can be evaluated in multiple-choice selection accuracy. *What decoy answers to use* is the focus of our work.

In VQA, the decoys consist of human-generated plausible answers as well as high-frequency and random answers from the datasets. In Visual7W, the decoys are all human-generated plausible ones. Note that, humans generate those decoys by *only looking at the questions and the correct answers but **not** the images*. Thus, the decoys might be unrelated to the corresponding images. A learning algorithm can potentially examine the image alone and be able to identify the correct answer.

In visual Madlibs, the questions are generated with a limited set of question templates and the detailed annotations (e.g., objects) of the images. Thus, similarly, a learning model can examine the image alone and deduce the correct answer.

We propose automatic procedures to revise VQA and Visual7W (and to create new datasets based on COCOQA (Ren et al., 2015), VQA2 (Goyal et al., 2017), and Visual Genome) such that the decoy generation is carefully orchestrated to prevent learning algorithms from exploiting the shortcuts in the datasets by overfitting on incidental statistics. In particular, our design goal is that a learning machine needs to understand all the 3 components of an image-question-candidate answers triplet in order to make the right choice — ignoring either one or two components will result in drastic degradation in performance.

Our work is inspired by the experiments in (Jabri et al., 2016) where they observe that machines without looking at images or questions can still perform well on the Visual QA task. Others have also reported similar issues (Goyal et al., 2017; Zhang et al., 2016; Johnson et al., 2017; Agrawal et al., 2016; Kafle and Kanan, 2017a; Agrawal et al., 2018), though not in the multiple-choice setting. Our work extends theirs by providing more detailed analysis *as well as automatic procedures* to remedy those design deficiencies.

Besides Visual QA, VisDial (Das et al., 2017) and Ding et al. (2016) also propose automatic ways to generate decoys for the tasks of multiple-choice visual captioning and dialog, respectively.

Recently, Lin and Parikh (2017) study active learning for Visual QA: i.e., how to select informative image-question pairs (for acquiring annotations) or image-question-answer triplets for machines to "learn" from. On the other hand, our work further focuses on designing better datasets for "evaluating" a machine.

3 Analysis of Decoy Answers' Effects

In this section, we examine in detail the dataset Visual7W (Zhu et al., 2016), a popular choice for the Visual QA task. We demonstrate how the deficiencies in designing decoy questions impact the performance of learning algorithms.

In multiple-choice Visual QA datasets, a training or test example is a triplet that consists of an image I, a question Q, and a candidate answer set A. The set A contains a target T (the correct answer) and K decoys (incorrect answers) denoted by D. An IQA triplet is thus $\{\mathrm{I}, \mathrm{Q}, \mathrm{A} = \{\mathrm{T}, \mathrm{D}_1, \cdots, \mathrm{D}_K\}\}$. We use C to denote either the target or a decoy.

3.1 Visual QA models

We investigate how well a learning algorithm can perform when supplied with different modalities of information. We concentrate on the one hidden-layer MLP model proposed in (Jabri et al., 2016), which has achieved state-of-the-art results on the dataset Visual7W. The model computes a scoring function $f(c, i)$

$$f(c, i) = \sigma(\boldsymbol{U} \max(0, \boldsymbol{W} g(c, i)) + b) \qquad (1)$$

over a candidate answer c and the multimodal information i, where g is the joint feature of (c, i) and $\sigma(x) = 1/(1 + \exp(-x))$. The information i can be null, the image (I) alone, the question (Q) alone, or the combination of both (I+Q).

Given an IQA triplet, we use the penultimate layer of ResNet-200 (He et al., 2016) as visual features to represent I and the average WORD2VEC embeddings (Mikolov et al., 2013) as text features to represent Q and C. To form the joint feature $g(c, i)$, we just concatenate the features together. The candidate $c \in$ A that has the highest $f(c, i)$ score in prediction is selected as the model output.

We use the standard training, validation, and test splits of Visual7W, where each contains 69,817, 28,020, and 42,031 examples respectively.

Information used	Machine	Human
random	25.0	25.0
A	52.9	-
I + A	62.4	75.3
Q + A	58.2	36.4
I + Q + A	65.7	88.4

Table 1: Accuracy of selecting the right answers out of 4 choices (%) on the Visual QA task on Visual7W.

Each question has 4 candidate answers. The parameters of $f(c, i)$ are learned by minimizing the binary logistic loss of predicting whether or not a candidate c is the target of an IQA triplet. Details are in Sect. 5 and the Supplementary Material.

3.2 Analysis results

Machines find shortcuts Table 1 summarizes the performance of the learning models, together with the human studies we performed on a subset of 1,000 triplets (c.f. Sect. 5 for details). There are a few interesting observations.

First, in the row of "A" where only the candidate answers (and whether they are right or wrong) are used to train a learning model, the model performs significantly better than random guessing and humans (52.9% vs. 25%) — humans will deem each of the answers equally likely *without* looking at both the image and the question! Note that in this case, the information i in eq. (1) contains nothing. The model learns the specific statistics of the candidate answers in the dataset and exploits those. Adding the information about the image (i.e., the row of "I+A"), the machine improves significantly and gets close to the performance when all information is used (62.4% vs. 65.7%). There is a weaker correlation between the question and the answers as "Q+A" improves over "A" only modestly. This is expected. In the Visual7W dataset, the decoys are generated by human annotators as plausible answers to the questions without being shown the images — thus, many decoy answers do not have visual groundings. For instance, a question of "what animal is running?" elicits equally likely answers such as "dog", "tiger", "lion", or "cat", while an image of a dog running in the park will immediately rule out all 3 but the "dog", see Fig. 1 for a similar example. Thus, the performance of "I+A" implies that many IQA triplets can be solved by object, attribute or concept detection on the image, without understanding the questions. This is indeed the case also for humans — humans can achieve 75.3% by considering "I+A" and not "Q". Note that the difference between machine and human on "I+A" are likely due to their difference in understanding visual information.

Note that human improves significantly from "I+A" to "I+Q+A" with "Q" added, while the machine does so only marginally. The difference can be attributed to the difference in understanding the question and correlating with the answers between the two. Since each image corresponds to multiple questions or have multiple objects, solely relying on the image itself will not work well in principle. Such difference clearly indicates that in the Visual QA model, the language component is weak as the model cannot fully exploit the information in "Q", making a smaller relative improvement 5.3% (from 62.4% to 65.7%) where humans improved relatively 17.4%.

Shortcuts are due to design deficiencies We probe deeper on how the decoy answers have impacted the performance of learning models.

As explained above, the decoys are drawn from all plausible answers to a question, irrespective of whether they are visually grounded or not. We have also discovered that the targets (i.e., correct answers) are infrequently used as decoys.

Specifically, among the 69,817 training samples, there are 19,503 unique correct answers and each one of them is used about 3.6 times as correct answers to a question. However, among all the $69,817 \times 3 \approx 210K$ decoys, each correct answer appears 7.2 times on average, far below a chance level of 10.7 times ($210K \div 19,503 \approx 10.7$). This disparity exists in the test samples too. Consequently, the following rule, computing each answer's likelihood of being correct,

$$P(\text{correct}|\text{C}) = \begin{cases} 0.5, & \text{if C is never seen in training,} \\ \frac{\text{\# times C as target}}{\text{\# times C as target} + (\text{\# times C as decoys})/K}, & \text{otherwise,} \end{cases} \quad (2)$$

should perform well. Essentially, it measures how unbiased C is used as the target and the decoys. Indeed, it attains an accuracy of 48.73% on the test data, far better than the random guess and is close to the learning model using the answers' information only (the "A" row in Table 1).

Good rules for designing decoys Based on our analysis, we summarize the following guidance rules to design decoys: (1) **Question only Unresolvable (QoU)**. The decoys need to be equally

plausible to the question. Otherwise, machines can rely on the correlation between the question and candidate answers to tell the target from decoys, even without the images. Note that this is a principle that is being followed by most datasets. (2) **Neutrality**. The decoys answers should be equally likely used as the correct answers. (3) **Image only Unresolvable (IoU)**. The decoys need to be plausible to the image. That is, they should appear in the image, or there exist questions so that the decoys can be treated as targets to the image. Otherwise, Visual QA can be resolved by objects, attributes, or concepts detection in images, even without the questions.

Ideally, each decoy in an IQA triplet should meet the three principles. **Neutrality** is comparably easier to achieve by *reusing terms in the whole set of targets as decoys.* On the contrary, a decoy may hardly meet **QoU** and **IoU** simultaneously[1]. However, as long as all decoys of an IQA triplet meet **Neutrality** and some meet **QoU** and others meet **IoU**, the triplet as a whole still achieves the three principles — a machine ignoring either images or questions will likely perform poorly.

4 Creating Better Visual QA Datasets

In this section, we describe our approaches of remedying design deficiencies in the existing datasets for the Visual QA task. We introduce two automatic and widely-applicable procedures to create new decoys that can prevent learning models from exploiting incident statistics in the datasets.

4.1 Methods

Main ideas Our procedures operate on a dataset that already contains image-question-target (IQT) triplets, i.e., we do not assume it has decoys already. For instance, we have used our procedures to create a multiple-choice dataset from the Visual Genome dataset which has no decoy. We assume that each image in the dataset is coupled with "multiple" QT pairs, which is the case in nearly all the existing datasets. Given an IQT triplet (I, Q, T), we create two sets of decoy answers.

- **QoU-decoys**. We search among all other triplets that have similar questions to Q. The **targets** of those triplets are then collected as the decoys for T. As the targets to similar questions are likely plausible for the question Q, QoU-decoys likely follow the rules of **Neutrality** and **Question only Unresolvable (QoU)**. We compute the average WORD2VEC (Mikolov et al., 2013) to represent a question, and use the cosine similarity to measure the similarity between questions.
- **IoU-decoys**. We collect the **targets** from other triplets of the *same* image to be the decoys for T. The resulting decoys thus definitely follow the rules of **Neutrality** and **Image only Unresolvable (IoU)**.

We then combine the triplet (I, Q, T) with QoU-decoys and IoU-decoys to form an IQA triplet as a training or test sample.

Resolving ambiguous decoys One potential drawback of automatically selected decoys is that they may be semantically similar, ambiguous, or rephrased terms to the target (Zhu et al., 2016). We utilize two filtering steps to alleviate it. First, we perform string matching between a decoy and the target, deleting those decoys that contain or are covered by the target (e.g., "daytime" vs. "during the daytime" and "ponytail" vs. "pony tail").

Secondly, we utilize the WordNet hierarchy and the Wu-Palmer (WUP) score (Wu and Palmer, 1994) to eliminate semantically similar decoys. The WUP score measures how similar two *word senses* are (in the range of $[0, 1]$), based on the depth of them in the taxonomy and that of their least common subsumer. We compute the similarity of two strings according to the WUP scores in a similar manner to (Malinowski and Fritz, 2014), in which the WUP score is used to evaluate Visual QA performance. We eliminate decoys that have higher WUP-based similarity to the target. We use the NLTK toolkit (Bird et al., 2009) to compute the similarity. See the Supplementary Material for more details.

Other details For QoU-decoys, we sort and keep for each triplet the top N (e.g., 10,000) similar triplets from the entire dataset according to the question similarity. Then for each triplet, we compute the WUP-based similarity of each potential decoy to the target successively, and accept those with similarity below 0.9 until we have K decoys. We choose 0.9 according to (Malinowski and Fritz, 2014). We also perform such a check among selected decoys to ensure they are not very similar to each other. For IoU-decoys, the potential decoys are sorted randomly. The WUP-based

[1]E.g., in Fig 1, for the question "What vehicle is pictured?", the only answer that meets both principles is "train", which is the correct answer instead of being a decoy.

similarity with a threshold of 0.9 is then applied to remove ambiguous decoys.

4.2 Comparison to other datasets

Several authors have noticed the design deficiencies in the existing databases and have proposed "fixes" (Antol et al., 2015; Yu et al., 2015; Zhu et al., 2016; Das et al., 2017). No dataset has used a procedure to generate IoU-decoys. We empirically show that how the IoU-decoys significantly remedy the design deficiencies in the datasets.

Several previous efforts have generated decoys that are similar in spirit to our **QoU**-decoys. Yu et al. (2015), Das et al. (2017), and Ding et al. (2016) automatically find decoys from similar questions or captions based on question templates and annotated objects, tri-grams and GLOVE embeddings (Pennington et al., 2014), and paragraph vectors (Le and Mikolov, 2014) and linguistic surface similarity, respectively. The later two are for different tasks from Visual QA, and only Ding et al. (2016) consider removing semantically ambiguous decoys like ours. Antol et al. (2015) and Zhu et al. (2016) ask humans to create decoys, given the questions and targets. As shown earlier, such decoys may disobey the rule of **Neutrality**.

Goyal et al. (2017) augment the VQA dataset (Antol et al., 2015) (by human efforts) with additional IQT triplets to eliminate the shortcuts (language prior) in the open-ended setting. Their effort is complementary to ours on the multiple-choice setting. Note that an extended task of Visual QA, visual dialog (Das et al., 2017), also adopts the latter setting.

5 Empirical Studies

5.1 Dataset

We examine our automatic procedures for creating decoys on five datasets. Table 2 summarizes the characteristics of the three datasets we focus on.

VQA Real (Antol et al., 2015) The dataset uses images from MSCOCO (Lin et al., 2014) under the same training/validation/testing splits to construct IQA triplets. Totally 614,163 IQA triplets are generated for 204,721 images. Each question has 18 candidate answers: in general 3 decoys are human-generated, 4 are randomly sampled, and 10 are randomly sampled frequent-occurring targets. *As the test set does not indicate the targets, our studies focus on the training and validation sets.*

Dataset Name	# of Images			# of triplets			# of decoys per triplet
	train	val	test	train	val	test	
VQA	83k	41k	81k	248k	121k	244k	17
Visual7W	14k	5k	8k	69k	28k	42k	3
VG	49k	19k	29k	727k	283k	433k	-

Table 2: Summary of Visual QA datasets.

Visual7W Telling (Visual7W) (Zhu et al., 2016) The dataset uses 47,300 images from MSCOCO (Lin et al., 2014) and contains 139,868 IQA triplets. Each has 3 decoys generated by humans.

Visual Genome (VG) (Krishna et al., 2017) The dataset uses 101,174 images from MSCOCO (Lin et al., 2014) and contains 1,445,322 IQT triplets. No decoys are provided. Human annotators are asked to write diverse pairs of questions and answers freely about an image or with respect to some regions of it. On average an image is coupled with 14 question-answer pairs. We divide the dataset into non-overlapping 50%/20%/30% for training/validation/testing. Additionally, we partition such that each portion is a "superset" of the corresponding one in Visual7W, respectively.

VQA2 (Goyal et al., 2017) and COCOQA (Ren et al., 2015) We describe the datasets and experimental results in the Supplementary Material.

Creating decoys We create 3 QoU-decoys and 3 IoU-decoys for every IQT triplet in each dataset, following the steps in Sect. 4.1. In the cases that we cannot find 3 decoys, we include random ones from the original set of decoys for VQA and Visual7W; for other datasets, we randomly include those from the top 10 frequently-occurring targets.

5.2 Setup

Visual QA models We utilize the MLP models mentioned in Sect. 3 for all the experiments. We denote **MLP-A**, **MLP-QA**, **MLP-IA**, **MLP-IQA** as the models using A (Answers only), Q+A (Question plus Answers), I+A (Image plus Answers), and I+Q+A (Image, Question and Answers) for multimodal information, respectively. The hidden-layer has 8,192 neurons. We use a 200-layer ResNet (He et al., 2016) to compute visual features which are 2,048-dimensional. The ResNet is pre-trained on ImageNet (Russakovsky et al., 2015). The WORD2VEC feature (Mikolov et al., 2013) for questions and answers are 300-dimensional, pre-trained on Google News[2]. The

[2]We experiment on using different features in the Supplementary Material.

parameters of the MLP models are learned by minimizing the binary logistic loss of predicting whether or not a candidate answer is the target of the corresponding IQA triplet. Please see the Supplementary Material for details on optimization.

We further experiment with a variant of the spatial memory network (denoted as **Attention**) (Xu and Saenko, 2016) and the **HieCoAtt** model (Lu et al., 2016) adjusted for the multiple-choice setting. Both models utilize the attention mechanism. Details are listed in the Supplementary Material.

Evaluation metric For VQA and VQA2, we follow their protocols by comparing the picked answer to 10 human-generated targets. The accuracy is computed based on the number of exactly matched targets (divided by 3 and clipped at 1). For others, we compute the accuracy of picking the target from multiple choices.

Decoy sets to compare For each dataset, we derive several variants: (1) **Orig**: the original decoys from the datasets, (2) **QoU**: **Orig** replaced with ones selected by our QoU-decoys generating procedure, (3) **IoU**: **Orig** replaced with ones selected by our IoU-decoys generating procedure, (4) **QoU +IoU**: **Orig** replaced with ones combining **QoU** and **IoU**, (5) **All**: combining **Orig**, **QoU**, and **IoU**.

User studies Automatic decoy generation may lead to ambiguous decoys as mentioned in Sect. 4 and (Zhu et al., 2016). We conduct a user study via Amazon Mechanic Turk (AMT) to test humans' performance on the datasets after they are remedied by our automatic procedures. We select 1,000 IQA triplets from each dataset. Each triplet is answered by three workers and in total 169 workers get involved. The total cost is $215 — the rate for every 20 triplets is $0.25. We report the average human performance and compare it to the learning models'. See the Supplementary Material for more details.

5.3 Results

The performances of learning models and humans on the 3 datasets are reported in Table 3, 4, and 5[3].

[3]We note that in Table 3, the 4.3% drop of the human performance on IoU +QoU, compared to Orig, is likely due to that IoU +QoU has more candidates (7 per question). Besides, the human performance on qaVG cannot be directly compared to that on the other datasets, since the questions on qaVG tend to focus on local image regions and are considered harder.

Method	Orig	IoU	QoU	IoU +QoU	All
MLP-A	52.9	27.0	34.1	17.7	15.6
MLP-IA	62.4	27.3	55.0	23.6	22.2
MLP-QA	58.2	84.1	40.7	37.8	31.9
MLP-IQA	65.7	84.1	57.6	52.0	45.1
HieCoAtt∗	63.9	-	-	51.5	-
Attntion∗	65.9	-	-	52.8	-
Human	88.4	-	-	84.1	-
Random	25.0	25.0	25.0	14.3	10.0

∗: based on our implementation or modification

Table 3: Test accuracy (%) on Visual7W.

Effectiveness of new decoys A better set of decoys will force learning models to integrate all 3 pieces of information — images, questions and answers — to make the correct selection from multiple-choices. In particular, they should prevent learning algorithms from exploiting shortcuts such that partial information is sufficient for performing well on the Visual QA task.

Table 3 clearly indicates that those goals have been achieved. With the Orig decoys, the relatively small gain from MLP-IA to MLP-IQA suggests that the question information can be ignored to attain good performance. However, with the IoU-decoys which require questions to help to resolve (as image itself is inadequate to resolve), the gain is substantial (from 27.3% to 84.1%). Likewise, with the QoU-decoys (question itself is not adequate to resolve), including images information improves from 40.7% (MLP-QA) substantially to 57.6% (MLP-IQA). Note that with the Orig decoys, this gain is smaller (58.2% vs. 65.7%).

It is expected that MLP-IA matches better QoU-decoys but not IoU-decoys, and MLP-QA is the other way around. Thus it is natural to combine these two decoys. What is particularly appealing is that MLP-IQA improves noticeably over models learned with partial information on the combined IoU +QoU-decoys (and "All" decoys[4]). Furthermore, using answer information only (MLP-A) attains about the chance-level accuracy.

On the VQA dataset (Table 4), the same observations hold, though to a lesser degree. On any of the IoU or QoU columns, we observe substan-

[4]We note that the decoys in Orig are not trivial, which can be seen from the gap between All and IoU +QoU. Our main concern on Orig is that for those questions that machines can accurately answer, they mostly rely on only partial information. This will thus hinder designing machines to fully comprehend and reason from multimodal information. We further experiment on random decoys, which can achieve **Neutrality** but not the other two principles, to demonstrate the effectiveness of our methods in the Supplementary Material.

Method	Orig	IoU	QoU	IoU +QoU	All
MLP-A	31.2	39.9	45.7	31.2	27.4
MLP-IA	42.0	39.8	55.1	34.1	28.7
MLP-QA	58.0	84.7	55.1	54.4	50.0
MLP-IQA	64.6	85.2	65.4	63.7	58.9
HieCoAtt∗	63.0	-	-	63.7	-
Attntion∗	66.0	-	-	66.7	-
Human	88.5†	-	-	89.0	-
Random	5.6	25.0	25.0	14.3	4.2

∗: based on our implementation or modification
†: taken from (Antol et al., 2015)

Table 4: Accuracy (%) on the validation set in VQA.

Method	IoU	QoU	IoU +QoU
MLP-A	29.1	36.2	19.5
MLP-IA	29.5	60.2	25.2
MLP-QA	89.3	45.6	43.9
MLP-IQA	89.2	64.3	58.5
HieCoAtt∗	-	-	57.5
Attntion∗	-	-	60.1
Human	-	-	82.5
Random	25.0	25.0	14.3

∗: based on our implementation or modification

Table 5: Test accuracy (%) on qaVG.

tial gains when the complementary information is added to the model (such as MLP-IA to MLP-IQA). All these improvements are much more visible than those observed on the original decoy sets.

Combining both Table 3 and 4, we notice that the improvements from MLP-QA to MLP-IQA tend to be lower when facing IoU-decoys. This is also expected as it is difficult to have decoys that are simultaneously both IoU and QoU — such answers tend to be the target answers. Nonetheless, we deem this as a future direction to explore.

Differences across datasets Contrasting Visual7W to VQA (on the column IoU +QoU), we notice that Visual7W tends to have bigger improvements in general. This is due to the fact that VQA has many questions with "Yes" or "No" as the targets — the only valid decoy to the target "Yes" is "No", and vice versa. As such decoys are already captured by Orig of VQA ('Yes" and "No" are both top frequently-occurring targets), adding other decoy answers will not make any noticeable improvement. In Supplementary Material, however, we show that once we remove such questions/answers pairs, the degree of improvements increases substantially.

Comparison on Visual QA models As presented in Table 3 and 4, MLP-IQA is on par with or even outperforms **Attention** and **HieCoAtt** on the Orig decoys, showing how the shortcuts make it difficult to compare different models. By eliminating the shortcuts (i.e., on the combined IoU +QoU-decoys), the advantage of using sophisticated models becomes obvious (**Attention** outperforms MLP-IQA by 3% in Table 4), indicating the importance to design advanced models for achieving human-level performance on Visual QA.

Datasets	Decoys	Best w/o using qaVG	qaVG model	
			initial	fine-tuned
Visual7W	Orig	65.7	60.5	**69.1**
	IoU +QoU	52.0	58.1	**58.7**
	All	45.1	48.9	**51.0**
VQA	Orig	64.6	42.2	**65.6**
	IoU +QoU	63.7	47.9	**64.1**
	All	58.9	37.5	**59.4**

Table 6: Using models trained on qaVG to improve Visual7W and VQA (Accuracy in %).

For completeness, we include the results on the Visual Genome dataset in Table 5. This dataset has no "Orig" decoys, and we have created a multiple-choice based dataset **qaVG** from it for the task — it has over 1 million triplets, the largest dataset on this task to our knowledge. On the combined IoU +QoU-decoys, we again clearly see that machines need to use all the information to succeed.

With **qaVG**, we also investigate whether it can help improve the multiple-choice performances on the other two datasets. We use the MLP-IQA trained on qaVG with both IoU and QoU decoys to initialize the models for the Visual7W and VQA datasets. We report the accuracies before and after fine-tuning, together with the best results learned solely on those two datasets. As shown in Table 6, fine-tuning largely improves the performance, justifying the finding by Fukui et al. (2016).

5.4 Qualitative Results

In Fig. 2, we present examples of image-question-target triplets from **V7W**, **VQA**, and **VG**, together with our IoU-decoys (A, B, C) and QoU-decoys (D, E, F). G is the target. The predictions by the corresponding MLP-IQA are also included. Ignoring information from images or questions makes it extremely challenging to answer the triplet correctly, even for humans.

Our automatic procedures do fail at some triplets, resulting in ambiguous decoys to the targets. See Fig. 3 for examples. We categorized those failure cases into two situations.

- Our filtering steps in Sect. 4 fail, as observed in the top example. The WUP-based similarity re-

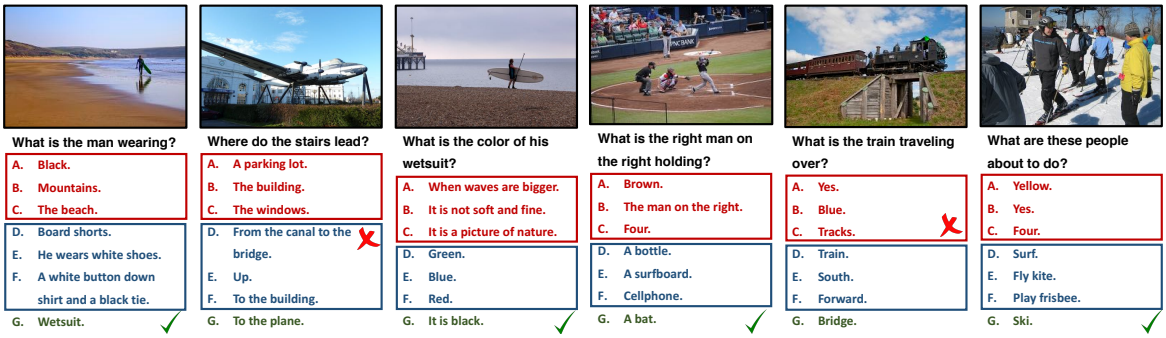


Figure 2: Example image-question-target triplets from Visual7W, VQA, and VG, together with our IoU-decoys (A, B, C.) and QoU-decoys (D, E, F). G is the target. Machine's selections are denoted by green ticks (correct) or red crosses (wrong).

Figure 3: Ambiguous examples by our IoU-decoys (A, B, C) and QoU-decoys (D, E, F). G is the target. Ambiguous decoys F are marked.

lies on the WordNet hierarchy. For some semantically similar words like "lady" and "woman", the similarity is only 0.632, much lower than that of 0.857 between "cat" and "dog". This issue can be alleviated by considering alternative semantic measures by WORD2VEC or by those used in (Das et al., 2017; Ding et al., 2016) for searching similar questions.

- The question is ambiguous to answer. In the bottom example in Fig. 3, both candidates D and F seem valid as a target. Another representative case is when asked about the background of a image. In images that contain sky and mountains in the distance, both terms can be valid.

6 Conclusion

We perform detailed analysis on existing datasets for multiple-choice Visual QA. We found that the design of decoys can inadvertently provide "shortcuts" for machines to exploit to perform well on the task. We describe several principles of constructing good decoys and propose automatic procedures to remedy existing datasets and create new ones. We conduct extensive empirical studies to demonstrate the effectiveness of our methods in creating better Visual QA datasets. The remedied datasets and the newly created ones are released and available at http://www.teds.usc.edu/website_vqa/.

Acknowledgments

This work is partially supported by USC Graduate Fellowship, NSF IIS-1065243, 1451412, 1513966/1632803, 1208500, CCF-1139148, a Google Research Award, an Alfred. P. Sloan Research Fellowship and ARO# W911NF-12-1-0241 and W911NF-15-1-0484.

References

Aishwarya Agrawal, Dhruv Batra, and Devi Parikh. 2016. Analyzing the behavior of visual question answering models. In *EMNLP*.

Aishwarya Agrawal, Dhruv Batra, Devi Parikh, and Aniruddha Kembhavi. 2018. Don't just assume; look and answer: Overcoming priors for visual question answering. In *CVPR*.

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. Spice: Semantic propositional image caption evaluation. In *ECCV*.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *ICCV*.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. 2015. Microsoft coco captions:

Data collection and evaluation server. *arXiv preprint arXiv:1504.00325* .

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José MF Moura, Devi Parikh, and Dhruv Batra. 2017. Visual dialog. In *CVPR*.

Nan Ding, Sebastian Goodman, Fei Sha, and Radu Soricut. 2016. Understanding image and text simultaneously: a dual vision-language machine comprehension task. *arXiv preprint arXiv:1612.07833* .

Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. 2010. Every picture tells a story: Generating sentences from images. In *ECCV*.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding. In *EMNLP*.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *CVPR*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

Micah Hodosh and Julia Hockenmaier. 2016. Focused evaluation for image description with binary forced-choice tasks. In *ACL Workshop*.

Micah Hodosh, Peter Young, and Julia Hockenmaier. 2013. Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artificial Intelligence Research* .

Allan Jabri, Armand Joulin, and Laurens van der Maaten. 2016. Revisiting visual question answering baselines. In *ECCV*.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*.

Kushal Kafle and Christopher Kanan. 2017a. An analysis of visual question answering algorithms. In *ICCV*.

Kushal Kafle and Christopher Kanan. 2017b. Visual question answering: Datasets, algorithms, and future challenges. *Computer Vision and Image Understanding* .

Mert Kilickaya, Aykut Erdem, Nazli Ikizler-Cinbis, and Erkut Erdem. 2017. Re-evaluating automatic metrics for image captioning. In *EACL*.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *IJCV* .

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *ECCV*.

Xiao Lin and Devi Parikh. 2016. Leveraging visual question answering for image-caption ranking. In *ECCV*.

Xiao Lin and Devi Parikh. 2017. Active learning for visual question answering: An empirical study. *arXiv preprint arXiv:1711.01732* .

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *EMNLP*.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image co-attention for visual question answering. In *NIPS*.

Mateusz Malinowski and Mario Fritz. 2014. A multi-world approach to question answering about real-world scenes based on uncertain input. In *NIPS*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

Vicente Ordonez, Girish Kulkarni, and Tamara L Berg. 2011. Im2text: Describing images using 1 million captioned photographs. In *NIPS*.

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

Mengye Ren, Ryan Kiros, and Richard Zemel. 2015. Exploring models and data for image question answering. In *NIPS*.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet large scale visual recognition challenge. *IJCV* .

Qi Wu, Damien Teney, Peng Wang, Chunhua Shen, Anthony Dick, and Anton van den Hengel. 2017. Visual question answering: A survey of methods and datasets. *Computer Vision and Image Understanding* .

Zhibiao Wu and Martha Palmer. 1994. Verbs semantics and lexical selection. In *ACL*.

Huijuan Xu and Kate Saenko. 2016. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In *ECCV*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*.

Licheng Yu, Eunbyung Park, Alexander C Berg, and Tamara L Berg. 2015. Visual madlibs: Fill in the blank description generation and question answering. In *ICCV*.

Peng Zhang, Yash Goyal, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2016. Yin and yang: Balancing and answering binary visual questions. In *CVPR*.

Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. 2016. Visual7w: Grounded question answering in images. In *CVPR*.

Abstract Meaning Representation for Paraphrase Detection

Fuad Issa[†] **Marco Damonte**[†] **Shay B. Cohen**[†] **Xiaohui Yan**[‡] **Yi Chang**[‡]
[†] School of Informatics, University of Edinburgh, 10 Crichton Street, Edinburgh EH8 9AB, UK
[‡] Huawei Technologies, San Jose, CA 95050, USA
issa.fuad@gmail.com m.damonte@sms.ed.ac.uk
scohen@inf.ed.ac.uk {yanxiaohui2,yi.chang}@huawei.com

Abstract

Abstract Meaning Representation (AMR) parsing aims at abstracting away from the syntactic realization of a sentence, and denoting only its meaning in a canonical form. As such, it is ideal for paraphrase detection, a problem in which one is required to specify whether two sentences have the same meaning. We show that naïve use of AMR in paraphrase detection is not necessarily useful, and turn to describe a technique based on latent semantic analysis in combination with AMR parsing that significantly advances state-of-the-art results in paraphrase detection for the Microsoft Research Paraphrase Corpus. Our best results in the transductive setting are 86.6% for accuracy and 90.0% for F_1 measure.

1 Introduction

Abstract Meaning Representation (AMR) parsing focuses on the conversion of natural language sentences into AMR graphs, aimed at abstracting away from the surface realizations of the sentences while preserving their meaning.

We make a first step towards showing that AMR can be used in practice for a task that requires identifying the canonicalization of language: paraphrase detection. In a "perfect world" using AMR to test for paraphrasing relation of two sentences should be simple. It would require finding the two AMR parses for each of the sentences, and then checking whether they are identical. Since AMR is aimed at abstracting away from the surface form which is used to express meaning, two sentences should be paraphrases only if they have identical AMRs. For instance, the three sentences:

1. He described her as a curmudgeon,
2. His description of her: curmudgeon,
3. She was a curmudgeon, according to his description.

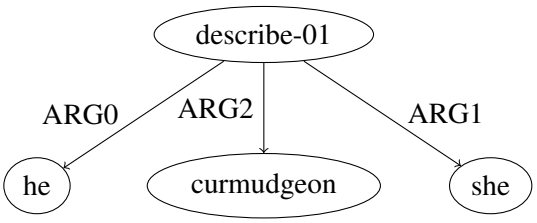


Figure 1: AMR graph for "He described her as a curmudgeon", "His description of her: curmudgeon" and "She was a curmudgeon, according to his description"

should result in the same AMR graph as shown in Figure 1.

However, in practice, things are different. First, there are no known AMR parsers that really distil only the meaning in text. For example, predicates which have interchangeable meaning use different AMR concepts, and there are errors that exist because of the machine learning techniques that are used for learning the parsers from data. Finally, even human annotations do not yield perfect AMRs, as the interannotator agreement reported in the literature for AMR is around 80% (Banarescu et al., 2013).

Second, meaning is often contextual, and it is not fully possible to determine the corresponding AMR parse just by looking at a given sentence. Entity mentions denote different entities in different contexts, and similarly predicates and nouns are ambiguous and depend on context. As such, one cannot expect to use AMR in the transparent way mentioned above to identify paraphrase relations. However, we demonstrate in this paper that AMR can be used in a "softer" way to detect such relations.

Evaluation of AMR parsers is traditionally performed using the Smatch score (Cai and Knight, 2013). However, Damonte et al. (2017) argue that more ad-hoc metrics can be useful for advancing AMR research. Paraphrase detection can be seen

Proceedings of NAACL-HLT 2018, pages 442–452
New Orleans, Louisiana, June 1 - 6, 2018.

as a further benchmark for AMR parsers, highlighting their ability of abstracting away from syntax and representing the core concepts expressed in the sentence. In order to advance research in AMR and its applications, it is important to have metrics that reflect on the ability of AMR graphs to have impact on subsequent tasks. In this work we therefore use two different AMR parsers, comparing them throughout all experiments.

2 Background

AMRs are rooted, edge labeled, node labeled, directed graphs. They are biased towards the English language and rely on PropBank (Kingsbury and Palmer, 2002) for the definition of the main events in the sentence. Nodes in an AMR graph represent events and concepts, while edges represent the relationships between them. Banarescu et al. (2013) state that AMR are aimed at canonicalizing multiple ways of expressing the same idea, which could be of great assistance to solve the problem of paraphrase detection. However, this goal is not entirely achieved in practice, and it will take long for AMR parsers to mature and achieve such canonicalization. At the moment, for example, even a simple pair of sentences such as "the boy desires the cake" and the "the boy wants the cake" would not have the same canonical form by state-of-the-art AMR parsers.

While some researchers (Fodor, 1975) have doubted the practical possibility of canonicalizing language or finding identical paraphrases in English or otherwise, much work in NLP has been devoted to the problem of paraphrase identification (Mitchell and Lapata, 2010; Baroni and Lenci, 2010; Socher et al., 2011; Guo and Diab, 2012; Ji and Eisenstein, 2013) and more weakly, finding entailment between sentences and phrases (Dagan et al., 2006; Bos and Markert, 2005; Harabagiu and Hickl, 2006; Lewis and Steedman, 2013). In this work, we use the AMRs parsed for given sentences as a mean to extract useful information and train paraphrase detection classifiers on top of them.

2.1 Latent Semantic Analysis

Our work falls under the category of distributional methods for paraphrase detection (Turney and Pantel, 2010; Mihalcea et al., 2006; Mitchell and Lapata, 2010; Guo and Diab, 2012; Ji and Eisenstein, 2013) such as with latent semantic analysis (LSA, Landauer et al., 1998). The main principle behind this approach is to detect semantic similarity through distributional representations for a given sentence and its potential paraphrase, where these representations are compared against each other according to some similarity metric or used as features with a discriminative classification method (Mihalcea et al., 2006; Guo and Diab, 2012; Ji and Eisenstein, 2013).

LSA is indeed one of the main tools in obtaining such distributional representations for the problem of paraphrase detection. Most often, TF-IDF weighting has been used for building the sentence-term matrix, but Ji and Eisenstein (2013) have shown that a significant improvement can be achieved in detecting similarity if one re-weights the sentence-term matrix differently. Indeed, this is one of our main contributions: we build on previous work on LSA for paraphrase detection and propose a technique to re-weight a *sentence-concept* matrix based on the AMR graphs for the given sentences. More details on the use of LSA for paraphrase detection appear in Section 4.

2.2 AMR Parsing

AMR parsing is the task of converting natural language sentences into AMR graphs, which are Directed Acyclic Graphs (DAGs) in all cases except a few rare controversial cases. This task embeds several common NLP problems together, such as named entity recognition, sentential-level coreference resolution, semantic role labeling and word-sense disambiguation. Several parsers for AMR have been recently developed (Flanigan et al., 2014; Wang et al., 2015; Peng et al., 2015; Pust et al., 2015; Goodman et al., 2016; Rao et al., 2015; Vanderwende et al., 2015; Artzi et al., 2015; Barzdins and Gosko, 2016a; Zhou et al., 2016; Damonte et al., 2017; Barzdins and Gosko, 2016b; Konstas et al., 2017). Shared tasks were also organized in order to push forward the state-of-the-art (May, 2016; May and Priyadarshi, 2017).

Meaning representations are usually evaluated based on their compositionality (construction of a representation based on parts of the text in a consistent way), verifiability (ability to check whether a meaning representation is true in a given model of the world), unambiguity (ability to full disambiguate text into the representation in a way that does not leave any ambiguity lingering), inference (the existence of a calculus that can be used to

infer whether one meaning representation is logically implied by others) and canonicalization (the ability to map several surface forms, such as paraphrases, into a single unique meaning representation). In this paper, we evaluate AMR on its ability to canonicalize language through its assistance in deciding whether two sentences are paraphrases.

We note that this test is masked by the accuracy of the AMR parsers we use, which indeed do not give always fully correct predictions. These errors in our paraphrase detection due to the accuracy of the AMR parser are different than those which originate in an inherent difficulty of representing paraphrases using AMR because of the limitations of the formalism and the annotation guidelines that AMR follows.

We experiment with two AMR parsers for which a public version is available. The first is JAMR (Flanigan et al., 2014), which is a graph-based approach to AMR parsing. It works by performing two steps on the input sentence: concept identification and relation identification. The former discovers the concept fragments corresponding to span of words in the sentence, while the latter finds the optimal spanning connected subgraph from the concepts identified in the first step. The concept identification step has quadratic complexity and the relation identification step is $O(|V|^2 \log |V|)$, with $|V|$ being the set of nodes in the AMR graph.

The second is AMREager (Damonte et al., 2017), which is a transition-based parser that works by scanning the string left-to-right and building the graph as the scan proceeds. This transition-based system is akin to the dependency parsing transition-system ArcEager of Nivre (2004), only without constraints that ensure that the resulting structure is a tree. In addition, there are operations that make the system create additional non-projective structures by checking after transition step whether siblings should be connected together with an edge. The complexity of AMREager is linear in the length of the sentence. AMREager was extended to other languages (Damonte and Cohen, 2018), and we leave it for future work to test the utility of AMR for paraphrase detection in these languages.

3 Problem Formulation

Let $\mathcal{S}$ be a set of sentences. We are given input data in the form of $(x_1^{(i)}, x_2^{(i)}, b^{(i)})$ for $i \in [n]$ where n is the number of training examples, $x_j^{(i)} \in \mathcal{S}$, $j \in \{1, 2\}$ and $b^{(i)} \in \{0, 1\}$ is a binary indicator that tells whether $x_1^{(i)}$ is a paraphrase of $x_2^{(i)}$.

The goal is to learn a classifier

$$c \colon \mathcal{S} \times \mathcal{S} \rightarrow \{0, 1\}$$

that tells for unseen instances whether the pair of sentences given as input are paraphrases of each other. We denote by $[n]$ the set $\{1, \ldots, n\}$.

4 Latent Semantic Analysis for Paraphrase Detection

The first step in our approach is the construction of lower-dimensional representations for the sentences in the training data. We use latent semantic analysis to get the sentence representations, which are then used to detect paraphrases using a classifier. More specifically, given a set of sentences $S = \{x_j^{(i)} \mid j \in \{1, 2\}, i \in [n]\}$, we build a sentence-term matrix T such that $T_{k\ell}$ indicates the use of the ℓth word in the kth sentence in S. The number of rows is the number of sentences in the dataset and the number of columns is the vocabulary size. This follows previous work with the use of LSA for paraphrasing (Guo and Diab, 2012; Ji and Eisenstein, 2013).

As a baseline, we experiment with two ways of assigning the values to the matrix:

- $T_{k\ell}$ is the count of the ℓth word in the kth sentence:

$$T_{k\ell} = \text{count}(\ell, k)$$

- $T_{k\ell}$ is the term frequency-inverse document frequency (TF-IDF) for the kth sentence with respect to the ℓth word. TF-IDF is commonly used in Information Retrieval to score words in a document and combines the frequency of the words in a document with the rarity of the term across documents. With TF-IDF, in order to have a high score, concepts must appear in this sentence and not in many others. In that case, we define:

$$T_{k\ell} = \text{count}(\ell, k) \times \frac{n}{\text{csent}(\ell, k)}$$

where $\text{count}(\ell, k)$ gives the count of the ℓth word in the kth sentence and csent is the

number of sentences which contain the ℓth word:

$$\text{csent}(\ell, k) = |\{k \in [|S|] : \text{count}(k, \ell) > 0\}|.$$

The AMR-based systems of Section 5 build upon this by re-weighting $T_{k\ell}$ with terms depending on the AMRs of the sentences.

For paraphrasing, previous work (Ji and Eisenstein, 2013) has also considered the *transductive* setting (Gammerman et al., 1998), which we also use in our experiments. In the transductive setting, S also includes the sentences on which we expect to perform the final evaluation for the purpose of learning the latent representations. Note that, in this case, the labels $b^{(i)}$ are not used in the process of constructing word representations. In the inductive setting, on the other hand, the sentences in the testing set are not included in training and we project them instead using the LSA projection matrices onto the latent space learned to find their representations.

Once we constructed the matrix T, we perform truncated singular value decomposition (SVD) on it, such that:

$$T \approx U \Sigma V^\top$$

where $U \in \mathbb{R}^{k \times m}$, $V \in \mathbb{R}^{\ell \times m}$ and $\Sigma \in \mathbb{R}^{m \times m}$ is a diagonal matrix of singular values. The final sentence representations are the rows of the U matrix which range over the sentences and have m dimensions.

The output of this process is a function $f : \mathcal{S} \to \mathbb{R}^m$ which attaches to each sentence a representation. The idea behind LSA is that this matrix decomposition will make semantically similar sentences to appear close in the latent space, hence alleviating the problem of data sparsity and making it easier to detect when two sentences are paraphrases of each other.

Once we construct the sentence representations from the training data (either in the inductive or the transductive setting) we use the function f to map each pair of sentences from the training data $(x_1^{(i)}, x_2^{(i)})$ to two vectors $f(x_1^{(i)}) + f(x_2^{(i)})$ and $|f(x_1^{(i)}) - f(x_2^{(i)})|$ (where the absolute value is taken coordinate-wise) and then concatenate them into a feature vector $\phi(x_1^{(i)}, x_2^{(i)})$, which is then used as input to a support vector machine (SVM) classifier (Ji and Eisenstein, 2013).[1]

5 Abstract Meaning Representation Features

The main hypothesis tested in this work is that AMR can be useful in deciding whether two sentences are paraphrases of each other. We investigate two ways to use AMR information to better inform the classifier: similarity-based and LSA-based.

5.1 Graph Similarity and Bag of AMR Concepts

An obvious way to use AMR information is to just compute the similarity between the two graphs and use the score as an additional feature. As a score we use Smatch, which computes the overlap in terms of recall, precision and F-score between two unaligned graphs by finding the alignments between the graphs that maximizes the overlap. The alignment step is necessary because in AMR multiple nodes can have the same labels and arbitrary variable names are used to distinguish between them. Smatch is the standard metric to evaluate the overlap between AMR graphs. The score returned by Smatch is used as a single additional feature for the SVM.

The amount of overlap in the AMR nodes of the two graphs can be a good indicator of whether the sentences are paraphrases of each other. To test this hypothesis, we extract the unordered sets of AMR nodes and use the Jaccard similarity coefficient as a feature. This is directly related to the concept identification step of the AMR parsing process, which is concerned with generating and labeling the nodes of the AMR graph. Concept identification is arguably one of the most challenging part of AMR parsing as the mapping between word spans and AMR nodes is not trivial (Werling et al., 2015). It is often considered as the first stage in the AMR parsing pipeline and it is therefore reasonable to attempt using its intermediate results. We choose Jaccard as a metric for bag of concepts overlap following previous work in paraphrase detection (Achananuparp et al., 2008; Be-

[1] We note that while the NLP community has largely switched to the use of neural networks for classification problems, in our case support vector machines prove to be a simpler and more efficient solution. They also tend to generalize better than neural networks, as the number of features we use is not large.

rant and Liang, 2014).

We note that while this approach of using AMR to detect paraphrase may sound plausible, it does not perform very well. As such, we compare and contrast this as an AMR baseline with the approach that makes use of PageRank with TF-IDF reweighting for LSA, as described next.

5.2 PageRank and TF-IDF Reweighting for LSA

The main idea is to re-weight the LSA sentence-term matrix T (Section 4) according to a probability distribution over the AMR nodes, which we accomplish by means of PageRank (Page et al., 1999). The utility of re-weighting terms in the sentence-term matrix has been previously proved (Turney and Pantel, 2010). PageRank is a method, originally developed for web pages, for ranking nodes in a graph according to their impact on other nodes. The algorithm works iteratively by adjusting at each iteration the score of each node based on the number and scores of nearby nodes that is connected to it, until convergence. Prior to applying PageRank, we merge the two graphs by collapsing the concepts in the two graphs that have the same labels, similarly to Liu et al. (2015), as shown in Figure 2. We then compute the PageRank score for each node in the merged graph and multiply them by the corresponding frequency count of that concept in the sentence-term matrix. The graph merging step is necessary in order to ensure that overlapping concepts obtain high PageRank scores. The PageRank step applied to the merged graph ensures that this importance propagates to nearby nodes.

For a given graph $G = (V, E)$, PageRank takes as input a list of edges between nodes:

$$E = \{(n_i, m_i)\}, \forall i = 0, \ldots, n$$
$$n = |E|$$

and outputs a PageRank score for each node by solving the following equations with respect to $\mathrm{PG}(\cdot)$:

$$\mathrm{PG}(n) = \sum_{m \in I(n)} \frac{\mathrm{PG}(m)}{l(m)}$$

where $I(n)$ are the input edges to node n and $l(m)$ is the number of edges coming out of m.

For each concept of the merged AMR graph, we compute $T_{k\ell}$, the weight for the LSA matrix introduced in Section 4, as follows:

$$T_{k\ell} = \mathrm{PG}(l, k) \times \mathrm{count}(l, k)$$

where $\mathrm{PG}(l, k)$ is the PageRank of ℓth concept for the kth sentence.

As a baseline for the PageRank system, the TF-IDF re-weighting scheme, as described in Section 4, is also used to re-weight the AMR concepts.

6 Experiments

We now describe the experiments that we devised to discover whether AMR is useful for paraphrase detection. For AMR parsing, we used the JAMR[2] version published for SemEval 2016 (Flanigan et al., 2016), reporting 0.67 Smatch score on LDC2015E86 and the first and only version available for AMREager,[3] obtaining 0.64 Smatch score on the same dataset. First, we discuss experiments where the AMRs are used as a mean to extract additional sparse features for a SVM classifier. Then we turn to LSA to construct a representation of the sentence based on the reweighting on the AMR nodes achieved through either PageRank or TF-IDF. Results show how the latter, which builds on state-of-the-art systems for this task, is a much more promising approach. Finally, we analyze how performance changes as a function of the number of dimensions used in the truncated matrix.

For evaluation, we use the Microsoft Research Paraphrase Corpus (Dolan et al., 2004). We use 70% of the dataset as training data and 30% as a test set. The total number of sentence pairs in the corpus is 5,801.

6.1 Graph Similarity and Bag of AMR Concepts

The Bag of words (BOW) baseline consists of a SVM that takes into account one single feature: the Jaccard score between the BOW representations for the two sentences, i.e., one-hot vectors indicating whether each word in the vocabulary is used or not. The use of the single Jaccard feature means that for the linear kernel we just learn a threshold on the score.

We note that the addition of the similarity-based features does not suffice to outperform the BOW baseline, as described in Table 1. Unlike Smatch, the bag of concepts feature does not need to find a, possibly wrong, alignment between the two graphs

[2]JAMR is available from https://github.com/jflanigan/jamr.

[3]AMREager is available from http://cohort.inf.ed.ac.uk/amreager.html.

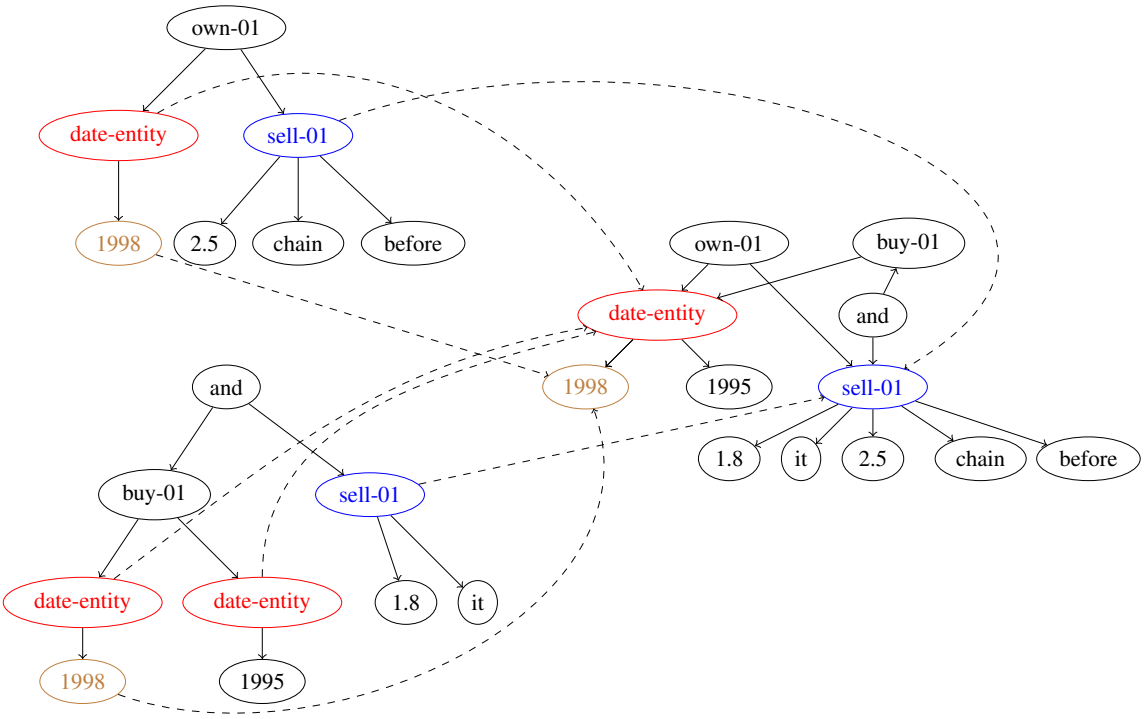


Figure 2: Visualization of the graph merging procedure for the sentence *Yucaipa owned Dominick's before selling the chain to Safeway in 1998 for $2.5 billion.* (above) and *Yucaipa bought Dominick's in 1995 for $693 million and sold it to Safeway for $1.8 billion in 1998.* (below). The "date-entity", "sell-01" and "1998" nodes in the two AMR graphs on the left are merged in the resulting graph on the right.

because it considers the node labels only. Interestingly, the addition of the bag of concepts feature is beneficial only for AMREager. It is indeed worth noting the different behaviors of the two parsers: when using the Smatch score only, JAMR reports slightly higher numbers than AMREager. However, when using the bag of concepts features too, AMREager is considerably better than JAMR, which is unexpected as the concept identification performance of the two parsers is reported to be identical (Damonte et al., 2017).

There is also some variability with the kernel used for the SVM classifier. The polynomial kernel does consistently better than the RBF and linear kernel. This means that a low-level interaction between the sentence representations does exist (when trying to determine whether they are paraphrases), but a higher order interaction, such as implied with RBF, is not necessary to be modeled.

6.2 PageRank and TF-IDF Reweighting for LSA

We now turn to experiments involving LSA as a mean to represent the candidate paraphrases. In this set of experiments, the baseline consists of using TF-IDF to weight the bag of words in the sentence-term matrix.

We first try to replace the bag of words with the bag of concepts from the AMR graphs, also re-weighted by TF-IDF. Then, we also replace the TF-IDF with PageRank as it is more appropriate to re-weight graph structures than TF-IDF. We report experiments for both inductive setting and transductive setting (Table 3). Our first finding is that, regardless of the parser, AMR is very helpful in the tranductive setting while it is harmful in the inductive setting. When using bag of words, it is easy to project sentences of the test set into the latent space learned on the training set only. However, our experiments indicate that this is not as easy with the AMR concepts produced by the two parsers. On the other hand, when the latent space is learned using also the sentences in the test set, the abstractive power of AMRs is helpful for this task. In the inductive setting, PageRank fails to improve over the TF-IDF scheme and neither of them outperform the BOW baseline. AMREager outperforms JAMR in this case. In the transduc-

		kernel	acc.	P	R	F_1
Smatch + BOC	AMREager	linear	65.8	81.8	62.9	71.1
		poly	**68.5**	75.4	78.3	76.8
		rbf	65.0	84.0	58.8	69.2
	JAMR	linear	63.4	79.4	61.0	69.0
		poly	64.9	76.0	69.3	72.5
		rbf	61.8	82.0	55.3	65.9
Smatch	AMREager	linear	63.8	79.4	61.7	69.5
		poly	67.8	72.0	84.7	77.8
		rbf	61.6	81.2	55.3	65.8
	JAMR	linear	63.5	77.8	63.5	69.9
		poly	68.1	71.6	**86.5**	78.3
		rbf	62.0	0.80	57.4	66.9
	BOW	linear	68.1	**84.1**	64.3	72.9
		poly	72.7	77.8	82.7	**80.2**
		rbf	68.1	**84.1**	64.3	72.9

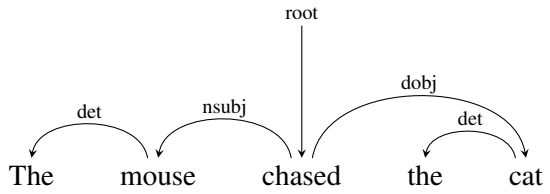
Table 1: Baseline results for paraphrase detection with AMR and with bag-of-words (BOW). "linear," "poly" and "rbf" denote the kernel which is used with a support vector machine classifier. "Smatch" denotes the use of the additional graph similarity feature and "BOC" the use of the additional Jaccard score on the bag of concept. Best result in each column is in bold.

tive case, the AMRs provided by JAMR are helpful with both TF-IDF and PageRank, while the graphs provided by AMREager give good results only for the PageRank scheme. The best result is achieved with JAMR, PageRank and a linear kernel for the SVM classifier.

We wanted to test in our experiments whether the same gains that are achieved with AMR parsing can also be achieved with just a syntactic parser. To test that, we parsed the paraphrase dataset with a dependency parser and reduced the syntactic parse trees to AMR graphs (meaning, we represented the dependency trees as graphs by representing each word as a node and labeled dependency relations as edges). Figure 3 gives an example of such conversion. As can be see, the AMR-like representation for the dependency trees retains words such as determiners ("the"). It also uses a different set of relations, as reflected by the edge labels that the dependency parser returns.

We chose to do this reduction instead of directly building a classifier that makes use of the dependency trees to ensure we are conducting a controlled experiment in which we precisely compare the use of syntax for paraphrase against the use of semantics. Once the syntactic trees are converted

(a)



(b)

(**c** / *chased*
nsubj (**m** / *mouse*
det (**t1** / *The*))
dobj (**c** / *cat*
det (**t2** / *the*))
)

Figure 3: An example of a dependency tree (a) converted to an AMR graph (b).

to AMR graphs, the same code is used to run the experiments as in the case of AMR parsing, with both the PageRank and TF-IDF reweighting settings. We used the dependency parser from the Stanford CoreNLP (Manning et al., 2014). The results are given in Table 3, under "dep." As can be seen, these results lag behind the bag-of-words model in the inductive case and the AMR models in the transductive case. This could be attributed to AMR parsers better abstracting away from the surface form than dependency parsers.

System	acc.	F_1
Most common class	66.5	79.9
Mitchell and Lapata (2010)	73.0	82.3
Baroni and Lenci (2010)	73.5	82.2
Socher et al. (2011)	76.8	83.6
Guo and Diab (2012)	71.5	NR
Ji and Eisenstein (2013) (ind.)	80.0	85.4
Ji and Eisenstein (2013) (trans.)	80.4	86.0
Our paper (inductive)	68.7	80.9
Our paper (transductive)	86.6	90.0

Table 2: Comparison of our results with previous work ("NR" stands for "not reported"). All work mentioned above was done in the inductive setting, except for Ji and Eisenstein (2013), which, like us, was done in both settings.

6.3 Dimensionality of the Truncated Matrix

Figure 4 shows how performance changes as a function of the number of dimensions used in the

			inductive				transductive			
		kernel	acc.	P	R	F_1	acc.	P	R	F_1
PageRank	AMREager	linear	68.7	73.1	84.0	78.2	79.4	79.9	92.6	85.7
		poly	67.6	67.9	97.6	80.1	67.0	67.3	**98.6**	80.0
		rbf	68.0	75.0	78.1	76.5	79.6	84.8	84.5	84.7
	JAMR	linear	59.3	70.4	67.4	68.9	**86.6**	88.3	92.0	**90.0**
		poly	66.9	67.6	96.7	79.6	80.0	69.6	98.1	81.5
		rbf	67.4	73.5	79.9	76.6	**86.6**	**90.4**	89.5	89.9
	dep.	linear	62.4	71.6	72.5	72.1	79.0	83.2	85.0	84.1
		poly	68.3	69.0	95.3	80.1	74.0	77.6	85.1	81.2
		rbf	68.8	71.0	90.0	79.4	77.0	89.4	73.9	80.9
TFIDF	AMREager	linear	68.9	73.4	83.7	78.2	73.0	75.8	87.1	81.1
		poly	68.7	68.7	98.2	80.9	68.0	68.2	98.5	80.6
		rbf	68.3	77.9	73.4	75.6	72.0	81.2	75.6	78.3
	JAMR	linear	57.6	68.8	66.8	67.8	82.0	84.7	88.5	86.5
		poly	67.4	67.8	97.4	79.9	82.0	84.7	88.5	86.5
		rbf	69.1	76.7	77.2	76.8	85.0	88.0	88.9	88.4
	dep.	linear	70.3	74.2	85.1	79.3	69.0	73.8	82.8	78.1
		poly	68.8	68.7	97.8	80.7	68.0	68.2	98.2	80.5
		rbf	**70.6**	79.1	76.0	77.5	70.0	79.3	75.5	77.4
	BOW	linear	71.9	75.4	86.0	80.4	73.0	75.8	87.7	81.3
		poly	70.5	69.8	**98.3**	**81.6**	71.0	69.9	98.1	81.7
		rbf	70.5	**81.3**	72.5	76.6	73.0	82.5	75.7	79.0

Table 3: LSA experiments in the inductive and transductive settings, with two different reweighting schema: "PageRank" and "TF-IDF". "linear," "poly" and "rbf" denote the kernel for the SVM. "dep." denotes the use of syntactic parsing instead of semantic parsing.

truncated matrix U (Section 4). More specifically, on the x axis of the plots we have m/l, where m is the number of columns in the truncated matrix and l the number of words in the vocabulary. The plot shows that the performance stays stable for inductive inference. With transductive inference, however, performance peaks when m is very close to the vocabulary size. This shows that, in order to achieve good results, it is not necessary to remove a large number of columns from the original sentence-term matrix. The plot gives us more evidence on how the inductive setting is not ideal for the AMR-based approach. For the TF-IDF reweighting, the systems that show a considerably different behavior are JAMR with linear and RBF kernels, where we show clear peaks for the transductive case. For PageRank also the AMREager systems with linear and RBF kernel follow this trend. In general the polynomial kernel is the one less affected by this variable.

Table 2 shows that our best result for the transductive case, which we obtain with JAMR and PageRank, outperforms the current state of the art for paraphrase detection in the transductive setting. This is not true for the inductive case, proving the preference of the AMR-based LSA approach for the former setting.

7 Conclusion

We described an approach to incorporate an AMR parser output into the detection of paraphrases. Our method works by merging two graphs that need to be tested for a paraphrase relation, and then re-weighting a sentence-term matrix by the PageRank values of the nodes in the merged graph. We find that our method gives significant improvements over state of the art in paraphrase detection in the transductive setting, showing that AMR is indeed helpful for this task. We further show that the inductive settings is instead not ideal for this type of approach.

We are encouraged by the results, and believe that paraphrase detection can also be used as a proxy test for the performance of an AMR parser: if an AMR parser is close to canonicalizing language, it should be of significant help in detecting

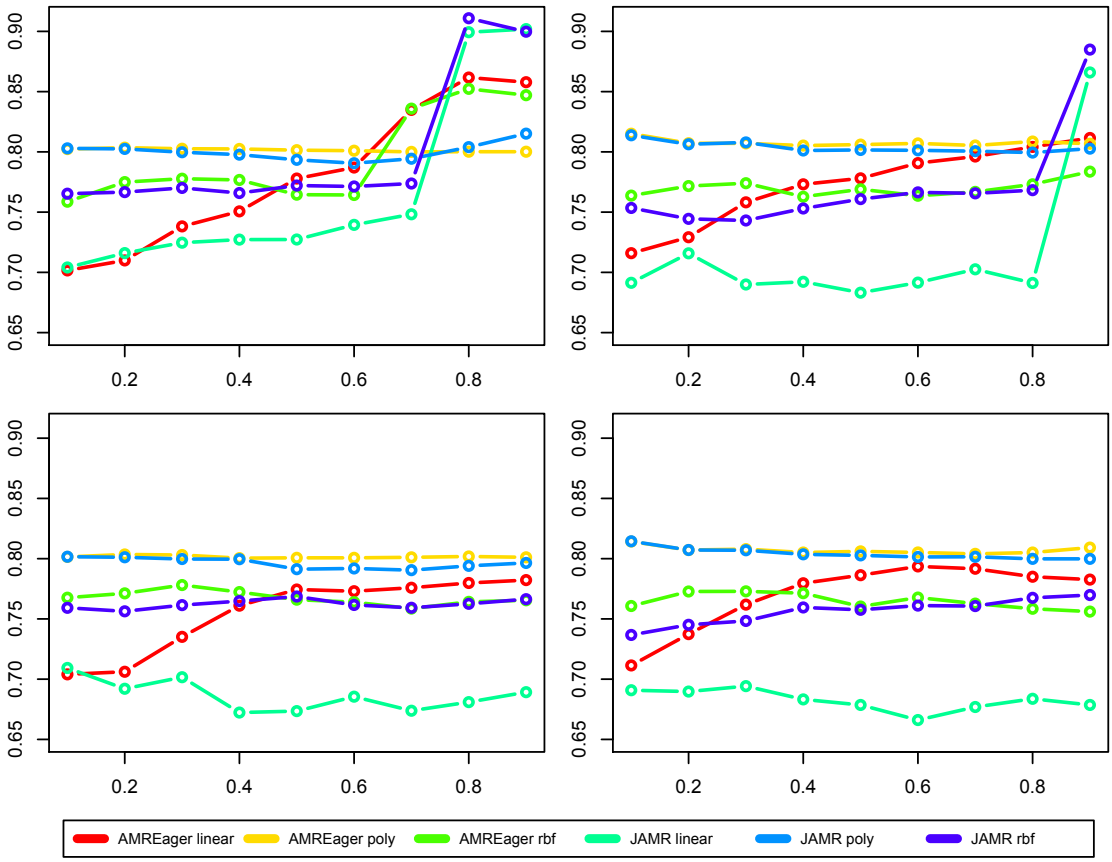


Figure 4: Plots of F_1 measure as a function of m/l, where m is the number of columns in the truncated matrix and l the number of words in the vocabulary. The top plots are in the transductive setting (with the left using PageRank and the right using TF-IDF weighting) while the bottom plots are in the inductive setting.

paraphrase relations. In our experiments, the overall best result was achieved by JAMR. More generally, our results show that JAMR has been more helpful in the transductive setting and in the first set of experiment when using the Smatch score only, while AMREager wins the comparison in the inductive case as well as in the first set of experiments when using both the Smatch score and the bag of concepts score as additional features.

Acknowledgments

The authors would like to thank the three anonymous reviewers for their helpful comments. This research was supported by a grant from Bloomberg, a grant from Huawei Technologies and by the EU H2020 project SUMMA, under grant agreement 688139.

References

Palakorn Achananuparp, Xiaohua Hu, and Xiajiong Shen. 2008. The evaluation of sentence similarity measures. In *Proceedings of DaWaK*.

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. *Proceedings of EMNLP* .

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. *Linguistic Annotation Workshop* .

Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics* 36(4):673–721.

Guntis Barzdins and Didzis Gosko. 2016a. RIGA at SemEval-2016 task 8: Impact of smatch extensions and character-level neural translation on AMR parsing accuracy. *arXiv preprint arXiv:1604.01278* .

Guntis Barzdins and DIdzis Gosko. 2016b. Riga: Impact of smatch extensions and character-level neural translation on AMR parsing accuracy. *Proceedings of SemEval* .

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of ACL*.

Johan Bos and Katja Markert. 2005. Recognising textual entailment with logical inference. In *Proceedings of HLT/EMNLP*.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. *Proceedings of ACL* .

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, Springer, pages 177–190.

Marco Damonte and Shay B. Cohen. 2018. Cross-lingual abstract meaning representation parsing. *Proceedings of NAACL* .

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for abstract meaning representation. In *Proceedings of EACL*.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of COLING*.

Jeffrey Flanigan, Chris Dyer, Noah A Smith, and Jaime Carbonell. 2016. CMU at SemEval-2016 task 8: Graph-based AMR parsing with infinite ramp loss. *Proceedings of SemEval* .

Jeffrey Flanigan, Sam Thomson, Jaime G Carbonell, Chris Dyer, and Noah A Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. *Proceedings of ACL* .

Jerry A Fodor. 1975. *The language of thought*, volume 5. Harvard University Press.

Alexander Gammerman, Volodya Vovk, and Vladimir Vapnik. 1998. Learning by transduction. In *Proceedings of AUAI*.

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016. Noise reduction and targeted exploration in imitation learning for abstract meaning representation parsing. *Proceedings of ACL* .

Weiwei Guo and Mona Diab. 2012. Modeling sentences in the latent space. In *Proceedings of ACL*.

Sanda Harabagiu and Andrew Hickl. 2006. Methods for using textual entailment in open-domain question answering. In *Proceedings of COLING/ACL*.

Yangfeng Ji and Jacob Eisenstein. 2013. Discriminative improvements to distributional sentence similarity. In *Proceedings of EMNLP*.

Paul Kingsbury and Martha Palmer. 2002. From treebank to propbank. *Proceedings of LREC* .

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural amr: Sequence-to-sequence models for parsing and generation. *arXiv preprint arXiv:1704.08381* .

Thomas K. Landauer, Peter W. Foltz, and Darrell Laham. 1998. An introduction to latent semantic analysis. *Discourse Processes* 25:259–284.

Mike Lewis and Mark Steedman. 2013. Combining distributional and logical semantics. *Transactions of the Association for Computational Linguistics* 1:179–192.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A Smith. 2015. Toward abstractive summarization using semantic representations. *Proceedings of NAACL* .

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of ACL*.

Jonathan May. 2016. Semeval-2016 task 8: Meaning representation parsing. In *Proceedings of SemEval*.

Jonathan May and Jay Priyadarshi. 2017. Semeval-2017 task 9: Abstract meaning representation parsing and generation. In *Proceedings of SemEval*.

Rada Mihalcea, Courtney Corley, Carlo Strapparava, et al. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of AAAI*.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive science* 34(8):1388–1429.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. *Workshop on Incremental Parsing: Bringing Engineering and Cognition Together* .

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

Xiaochang Peng, Linfeng Song, and Daniel Gildea. 2015. A synchronous hyperedge replacement grammar based approach for AMR parsing. *Proceedings of CoNLL* .

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

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

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of EMNLP*.

Peter D Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research* 37:141–188.

Lucy Vanderwende, Arul Menezes, and Chris Quirk. 2015. An AMR parser for English, French, German, Spanish and Japanese and a new AMR-annotated corpus. *Proceedings of NAACL-HLT* .

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015. Boosting transition-based AMR parsing with refined actions and auxiliary analyzers. *Proceedings of ACL* .

Keenon Werling, Gabor Angeli, and Christopher Manning. 2015. Robust subgraph generation improves abstract meaning representation parsing. *arXiv preprint arXiv:1506.03139* .

Junsheng Zhou, Feiyu Xu, Hans Uszkoreit, Weiguang QU, Ran Li, and Yanhui Gu. 2016. AMR parsing with an incremental joint model. In *Proceedings of EMNLP*.

attr2vec: Jointly Learning Word and Contextual Attribute Embeddings with Factorization Machines

Fabio Petroni
Thomson Reuters
Corporate Research & Development
30 South Colonnade, London, UK
fabio.petroni@tr.com

Vassilis Plachouras*
Facebook
1 Rathbone Square, London, UK
vplachouras@fb.com

Timothy Nugent
Thomson Reuters
Corporate Research & Development
30 South Colonnade, London, UK
tim.nugent@tr.com

Jochen L. Leidner
Thomson Reuters
Corporate Research & Development
30 South Colonnade, London, UK
jochen.leidner@tr.com

Abstract

The widespread use of word embeddings is associated with the recent successes of many natural language processing (NLP) systems. The key approach of popular models such as word2vec and GloVe is to learn dense vector representations from the context of words. More recently, other approaches have been proposed that incorporate different types of contextual information, including topics, dependency relations, n-grams, and sentiment. However, these models typically integrate only limited additional contextual information, and often in *ad hoc* ways.

In this work, we introduce attr2vec, a novel framework for jointly learning embeddings for words and contextual attributes based on factorization machines. We perform experiments with different types of contextual information. Our experimental results on a text classification task demonstrate that using attr2vec to jointly learn embeddings for words and Part-of-Speech (POS) tags improves results compared to learning the embeddings independently. Moreover, we use attr2vec to train dependency-based embeddings and we show that they exhibit higher similarity between functionally related words compared to traditional approaches.

1 Introduction

Neural network-based methods have been successful in advancing the state-of-the-art in a wide range of NLP tasks, such as dependency parsing (Chen and Manning, 2014), sentence classification (Kim, 2014), machine translation (Sutskever et al., 2014; Luong and Manning, 2016), and information retrieval (Zhang et al., 2017). In all these approaches, vectorial distributed word representations, known as *word embeddings*, have become a fundamental building block. The use of word embeddings is considered a "secret sauce" for contributing to the success of many of these algorithms in recent years (Luong et al., 2013). Popular models for learning such word embeddings include word2vec (Mikolov et al., 2013a,b,c), GloVe (Pennington et al., 2014) and fastText (Bojanowski et al., 2017; Joulin et al., 2017).

The main idea behind these techniques is to represent a word by means of its context. The most popular forms of context are neighboring words in a window of text (Mikolov et al., 2013b; Pennington et al., 2014), though examples of additional contextual information might also include document topics (Li et al., 2016), dependency relations (Levy and Goldberg, 2014), morphemes (Luong et al., 2013), n-grams (Bojanowski et al., 2017), and sentiment (Tang et al., 2014). The embedding idea was originally devised to help overcome problems associated with the high dimensionality of sparse vector representations of words, particularly in the case of neural network modeling, though embeddings have since been used in a variety of machine learning approaches.

However, existing models generally exploit just a small portion of the available contextual information, and they tend to do so in *ad hoc* ways. The main purpose of context in these models is to shape the word vector space (that is, to associate a representation to the word), but contextual information is not usually represented in this space. For instance, Li and Jurafsky (2015) used document topics to derive multiple vectors for the same word, each capturing a different sense, but

*work conducted whilst the author was at Thomson Reuters.

Proceedings of NAACL-HLT 2018, pages 453–462
New Orleans, Louisiana, June 1 - 6, 2018.

their method does not represent topics in the vector space, that is, it does not generate topic vectors. Such contextual representations, jointly learned with the word representations, could potentially be useful for multiple tasks. For instance, pre-trained contextual vectors could be used as additional features, together with pre-trained word vectors, to improve the performance of existing models.

In this paper, we propose attr2vec, a novel framework for learning word embedding models that jointly associate distributed representations with words and with generic contextual attributes. attr2vec is inspired by the GloVe approach of Pennington et al. (2014) and can mimic it when no additional contextual attribute is considered. In contrast with GloVe, attr2vec uses Factorization Machines (FMs) (Rendle et al., 2011; Rendle, 2012). FMs are a generalization of matrix factorization approaches, such as GloVe, and can combine different generic feature types, even when the input data is sparse. Moreover, FMs do not consider input features as independent but model their interaction by factorizing their latent representation in pairwise fashion.

Here, we conduct an experimental study to assess whether the proposed embedding model can lead to better performance for a text classification task on the *Reuters-21578* dataset, using trained vectors as input to a convolutional neural network. The results show that jointly learned word and Part-of-Speech (POS) embeddings with attr2vec can achieve higher F1 and precision scores compared to embeddings learned independently. Moreover, we use attr2vec to train dependency-based word embeddings and show, using the publicly available *WordSim353* dataset, that such embeddings yield more *functional* similarities than embeddings trained using a linear bag-of-word approach (such as GloVe). We also performed a qualitative analysis that provides insights on how contextual attributes affects the distribution of words in the vector space.

Summing up, the main contributions of our work are the following:

- we extend the GloVe model to consider additional contextual information. To the best of our knowledge, this is the first work to present a general model able to jointly train dense vector representations for word and multiple arbitrary contextual attributes;
- we define a novel loss function based on factorization machines to jointly learn word and contextual attribute embeddings;
- we show how to model the input data and compute co-occurrence statistics using either a linear bag-of-word approach or syntactic dependency relations.

We provide the source code for the attr2vec model at `https://github.com/thomsonreuters/attr2vec`.

The remainder of this paper is organized as follows. Section 2 provides an overview of related work and Section 3 introduces the attr2vec model. In Section 4, we present the experimental results, and close this paper with some concluding remarks in Section 5.

2 Related Work

We have already introduced some of the main related approaches including word2vec and GloVe. Essentially, the GloVe model (Pennington et al., 2014) derives word representations by factorizing the word co-occurrence count matrix. The skip-gram and continuous bag-of-words (CBOW) models of Mikolov et al. (2013a), instead, build the vector space by trying to predict a word given its neighbouring words. Mnih and Kavukcuoglu (2013) proposed a closely related model that works in the opposite way, trying to predict neighbouring words given a word. Facebook's fastText model (Bojanowski et al., 2017; Joulin et al., 2017) augments word embeddings with subword-level information using character n-grams. Other examples of word embedding models include the work of Levy et al. (2014), where an explicit word vector space representation is derived using a PPMI metric, and WordRank of Ji et al. (2016), which learns word representations by adopting a ranking-based loss function. However, none of these models includes any contextual information beyond the neighbouring words.

Several forms of contextual information have been successfully integrated into word embedding models. For instance, Luong et al. (2013); Cotterell and Schütze (2015); Bhatia et al. (2016) capture morphological information into word representations; Bojanowski et al. (2017); Wieting et al. (2016) include character n-grams in their embedding model; Tang et al. (2014) learn sentiment-specific word embeddings by integrating sentiment information in the loss function; Li et al.

(2016) combine word embedding and topic modeling to jointly learn a representation for topics and words. In addition, several works in recent years focused on learning separate embeddings for multiple senses of a word (Neelakantan et al., 2015; Iacobacci et al., 2015; Pilehvar and Collier, 2016). However, all these techniques target a particular type of context. Our attr2vec model differs in that it can jointly represent generic contextual attributes and words in the embedding model. To do so, it makes use of factorization machines (Rendle, 2012), which have been successfully used to exploit contextual information in relation extraction tasks (Petroni et al., 2015) and recommender systems (Rendle et al., 2011)

It is well known that contextual information can improve performance of a wide range of NLP tasks, such as machine translation (Koehn and Hoang, 2007; García-Martínez et al., 2017), named entity typing (Corro et al., 2015) or sentiment analysis (Weichselbraun et al., 2013). In addition, (Melamud et al., 2016) observed that different contextual attributes work well for different tasks and that simple concatenation of embeddings, learned independently with different models, can yield further performance gains. Our attr2vec model can jointly learn embeddings for words and contextual attributes, and we show (see Section 4) that using such jointly learned embeddings yields to better performance on a text classification task compared to embeddings learned independently.

3 The attr2vec model

This section presents the attr2vec model. We first describe how we model the input data in terms of a feature matrix and a target vector (Section 3.1) and then how to factorize those using a factorization machines-based formulation (Section 3.3) to obtain word and contextual attribute embeddings.

3.1 Modeling input data

We consider as input a large corpus of text. Let the vocabulary of considered words be denoted by $W = \{w_1, ..., w_{|W|}\}$ and the set of all contextual variables denoted by $C = \{c_1, ..., c_{|C|}\}$. We denote $V = W \cup C$ the set of all considered words and contextual variables. In the rest of the paper we will refer to the elements of V as *variables*. We model the input data in terms of a target vector $Y \in \mathbb{R}^m$ and feature matrix $X \in \mathbb{R}^{m \times n}$, in which each row $x_i \in \mathbb{R}^n$ corresponds to a feature vector and there are as many columns as the number of variables (i.e., $|V| = n$). We group columns according to the type of the variables; e.g., there are word columns and a group of columns for each type of contextual information considered. Each target value $y_i \in Y$ represents the number of times the feature vector x_i has been observed in the input (i.e., the co-occurrence count). We consider a two-fold way to compute the co-occurrence count of a feature vector x_i: (1) *linear bag-of-words* and (2) *dependency-based*.

Linear Bag-of-Words The approach used in Pennington et al. (2014) is to compute the co-occurrence count using a linear bag-of-words assumption. The idea is to use a window of size k around the target word w, and considering the k words before and the k words after w to compute co-occurrence statistics[1]. Note that a small window size may miss some information, while a large window size might result in the algorithm capturing accidental pairs. A decay factor is commonly used to weight the contribution of an observation to the total co-occurrence count according to the distance to the target word. Here we consider a fractional decay, where the importance of a word is assumed to be inversely proportional to its distance from the target word.

To build the feature matrix X we set the values of the variables associated with the words pair and with each contextual variable observed in correspondence with that pair to 1. Note that there could be multiple rows in X referring to the same pair of words but associated with different contextual variables. When contextual variables can assume multiple values for a single observation (e.g., a document with multiple topics) we evenly distribute the unitary weight across all variables (e.g., across all document topics). The target values represent the number of times the corresponding combination of features (i.e., the pair of word and the contextual variables) has been observed in the input corpus, weighting each contribution with the fractional decay factor described above.

An example is shown in Figure 1. The first row in the figure corresponds to the observation of the pair of words *brothers* and *lehman* in a text window, with POS tags *nnp* and *nnps*, respectively, referring to the named entity *Lehman*

[1] In this paper we focus on symmetric windows, however the same model can be extended to asymmetric windows.

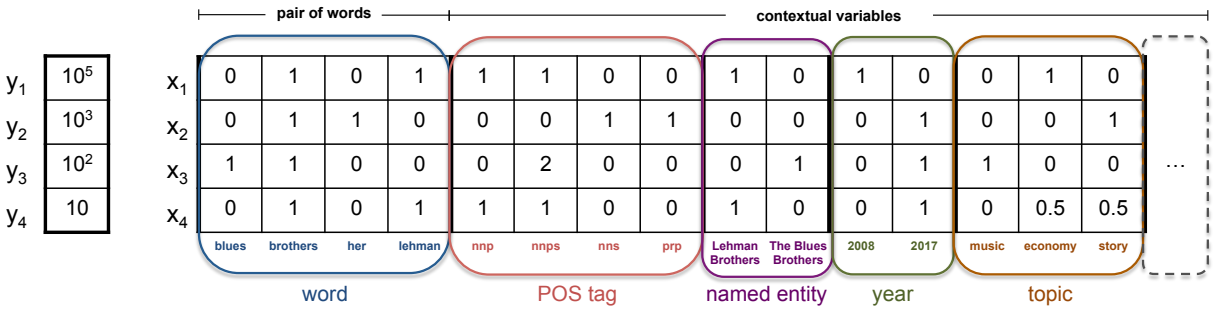


Figure 1: Example for representing input data with attr2vec using a linear window of text approach to compute co-occurrence count. Rows in X and Y are aligned: the row $y_i \in Y$ (a scalar) correspond to the row $x_i \in X$ (a feature vector); y_i represents the frequency of the particular combination of variables described by x_i in the corpus.

Brothers, from documents published in *2008* with topic *economy*. Such a combination of features (i.e., *brothers-lehman-nnp-nnps-Lehman Brothers-2008-economy* has been observed in the input 10^5 times (i.e., $y = 10^5$). The fourth row of Figure 1 conveys the information that the same pair of words (i.e., *brothers-lehman*) has been also observed 10 times in the input associated with different contextual information, in particular in documents published in *2017* with multi-topic *economy* and *story*. When contextual information is not considered (i.e., $c = \emptyset$) y is equal to the co-occurrence count of the pair of words, as in Pennington et al. (2014).

Dependency-Based Our attr2vec model can learn dependency-based embedding as well. In order to do so we adopted a similar strategy of Levy and Goldberg (2014). The main idea is to parse each sentence in the input corpus and to use the dependency tree to derive the co-occurrence count. In particular, for a target word w with modifiers $m_1, ..., m_o$ and a head h, we considered the dependency labels $(m_1, lbl_1), ..., (m_o, lbl_o), (h, lbl_h^{-1})$, where lbl is the type of the dependency relation between the head and the modifier and lbl^{-1} is used to mark the inverse relation. Moreover, edges that include a preposition are "collapsed" by connecting the head and the object of the proposition, and subsuming the preposition itself into the dependency label (see the example in the top part of Figure 2).

The feature matrix X in this case consists of two group of columns, one for the words and one for dependency labels, plus one group of columns for each additional contextual information considered. The co-occurrence count is driven by the dependency tree: each target value represents the number of times the corresponding word and dependency label (and all considered additional contextual information) appear in the dependency trees representing the sentences in the input corpus.

An example is shown in Figure 2. The first row in the matrix corresponds to the observation of the word *ganymede* connected to the word *discovered* through the inverse relation $dobj$ in the dependency tree (i.e., $discovered/dobj^{-1}$). Notice that using this approach it is possible to capture relevant relations between words "far apart" in the text including long-distance dependencies (e.g., *telescope* is not in the window of text around *discovered* for $k = 3$), and also filter out accidental neighbouring words (e.g., *his* is in the window of text of *discovered* for $k = 3$). An additional advantage of this approach with respect to a linear bag-of-words solution is that each observation is typed, indicating, for instance, that *Ganymede* is the object of *discovered* and *Galileo* is the subject.

3.2 GloVe factorization model

Before introducing the attr2vec model we briefly describe the GloVe factorization approach (Pennington et al., 2014). In particular, GloVe employs a matrix factorization model using as input the word-word co-occurrence count. In our example of Figure 1 this corresponds to considering in input just the first group of columns (i.e., the pair of words).

Starting from the observation that the ratio of co-occurrence probabilities is more appropriate for learning word representations as opposed to the probabilities of the words themselves, Pennington et al. propose the following weighted least

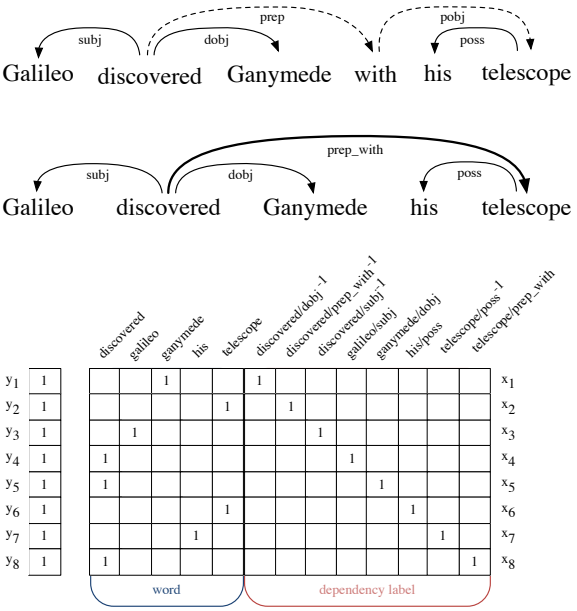


Figure 2: Example for representing input data with attr2vec using a dependency-based approach to compute co-occurrence count. **Top:** preposition relations are collapsed into single arcs, making telescope a direct modifier of discovered. **Bottom:** the attr2vec data matrix. Note that the Y vector contains all 1's since it refer to the modeling of a single sentence; in a real scenario, each row y_i of vector Y will contain a higher value (the frequency count of x_i in the corpus)

squares objective function:

$$J = \sum_{k=1}^{N} f(y_k)\Big(s(x_k) - \log(y_k)\Big)^2 \qquad (1)$$

where $x_k = (w_i, w_j)$ refers to the k-th pair of words in input with co-occurrence count y_k, and $s(x)$ is the score associated with the pair, computed as follows:

$$s(x) = b_{w_i} + b_{w_j} + \boldsymbol{f}_{w_i}^T \boldsymbol{f}_{w_j} \qquad (2)$$

where b_{w_i} and b_{w_j} are the biases, and $\boldsymbol{f}_{w_i}$ and $\boldsymbol{f}_{w_j}$ are the latent factor vectors associated with w_i and w_j, respectively.

The function $f(y)$ is used to reduce the importance of pairs of words that co-occur rarely, and is defined as follows:

$$f(y) = \begin{cases} (y/y_{max})^{\alpha} & \text{if } y < y_{max} \\ 1 & \text{otherwise} \end{cases} \qquad (3)$$

where y_{max} and α are hyperparameters of the model.

3.3 attr2vec factorization model

The model we propose, attr2vec, employs a matrix factorization model based on factorization machines. In particular, we associate with each variable $v \in V$ a *bias term* $b_v \in \mathbb{R}$ and a *latent factor vector* $\boldsymbol{f}_v \in \mathbb{R}^d$, where the *dimensionality* d of the latent factor space is a hyperparameter of the model. For each input feature vector $x \in X$, we denote by x_v the value of variable $v \in V$ in the corresponding row of the features data matrix.

We employ a weighted least squares model that is based of the formulation of Pennington et al. (2014) (Equation 1). In contrast to GloVe, we define a novel score $s(x)$ that takes into account both words and contextual attributes, computed as follows:

$$s(x) = \sum_{v \in V} x_v b_v + \sum_{v_1 \in V} \sum_{v_2 \in V \setminus \{v_1\}} x_{v_1} x_{v_2} \boldsymbol{f}_{v_1}^T \boldsymbol{f}_{v_2} \qquad (4)$$

Here, the bias terms b_v model the contribution of each individual variable to the final score, whereas the latent factor vectors $\boldsymbol{f}_v$ model the contribution of all pairwise interactions between variables. Rendle (2012) has shown that score computation is fast, since $s(x)$ can be computed in time linear to both the number of nonzero entries in x and the dimensionality d. Each latent factor vector can be interpreted as a low-dimensional representation of the corresponding variable, both for variables that refer to words and for variables that refer to contextual information. Note that, when contextual information is not considered the formulation of our factorization model is equivalent to the formulation of Pennington et al. (2014).

The model parameters $\Theta = \{b_v, \boldsymbol{f}_v | v \in V\}$ are estimated by minimizing J, for instance, through stochastic gradient descent. Each $\boldsymbol{f}_v$ can be interpret as a dense vector representation of variable $v \in V$.

4 Experiments

We conducted an experimental study on real-world data to compare our attr2vec model with other state-of-the-art approaches.

4.1 Dataset and Baseline

For a training corpus to learn embeddings, we used the Reuters News Archive, in particular, the collection of all news stories published by the Reuters News Agency from 2003 to 2017. We

embedding	*input*	*logistic regression*	*convolutional neural network*	
			static	*non-static*
random	$\vec{w_r}$	70.5 (69.3)	75.7 (77.5)	74.4 (76.2)
random	$\vec{w_r} \frown \vec{p_r}$	74.0 (73.9)	77.9 (79.7)	77.8 (79.8)
GloVe	$\vec{w_i}$	77.5 (77.5)	79.7 (81.5)	82.7 (84.3)
GloVe	$\vec{w_i} \frown \vec{p_r}$	80.2 (85.4)	82.5 (84.1)	84.5 (86.1)
GloVe	$\vec{w_i} \frown \vec{p_i}$	79.3 (83.3)	84.3 (85.8)	84.9 (86.4)
attr2vec	$\vec{w_j}$	77.5 (77.3)	80.6 (82.3)	82.8 (84.5)
attr2vec	$\vec{w_j} \frown \vec{p_j}$	80.1 (83.1)	84.9 (86.1)	**85.5** (**86.8**)

Table 1: Average F1 score (and precision in parentheses) for topic prediction on the *Reuters-21578* dataset. $\vec{w} \frown \vec{p}$ indicates the concatenation of word and POS tag vectors. $\vec{w_r}$ refer to randomly initialized word vectors and $\vec{p_r}$ to randomly initialized POS tag vector. $\vec{w_i}$ and $\vec{p_i}$ respectively refer to vectors independently trained with the GloVe model for words and POS tags; $\vec{w_j}$ and $\vec{p_j}$ respectively refer to vectors jointly trained with attr2vec for words and POS tags.

first applied a heuristic filtering approach to exclude non-textual documents, resulting in a collection of $\sim 8M$ news articles ($\sim 3B$ tokens). We then performed tokenization, part-of-speech tagging, and syntactic dependency parsing on the corpus using NLP4J[2] (Choi et al., 2015; Choi, 2016). The POS tagger achieves an accuracy score of 97.64% (Choi, 2016), the dependency parser achieve a label accuracy score of 94.94% (Choi et al., 2015). As a baseline we considered 200-dimensional GloVe vectors trained on the corpus using the code and hyperparameters of Pennington et al. (2014). In particular, we used $y_{max} = 100$ and $\alpha = 3/4$ for all our experiments.

4.2 Topic Classification Experiment

Experimental Setup For this experiment we trained 200-dimensional attr2vec vectors using part-of-speech tags as additional contextual information and a linear bag-of-words approach - i.e., each row in the feature matrix consists of a pair of words and the corresponding pair of POS tags (see the first two groups of columns for the example in Figure 1). We used the same hyperparameters as in GloVe. To make a fair comparison we trained two independent GloVe models, one to obtain word vectors ($\vec{w_i}$) and one to obtain POS tag vectors ($\vec{p_i}$). The latter model is trained by substituting each word in the corpus with the corresponding POS tag. Note that our attr2vec model can jointly learn a representation for both words ($\vec{w_j}$) and POS tags ($\vec{p_j}$). As a baseline we also considered randomly initialized vectors for words ($\vec{w_r}$) and POS tags ($\vec{p_r}$). To train attr2vec we used a modified version[3] of tffm (Trofimov and Novikov, 2016), an open-source TensorFlow implementation of Factorization Machines.

To evaluate the performance of the attr2vec model, we used the trained vectors as input for a convolutional neural network (CNN). We used the CNN architecture described by Kim (2014), in particular a modified version of the TensorFlow implementation in Britz (2015), where we add support for pre-trained embeddings. As hyperparameters, we used a batch size of 128 training samples, no dropout, one layer, filter windows of 3, 4, 5 with 100 feature maps each. We trained using the Adam optimizer and a learning rate of 0.001 and let the models train for 100 epochs (an epoch is an iteration over all the training points). We executed three independent runs for each experiment and we report averaged results.

As a benchmark we used the following text classification task: predict all the topic codes associated with an article using the first τ tokens in the article. We used the *Reuters-21578*[4] dataset. This corpus contains 10788 news documents classified into 90 topics. We used the provided training/test split. For each document, we considered the first $\tau = 250$ tokens as input text for the CNN. Note that, in contrast to other previous work (Li et al., 2016), we consider all topics and formulate a multi-label classification problem. For each test article we computed precision, recall and F1 score comparing the actual topic codes and those predicted by the CNN. As evaluation metrics we used the average F1 score and the average preci-

[2] https://emorynlp.github.io/nlp4j

[3] Source code available at https://github.com/thomsonreuters/attr2vec

[4] http://www.nltk.org/book/ch02.html

sion across all test articles.

We trained multiple CNN models, using either word vectors ($\vec{w}$) or the concatenation of word and POS tag vectors ($\vec{w}\frown\vec{p}$) as inputs, and keeping these vectors *static* throughout training or allowing the CNN to update them via backpropagation (*non-static*). We also considered logistic regression as baseline method, using averaged vectors calculated over the input text as features, as in Zhang and Wallace (2015). As this is a multilabel task, we used the one-vs-all formulation of logistic regression, which attempts to fit one classifier per class with each class being fitted against all other classes. L_1 regularization was applied with a weight of 0.005.

Results Table 1 reports the result of our experiments. Each entry shows the average F1 score and the average precision in parentheses.

First note that the CNN model consistently outperforms logistic regression for all considered settings. The CNN performance improves if it receives as input pre-trained vectors as opposed to random ones, consistently with other works (Kim, 2014). The performance is comparable when GloVe or attr2vec word vectors are used as input.

The key advantage of our attr2vec model over GloVe is demonstrated when additional contextual information is considered in the CNN model. The performance of the CNN model improves if POS vectors are considered together with GloVe word vectors in input, both when such POS vectors are randomly initialized ($\vec{w_i}\frown\vec{p_r}$) and independently trained with the GloVe model ($\vec{w_i}\frown\vec{p_i}$). However, the best performance is achieved when word and POS tags vectors are jointly trained with our attr2vec model ($\vec{w_j}\frown\vec{p_j}$).

Note that the aim of the paper was not to show that POS tags help for text classification tasks (to that end, an exhaustive exploration of the parameter space would have been needed); instead, the goal of this work is to introduce a new embedding model that jointly learns a representation for words and POS tags, capturing the interaction between them, and to show that such representation is beneficial for a CNN with respect to embeddings learned in an independent fashion, given the same network settings. Standard embedding models (like GloVe), in fact, can capture either the interactions between words or between POS tags in an independent fashion. Our attr2vec model, in addition, captures the cross-interaction between

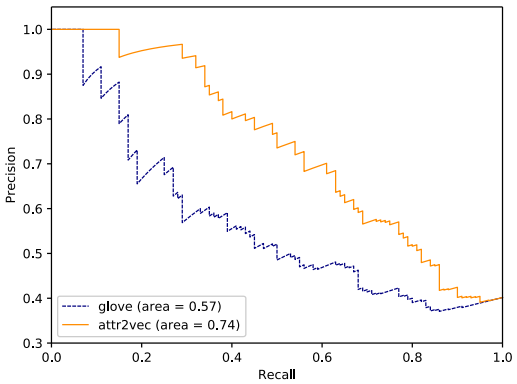


Figure 3: Recall-precision curve when attempting to rank the similar words above the related ones on the *WordSim353* dataset.

words and contextual attributes, jointly learning their representation, and our results suggest that this additional information is beneficial for the performance of the CNN model. Moreover, note that our attr2vec algorithm, unlike GloVe, can handle generic contextual information.

4.3 Word Similarity Experiment

In our second experiment we wanted to address if our attr2vec model was able to produce dependency-based embeddings that exhibit more functional similarity than GloVe embeddings (that usually yield broad topical similarities). To this end, we trained 200-dimensional attr2vec vectors using a dependency-based approach - i.e, each row in the feature matrix consist of a word and a dependency label (see the example in Figure 2).

Our evaluation closely follows the one in Levy and Goldberg (2014). In particular, we used the *WordSim353* dataset (Finkelstein et al., 2001; Agirre et al., 2009) containing pairs of similar words that reflect either *relatedness* (topical similarity) or *similarity* (functional similarity) relations. The pairs are ranked according to the cosine similarity between the corresponding word vectors. The idea is that a model that focuses on functional similarity should rank similar pairs in the dataset above the related ones. For instance, such a model should rank the pair *money-currency* (i.e., functionally similar words) above the pair *money-laundering* (i.e., topically similar words). We drew a recall-precision curve by considering related pair as a miss and similar pair as a hit. In this way we aimed to capture the embeddings affinity towards the *similarity* subset over the *re-*

latedness one.

Figure 3 reports the result of the experiment. The attr2vec curve (orange solid line) is higher than the GloVe one (blue dashed line) and the area under the cuve is larger (0.74 with respect to 0.57), suggesting that attr2vec yields more functional similarities with respect to GloVe. Note that a similar behaviour has been observed in Levy and Goldberg (2014) for context-predictive models (i.e., the skip-gram model with negative sampling). To the best of our knowledge, attr2vec is the first model that incorporates syntactic dependency relations in a co-occurrence counts based model (such as GloVe). Moreover, attr2vec is a general model that can handle additional arbitrary contextual information.

4.4 Qualitative Evaluation

Our final evaluation is qualitative. We trained 200-dimensional attr2vec embeddings using news topics as additional contextual information and a linear bag-of-words approach - i.e., each row in the feature matrix consists of a pair of words and the topic of the news article where such pair has been observed (see the first and the last group of columns for the example in Figure 1). In particular, we used the same collection of $\sim 8M$ news articles presented in Section 4.1 and we considered the following two article topics: general news stories (G) and sport news (SPO).

Figure 4 shows a two-dimensional projection of the 200-dimensional vector space where words and topics representations lie, obtained using the *t-SNE*[5] visualisation technique (Maaten and Hinton, 2008). Here the two topic points (G on the left and SPO on the right of the figure) seem to metaphorically act as "magnets", modifying the space and forming two clusters of words. The left cluster around the representation of topic G includes general words not related to sports such as "mars", "sound", "warranty", "finance", "train", while the right cluster around the representation of topic SPO contains words related to sports such as "football", "coach", "game", "stadium", "cricket". Words that are related with both general news stories and sport news lie somewhere in the middle between these two clusters. Examples of such words include "penalties", "transfer", "medical", "goal", "supporters". Note that there are other attractive and repulsive forces in the vector space driven by word similarity, and that a two-dimensional representation is only able to capture a small portion of all relations that take place in the higher dimensional space.

[5]We used the TensorBoard implementation of *t-SNE*.

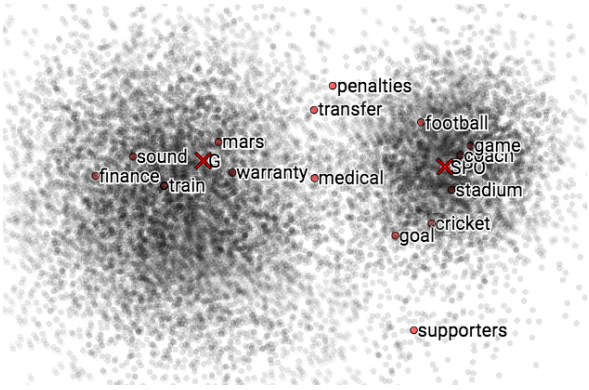


Figure 4: Two-dimensional projection of the 200-dimensional vector space, that contains representations of both words and topics, using *t-SNE*. In particular, we considered two topics: general news stories (G) and sport news (SPO).

5 Conclusions

In this paper, we proposed attr2vec, a novel embedding model that can jointly learn a distributed representation for words and contextual attributes. Our model is general and can handle multiple arbitrary contextual information simultaneously. To do so, we defined a novel loss function based on factorization machines. Moreover, attr2vec can mimic existing word embedding algorithms when no additional contextual information is considered. In particular, GloVe is a special case of our model.

We have presented an experimental study where we considered POS tags as additional contextual information, and fed a convolutional neural network (CNN) with both word and POS tag vectors. The results suggest that the CNN prediction performance improves when word and context vectors are jointly learned by our attr2vec model. In addition, we described how to train dependency-based attr2vec embeddings and showed that they produce different kinds of similarities. We also provided some insights into how the vector space is affected by contextual attributes, which seem to act like "magnets" that attract or repulse words, that are themselves subject to attractive or repulsive forces driven by similarity.

While attr2vec benefits from structural information, it has a price: the number of features is in-

creased, and the computational cost is increased compared to a model that does not use contextual information. Each additional attribute may furthermore introduce its own noise (component-specific errors) into the process. Nevertheless, the overall improvement can help in tasks where quality is of the utmost importance and high-quality annotation components are available.

In future work, we aim to investigate the effect of adding different contextual information, and we plan to test the resulting models in various applications.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and WordNet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Parminder Bhatia, Robert Guthrie, and Jacob Eisenstein. 2016. Morphological Priors for Probabilistic Neural Word Embeddings. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* 5:135–146.

Denny Britz. 2015. Implementing a CNN for Text Classification in Tensorflow. https://github.com/dennybritz/cnn-text-classification-tf.

Danqi Chen and Christopher D. Manning. 2014. A Fast and Accurate Dependency Parser using Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Jinho D Choi. 2016. Dynamic Feature Induction: The Last Gist to the State-of-the-Art. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Jinho D Choi, Joel Tetreault, and Amanda Stent. 2015. It depends: Dependency parser comparison using a web-based evaluation tool. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*.

Luciano del Corro, Abdalghani Abujabal, Rainer Gemulla, and Gerhard Weikum. 2015. Finet: Context-aware fine-grained named entity typing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Ryan Cotterell and Hinrich Schütze. 2015. Morphological Word-Embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*.

Mercedes García-Martínez, Loïc Barrault, and Fethi Bougares. 2017. Neural Machine Translation by Generating Multiple Linguistic Factors. In *International Conference on Statistical Language and Speech Processing*.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. SensEmbed: Learning Sense Embeddings for Word and Relational Similarity. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*.

Shihao Ji, Hyokun Yun, Pinar Yanardag, Shin Matsushima, and SVN Vishwanathan. 2016. Wordrank: Learning word embeddings via robust ranking. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* .

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Philipp Koehn and Hieu Hoang. 2007. Factored Translation Models. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning*.

Omer Levy and Yoav Goldberg. 2014. Dependency-Based Word Embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Omer Levy, Yoav Goldberg, and Israel Ramat-Gan. 2014. Linguistic Regularities in Sparse and Explicit Word Representations. In *Proceedings of the eighteenth conference on computational natural language learning*.

Jiwei Li and Dan Jurafsky. 2015. Do Multi-Sense Embeddings Improve Natural Language Understanding? *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* .

Shaohua Li, Tat-Seng Chua, Jun Zhu, and Chunyan Miao. 2016. Generative Topic Embedding: a Continuous Representation of Documents. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving Open Vocabulary Neural Machine Translation with Hybrid Word-Character Models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better Word Representations with Recursive Neural Networks for Morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9(Nov):2579–2605.

Oren Melamud, David McClosky, Siddharth Patwardhan, and Mohit Bansal. 2016. The Role of Context Types and Dimensionality in Learning Word Embeddings. In *Proceedings of The 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. *ICLR Workshop* .

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems*. pages 2265–2273.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2015. Efficient Non-parametric Estimation of Multiple Embeddings per Word in Vector Space. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* .

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Fabio Petroni, Luciano Del Corro, and Rainer Gemulla. 2015. CORE: Context-Aware Open Relation Extraction with Factorization Machines. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Mohammad Taher Pilehvar and Nigel Collier. 2016. De-Conflated Semantic Representations. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* .

Steffen Rendle. 2012. Factorization Machines with libFM. *ACM Transactions on Intelligent Systems and Technology (TIST)* 3(3):57:1–57:22.

Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. Fast Context-aware Recommendations with Factorization Machines. In *Proceedings of the 34th international ACM SIGIR Conference on Research and development in Information Retrieval*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Mikhail Trofimov and Alexander Novikov. 2016. tffm: TensorFlow implementation of an arbitrary order Factorization Machine. https://github.com/geffy/tffm.

Albert Weichselbraun, Stefan Gindl, and Arno Scharl. 2013. Extracting and Grounding Context-Aware Sentiment Lexicons. *IEEE Intelligent Systems* 28(2):39–46.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding Words and Sentences via Character n-grams. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Ye Zhang, Md Mustafizur Rahman, Alex Braylan, Brandon Dang, Heng-Lu Chang, Henna Kim, Quinten McNamara, Aaron Angert, Edward Banner, Vivek Khetan, Tyler McDonnell, An Thanh Nguyen, Dan Xu, Byron C. Wallace, and Matthew Lease. 2017. Neural Information Retrieval: A Literature Review. *arXiv preprint arXiv:1611.06792v3* .

Ye Zhang and Byron Wallace. 2015. A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. *arXiv preprint arXiv:1510.03820* .

Can Network Embedding of Distributional Thesaurus be Combined with Word Vectors for Better Representation?

Abhik Jana
IIT Kharagpur
Kharagpur, India
abhik.jana@iitkgp.ac.in

Pawan Goyal
IIT Kharagpur
Kharagpur, India
pawang@cse.iitkgp.ac.in

Abstract

Distributed representations of words learned from text have proved to be successful in various natural language processing tasks in recent times. While some methods represent words as vectors computed from text using predictive model (Word2vec) or dense count based model (GloVe), others attempt to represent these in a distributional thesaurus network structure where the neighborhood of a word is a set of words having adequate context overlap. Being motivated by recent surge of research in network embedding techniques (DeepWalk, LINE, node2vec etc.), we turn a distributional thesaurus network into dense word vectors and investigate the usefulness of distributional thesaurus embedding in improving overall word representation. This is the first attempt where we show that combining the proposed word representation obtained by distributional thesaurus embedding with the state-of-the-art word representations helps in improving the performance by a significant margin when evaluated against NLP tasks like word similarity and relatedness, synonym detection, analogy detection. Additionally, we show that even without using any handcrafted lexical resources we can come up with representations having comparable performance in the word similarity and relatedness tasks compared to the representations where a lexical resource has been used.

1 Introduction

Natural language understanding has always been a primary challenge in natural language processing (NLP) domain. Learning word representations is one of the basic and primary steps in understanding text and nowadays there are predominantly two views of learning word representations. In one realm of representation, words are vectors of distributions obtained from analyzing their contexts in the text and two words are considered meaningfully similar if the vectors of those words are close in the euclidean space. In recent times, attempts have been made for dense representation of words, be it using predictive model like Word2vec (Mikolov et al., 2013) or count-based model like GloVe (Pennington et al., 2014) which are computationally efficient as well. Another stream of representation talks about network like structure where two words are considered neighbors if they both occur in the same context above a certain number of times. The words are finally represented using these neighbors. Distributional Thesaurus is one such instance of this type, which gets automatically produced from a text corpus and identifies words that occur in similar contexts; the notion of which was used in early work about distributional semantics (Grefenstette, 2012; Lin, 1998; Curran and Moens, 2002). One such representation is JoBimText proposed by Biemann and Riedl (2013) that contains, for each word, a list of words that are similar with respect to their bigram distribution, thus producing a network representation. Later, Riedl and Biemann (2013) introduced a highly scalable approach for computing this network. We mention this representation as a DT network throughout this article. With the emergence of recent trend of embedding large networks into dense low-dimensional vector space efficiently (Perozzi et al., 2014; Tang et al., 2015; Grover and Leskovec, 2016) which are focused on capturing different properties of the network like neighborhood structure, community structure, etc., we explore representing DT network in a dense vector space and evaluate its useful application in various NLP tasks.

There has been attempt (Ferret, 2017) to turn distributional thesauri into word vectors for synonym extraction and expansion but the full utilization of DT embedding has not yet been explored. In this paper, as a main contribution, we

Proceedings of NAACL-HLT 2018, pages 463–473
New Orleans, Louisiana, June 1 - 6, 2018.

investigate the best way of turning a Distributional Thesaurus (DT) network into word embeddings by applying efficient network embedding methods and analyze how these embeddings generated from DT network can improve the representations generated from prediction-based model like Word2vec or dense count based semantic model like GloVe. We experiment with several combination techniques and find that DT network embedding can be combined with Word2vec and GloVe to outperform the performances when used independently. Further, we show that we can use DT network embedding as a proxy of WordNet embedding in order to improve the already existing state-of-the-art word representations as both of them achieve comparable performance as far as word similarity and word relatedness tasks are concerned. Considering the fact that the vocabulary size of WordNet is small and preparing WordNet like lexical resources needs huge human engagement, it would be useful to have a representation which can be generated automatically from corpus. We also attempt to combine both WordNet and DT embeddings to improve the existing word representations and find that DT embedding still has some extra information to bring in leading to better performance when compared to combination of only WordNet embedding and state-of-the-art word embeddings. While most of our experiments are focused on word similarity and relatedness tasks, we show the usefulness of DT embeddings on synonym detection and analogy detection as well. In both the tasks, combined representation of GloVe and DT embeddings shows promising performance gain over state-of-the-art embeddings.

2 Related Work

The core idea behind the construction of distributional thesauri is the distributional hypothesis (Firth, 1957): "You should know a word by the company it keeps". The semantic neighbors of a target word are words whose contexts overlap with the context of a target word above a certain threshold. Some of the initial attempts for preparing distributional thesaurus are made by Lin (1998), Curran and Moens (2002), Grefenstette (2012). The semantic relation between a target word and its neighbors can be of different types, e.g., synonymy, hypernymy, hyponymy or other relations (Adam et al., 2013; Budanitsky and Hirst, 2006) which prove to be very useful in different natural language tasks. Even though computation of sparse count based models used to be inefficient, in this era of high speed processors and storage, attempts are being made to streamline the computation with ease. One such effort is made by Kilgarriff et al. (2004) where they propose Sketch Engine, a corpus tool which takes as input a corpus of any language and corresponding grammar patterns, and generates word sketches for the words of that language and a thesaurus. Recently, Riedl and Biemann (2013) introduce a new highly scalable approach for computing quality distributional thesauri by incorporating pruning techniques and using a distributed computation framework. They prepare distributional thesaurus from Google book corpus in a network structure and make it publicly available.

In another stream of literature, word embeddings represent words as dense unit vectors of real numbers, where vectors that are close together in euclidean space are considered to be semantically related. In this genre of representation, one of the captivating attempt is made by Mikolov et al. (2013), where they propose Word2vec, basically a set of two predictive models for neural embedding whereas Pennington et al. (2014) propose GloVe, which utilizes a dense count based model to come up with word embeddings that approximate this. Comparisons have also been made between count-based and prediction-based distributional models (Baroni et al., 2014) upon various tasks like relatedness, analogy, concept categorization etc., where researchers show that prediction-based word embeddings outperform sparse count-based methods used for computing distributional semantic models. In other study, Levy and Goldberg (2014) show that dense count-based methods, using PPMI weighted co-occurrences and SVD, approximates neural word embeddings. Later, Levy et al. (2015) show the impact of various parameters and the best performing parameters for these methods. All these approaches are completely text based; no external knowledge source has been used.

More recently, a new direction of investigation has been opened up where researchers are trying to combine knowledge extracted from knowledge bases, images with distributed word representations prepared from text with the expectation of getting better representation. Some use

Knowledge bases like WordNet (Miller, 1995), FreeBase (Bollacker et al., 2008), PPDB (Ganitkevitch et al., 2013), ConceptNet (Speer et al., 2017), whereas others use ImageNet (Frome et al., 2013; Kiela and Bottou, 2014; Both et al., 2017; Thoma et al., 2017) for capturing visual representation of lexical items. There are various ways of combining multiple representations. Some of the works extract lists of relations from knowledge bases and use those to either modify the learning algorithms (Halawi et al., 2012; Wang et al., 2014; Tian et al., 2016; Rastogi et al., 2015) or post-process pre-trained word representations (Faruqui et al., 2015). Another line of literature prepares dense vector representation from each of the modes (text, knowledge bases, visual etc.) and tries to combine the vectors using various methods like concatenation, centroid computation, principal component analysis (Jolliffe, 1986), canonical correlation analysis (Faruqui and Dyer, 2014) etc. One such recent attempt is made by Goikoetxea et al. (2016) where they prepare vector representation from WordNet following the method proposed by Goikoetxea et al. (2015), which combines random walks over knowledge bases and neural network language model, and tries to improve the vector representation constructed from text using this. As in lexical knowledge bases, the number of lexical items involved is much less than the raw text and preparing such resources is a cumbersome task, our goal is to see whether we can use DT network instead of some knowledge bases like WordNet and achieve comparable performance on NLP tasks like word similarity and word relatedness. In order to prepare vector representation from DT network, we attempt to use various network embeddings like DeepWalk (Perozzi et al., 2014), LINE (Tang et al., 2015), struc2vec (Ribeiro et al., 2017), node2vec (Grover and Leskovec, 2016) etc. Some of those try to capture the neighbourhood or community structure in the network while others attempt to capture structural similarity between nodes, second order proximity, etc.

3 Proposed Methodology

Our aim is to analyze the effect of integrating the knowledge of Distributional Thesaurus network with the state-of-the-art word representation models to prepare a better word representation. We first prepare vector representations from Distributional Thesaurus (DT) network applying network representation learning model. Next we combine this thesaurus embedding with state-of-the-art vector representations prepared using GloVe and Word2vec model for analysis.

3.1 Distributional Thesaurus (DT) Network

Riedl and Biemann (2013) use the Google books corpus, consisting of texts from over 3.4 million digitized English books published between 1520 and 2008 and construct a distributional thesauri (DT) network using the syntactic n-gram data (Goldberg and Orwant, 2013). The authors first compute the lexicographer's mutual information (LMI) (Kilgarriff et al., 2004) for each bigram, which gives a measure of the collocational strength of a bigram. Each bigram is broken into a word and a feature, where the feature consists of the bigram relation and the related word. Then the top 1000 ranked features for each word are taken and for each word pair, intersection of their corresponding feature set is obtained. The word pairs having number of overlapping features above a threshold are retained in the network. In a nutshell, the DT network contains, for each word, a list of words that are similar with respect to their bigram distribution (Riedl and Biemann, 2013). In the network, each word is a node and there is a weighted edge between a pair of words where the weight corresponds to the number of overlapping features. A sample snapshot of the DT is shown in Figure 1.

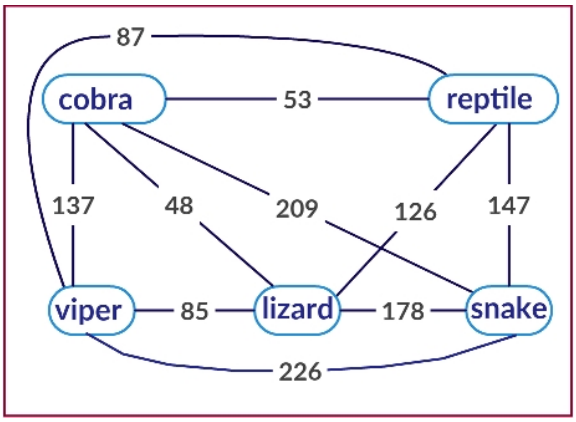


Figure 1: A sample snapshot of Distributional Thesaurus network where each node represents a word and the weight of an edge between two words is defined as the number of context features that these two words share in common.

3.2 Embedding Distributional Thesaurus

Now, from the DT network, we prepare the vector representation for each node using network representation learning models which produce vector representation for each of the node in a network. For this purpose, we use three state-of-the-art network representation learning models as discussed below.

DeepWalk: DeepWalk (Perozzi et al., 2014) learns social representations of a graph's vertices by modeling a stream of short random walks. Social representations signify latent features of the vertices that capture neighborhood similarity and community membership.

LINE: LINE (Tang et al., 2015) is a network embedding model suitable for arbitrary types of networks: undirected, directed and/or weighted. The model optimizes an objective which preserves both the local and global network structures by capturing both first-order and second-order proximity between vertices.

node2vec: node2vec (Grover and Leskovec, 2016) is a semi-supervised algorithm for scalable feature learning in networks which maximizes the likelihood of preserving network neighborhoods of nodes in a d-dimensional feature space. This algorithm can learn representations that organize nodes based on their network roles and/or communities they belong to by developing a family of biased random walks, which efficiently explore diverse neighborhoods of a given node.

Note that, by applying network embedding models on DT network we obtain 128 dimensional vectors for each word in the network. We only consider edges of the DT network having edge weight greater or equal to 50 for network embedding. Henceforth, we will use *D2V-D*, *D2V-L* and *D2V-N* to indicate vector representations obtained from DT network produced by DeepWalk, LINE and node2vec, respectively.

After obtaining vector representations, we also explore whether these can be combined with the pre-trained vector representation of Word2vec and GloVe to come up with a joint vector representation. For that purpose, we directly use very well-known GloVe 1.2 embeddings (Pennington et al., 2014) trained on 840 billion words of the common crawl dataset having vector dimension of 300. As an instance of pre-trained vector of Word2vec, we use prominent pre-trained vector representations prepared by Mikolov et al. (2013) trained on 100 billion words of Google News using skip-grams with negative sampling, having dimension of 300.

3.3 Vector Combination Methods

In order to integrate the word vectors, we apply two strategies inspired by Goikoetxea et al. (2016): concatenation (CC) and principal component analysis (PCA).

Concatenation (CC): This corresponds to the simple vector concatenation operation. Vector representations of both GloVe and Word2vec are of 300 dimensions and word embeddings learnt form DT are of 128 dimensions. The concatenated representation we use are of 428 dimensions.

Principal Component Analysis (PCA): Principal component analysis (Jolliffe, 1986) is a dimensionality reduction statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components (linear combinations of the original variables). We apply PCA to the concatenated representations (dimension of 428) reducing these to 300 dimensions. In addition to PCA, we try with truncated singular value decomposition procedure (Hansen, 1987) as well, but as per the experiment set up, it shows negligible improvement in performance compared to simple concatenation; hence we do not continue with the truncated singular value decomposition for dimensionality reduction. After obtaining the combined representations of words, we head towards evaluating the quality of the representation.

4 Experiments and Analysis

In order to evaluate the quality of the word representations, we first conduct qualitative analysis of the joint representation. Next, we follow the most acceptable way of applying on different NLP tasks like word similarity and word relatedness, synonym detection and word analogy as described next.

4.1 Qualitative Analysis:

On qualitative analysis of some of the word pairs from the evaluation dataset, we observe that the joint representation (PCA (GloVe,D2V-N)) captures the notion of similarity much better than GloVe. For example, it gives a higher cosine similarity scores to the pairs (car, cab), (sea, ocean), (cottage,cabin), (vision, perception) etc. in com-

Dataset	GloVe	W2V	D2V-D	D2V-L	D2V-N
WSSim	**0.799**	0.779	0.737	0.073	0.764
SimL-N	0.427	**0.454**	0.418	0.015	0.421
RG-65	0.791	0.777	0.804	-0.121	**0.813**
MC-30	0.799	0.819	0.859	-0.067	**0.869**
WSR	**0.637**	0.631	0.287	0.077	0.333
M771	**0.707**	0.655	0.636	0.027	0.63
M287	**0.8**	0.755	0.558	-0.027	0.591
MEN-N	**0.819**	0.764	0.619	0.004	0.612
WS-353	**0.706**	0.697	0.51	0.088	0.547

Table 1: Comparison of individual performances of different vector representation models w.r.t. word similarity and relatedness tasks. The performance metric is Spearman's rank correlation coefficient (ρ). Best result of each row in bold showing the best vector representation for each dataset.

parison to GloVe. However, in some cases, where words are not similar but are related, e.g., (airport, flight), (food, meat), (peeper, soup), (harbour, shore), the joint representation gives a lower cosine similarity score than GloVe comparatively. In the next set of evaluation experiments, we observe this utility of joint representation towards word similarity task and word relatedness task to some extent.

4.2 Word Similarity and Relatedness

In this genre of tasks, the human judgment score for each word pair is given; we report the Spearman's rank correlation coefficient (ρ) between human judgment score and the predicted score by distributional model. Note that, we take cosine similarity between vector representations of words in a word pair as the predicted score.

Datasets: We use the benchmark datasets for evaluation of word representations. Four word similarity datasets and four word relatedness datasets are used for that purpose. The descriptions of the word similarity datasets are given below.

WordSim353 Similarity (WSSim) : 203 word pairs extracted from WordSim353 dataset (Finkelstein et al., 2001) by manual classification, prepared by Agirre et al. (2009), which deals with only similarity.

SimLex999 (SimL) : 999 word pairs rated by 500 paid native English speakers, recruited via Amazon Mechanical Turk,[1] who were asked to rate the similarity. This dataset is introduced by Hill et al. (2016).

RG-65 : It consists of 65 word pairs collected by Rubenstein and Goodenough (1965). These word pairs are judged by 51 humans in a scale from 0 to 4 according to their similarity, but ignoring any other possible semantic relationships.

MC-30 : 30 words judged by 38 subjects in a scale of 0 and 4 collected by Miller and Charles (1991).

Similarly, a brief overview of word relatedness datasets is given below:

WordSim353 Relatedness (WSR) : 252 word pairs extracted from WordSim353 (Finkelstein et al., 2001) dataset by manual classification, prepared by Agirre et al. (2009) which deals with only relatedness.

MTURK771 (M771) : 771 word pairs evaluated by Amazon Mechanical Turk workers, with an average of 20 ratings for each word pair, where each judgment task consists of a batch of 50 word pairs. Ratings are collected on a 15 scale. This dataset is introduced by Halawi et al. (2012).

MTURK287 (M287) : 287 word pairs evaluated by Amazon Mechanical Turk workers, with an average of 23 ratings for each word pair. This dataset is introduced by Radinsky et al. (2011).

MEN : MEN consists of 3,000 word pairs with [0, 1]-normalized semantic relatedness ratings provided by Amazon Mechanical Turk workers. This dataset was introduced by Bruni et al. (2014).

Along with these datasets we use the full **WordSim353 (WS-353)** dataset (includes both similarity and relatedness pairs) (Finkelstein et al., 2001) which contains 353 word pairs, each associated with an average of 13 to 16 human judgments in a scale of 0 to 10. Being inspired by Baroni et al. (2014), we consider only noun pairs from **SimL** and **MEN** datasets, which will be denoted as **SimL-N** and **MEN-N** whereas other datasets only contain the noun pairs.

We start with experiments to inspect individual performance of each of the vector representations for each of the datasets. Table 1 represents individual performances of GloVe, Word2vec, *D2V-*

[1] www.mturk.com

Dataset	GloVe	CC (GloVe,D2V-D)	PCA (GloVe,D2V-D)	CC (GloVe,D2V-N)	PCA (GloVe,D2V-N)
WSSim	0.799	0.838	0.839	**0.84**	0.832
SimL-N	0.427	0.443	0.468	0.446	**0.483**
RG-65	0.791	0.816	**0.879**	0.809	0.857
MC-30	0.799	0.86	**0.89**	0.866	0.874
WSR	0.637	**0.676**	0.645	0.67	0.657
M771	0.707	0.708	0.707	0.711	**0.719**
M287	0.8	0.781	0.807	0.795	**0.82**
MEN-N	**0.819**	0.792	0.799	0.806	0.817
WS-353	0.706	**0.751**	0.74	0.75	0.75

Table 2: Comparison of performances (Spearman's ρ) of GloVe against the combined representation of word representations obtained from DT network using network embeddings (DeepWalk, node2vec) with GloVe. Two combination methods – concatenation (CC) and PCA – are used among which PCA performs better than concatenation (CC) in most of the cases. Also the results show that the combined representation leads to better performance in almost all the cases.

Dataset	W2V	CC (W2V,D2V-D)	PCA (W2V,D2V-D)	CC (W2V,D2V-N)	PCA (W2V,D2V-N)
WSSim	0.779	0.774	0.786	**0.806**	0.805
SimL-N	0.454	0.438	0.456	0.448	**0.493**
RG-65	0.777	0.855	0.864	0.867	**0.875**
MC-30	0.819	0.866	0.891	0.903	**0.909**
WSR	**0.631**	0.441	0.443	0.459	0.497
M771	0.655	0.633	0.637	0.656	**0.676**
M287	**0.755**	0.714	0.701	0.722	**0.755**
MEN-N	**0.764**	0.703	0.717	0.714	0.747
WS-353	**0.697**	0.602	0.61	0.623	0.641

Table 3: A similar experiment as Table 2 with Word2vec (W2V) instead of GloVe.

Dataset	PCA (GloVe,W2V)	PCA (GloVe,D2V-N)
WSSim	0.8	**0.832**
SimL-N	0.476	**0.483**
RG-65	0.794	**0.857**
MC-30	0.832	**0.874**
WSR	**0.68**	0.657
M771	0.717	**0.719**
M287	**0.82**	**0.82**
MEN-N	**0.829**	0.817
WS-353	0.746	**0.75**

Table 4: Comparison of performances (Spearman's ρ) between GloVe combined with Word2vec (W2V) against GloVe combined with DT embedding obtained using node2vec (D2V-N). PCA has been taken as combination method. Clearly, DT embedding outperforms Word2vec in terms of enhancing the performance of GloVe.

Dataset	PCA (GloVe, D2V-N)	PCA (GloVe, WN2V)	PCA (GloVe, WN2V, D2V-N)
WSSim	0.832	0.828	**0.853**
SimL-N	0.483	0.525	**0.531**
RG-65	0.857	0.858	**0.91**
MC-30	0.874	0.882	**0.92**
WSR	0.657	**0.699**	0.682
M771	0.719	0.762	**0.764**
M287	**0.82**	0.816	0.81
MEN-N	0.817	**0.848**	0.7993
WS-353	0.75	**0.7801**	0.7693

Table 5: Performance (ρ) reported for three combined representations: GloVe and DT embedding using node2vec (D2V-N), GloVe and WordNet embedding (WN2V), GloVe, WN2V and D2V-N. Results show that, DT embedding produces comparable performance in comparison to the WordNet embedding. Combining DT embedding along with WordNet embedding helps to boost performance further in many of the cases.

D, *D2V-L* and *D2V-N* for different datasets. In most of the cases, GloVe produces the best results although no model is a clear winner for all the datasets. Interestingly, *D2V-D* and *D2V-N* give results comparable to GloVe and Word2vec for the word similarity datasets, even surpassing GloVe and Word2vec for few of these. *D2V-L* gives very poor performance, indicating that considering second order proximity in the DT network while embedding has an adverse effect on performance in word similarity and word relatedness tasks, whereas random walk based *D2V-D* and *D2V-N* which take care of neighborhood and community, produce decent performance. Hence-

Dataset	GloVe	GloVe with retrofitting	PCA (GloVe,D2V-N)
WSSim	0.799	0.799	**0.832**
SimL-N	0.427	0.423	**0.483**
RG-65	0.791	0.791	**0.857**
MC-30	0.799	0.799	**0.874**
WSR	0.637	**0.69**	0.657
M771	0.707	0.708	**0.719**
M287	0.8	0.795	**0.82**
MEN-N	**0.819**	**0.819**	0.817
WS-353	0.706	0.703	**0.75**

Table 6: Comparison of performances (Spearman's ρ) between GloVe representation and retrofitted (by DT network) GloVe representation. Clearly, DT retrofitting is not helping much to improve the performance of GloVe.

forth, we ignore the *D2V-L* model for the rest of our experiments.

Next, we investigate whether network embeddings applied on Distributional Thesaurus network can be combined with GloVe and Word2vec to improve the performance on the pre-specified tasks. In order to do that, we combine the vector representations using two operations: concatenation (CC), and principal component analysis (PCA). Table 2 represents the performance of combining GloVe with *D2V-D* and *D2V-N* for all the datasets using these combination strategies. In general, PCA turns out to be better technique for vector combination than CC. Clearly, combining DT embeddings and GloVe boosts the performance for all the datasets except for the **MEN-N** dataset, where the combined representation produces comparable performance.

In order to ensure that this observation is consistent, we try combining DT embeddings with Word2vec. The results are presented in Table 3 and we see very similar improvements in the performance except for a few cases, indicating the fact that combining word embeddings prepared form DT network is helpful in enhancing performances. From Tables 1, 2 and 3, we see that GloVe proves to be better than word2Vec for most of the cases, *D2V-N* is the best performing network embedding, and PCA turns out to be the best combination technique. Henceforth, we consider PCA (GloVe, *D2V-N*) as our model for comparison with the baselines for the rest of the experiments.

Further, to scrutinize that the achieved result is not just the effect of combining two different word vectors, we compare PCA (GloVe, D2V-N) against combination of GloVe and Word2vec (W2V). Table 4 shows the performance comparison on different datasets and it is evident that PCA (GloVe, D2V-N) gives better results compared to PCA (GloVe, W2V) in most of the cases.

Now, as we observe that the network embedding from DT network helps to boost the performance of Word2vec and GloVe when combined with them, we further compare the performance against the case when text based embeddings are combined with embeddings from lexical resources. For that purpose, we take **one baseline** (Goikoetxea et al., 2016), where authors combined the text based representation with WordNet based representation. Here we use GloVe as the text based representation and PCA as the combination method as prescribed by the author. Note that, WordNet based representation is made publicly available by Goikoetxea et al. (2016). From the second and third columns of Table 5, we observe that even though we do not use any manually created lexical resources like WordNet our approach achieves comparable performance. Additionally we check whether we gain in terms of performance if we integrate the three embeddings together. Fourth column of Table 5 shows that we gain for some of the datasets and for other cases, it has a negative effect. Looking at the performance, we can conclude that automatically generated DT network from corpus brings in useful additional information as far as word similarity and relatedness tasks are concerned.

So far, we use concatenation and PCA as methods for combining two different representations. However, as per the literature, there are different ways of infusing knowledge from different lexical sources to improve the quality of pre-trained vector embeddings. So we compare our proposed way of combination with a completely different way of integrating information from both dimensions, known as *retrofitting*. Retrofitting is a novel way proposed by Faruqui et al. (2015) for refining vector space representations using relational information from semantic lexicons by encouraging linked words to have similar vector representations. Here instead of using semantic lexicons, we use the DT network to produce the linked words to have similar vector representation. Note that, for a target word, we consider only those words as linked words which are having edge weight greater than a certain threshold. While experimenting with various thresholds, the best results were obtained for a

threshold value of 500. Table 6 shows the performance of GloVe representations when retrofitted with information from DT network. Even though in very few cases it gives little improved performance, compared to other combinations presented in Table2, the correlation is not very good, indicating the fact that retrofitting is probably not the best way of fusing knowledge from a DT network.

Further, we extend our study to investigate the usefulness of DT embedding on other NLP tasks like synonym detection, SAT analogy task as will be discussed next.

4.3 Synonym Detection

We consider two gold standard datasets for the experiment of synonym detection. The descriptions of the used datasets are given below.

TOEFL: It contains 80 multiple-choice synonym questions (4 choices per question) introduced by Landauer and Dumais (1997), as a way of evaluating algorithms for measuring degree of similarity between words. Being consistent with the previous experiments, we consider only nouns for our experiment and prepare **TOEFL-N** which contains 23 synonym questions.

ESL: It contains 50 multiple-choice synonym questions (4 choices per question), along with a sentence for providing context for each of the question, introduced by Turney (2001). Here also we consider only nouns for our experiment and prepare **ESL-N** which contains 22 synonym questions. Note that, in our experimental setup we do not use the context per question provided in the dataset for evaluation.

While preparing both the datasets, we also keep in mind the availability of word vectors in both downloaded GloVe representation and prepared DT embedding. For evaluation of the word embeddings using **TOEFL-N** and **ESL-N**, we consider the option as the correct answer which is having highest cosine similarity with the question and report accuracy. From the results presented in Table 7, we see that DT embedding leads to boost the performance of GloVe representation.

4.4 Analogy Detection

For analogy detection we experiment with **SAT** analogy dataset. This dataset contains 374 multiple-choice analogy questions (5 choices per question) introduced by Turney and Bigham (2003) as a way of evaluating algorithms for measuring relational similarity. Considering only noun questions, we prepare **SAT-N**, which contains 159 questions.

Dataset	GloVe	D2V-N	PCA (GloVe, D2V-N)
TOEFL-N	0.826	0.739	**0.869**
ESL-N	0.636	0.591	**0.682**
SAT-N	0.465	0.509	**0.515**

Table 7: Comparison of accuracies between GloVe representation, DT embedding using node2vec and combination of both where PCA is the combination technique. Clearly DT embedding is helping to improve the performance of GloVe for synonym detection as well as analogy detection.

In order to find out the correct answer from the 5 options given for each question, we take up a score (s) metric proposed by Speer et al. (2017), where for a question 'a_1 is to b_1', we will consider 'a_2 is to b_2' as the correct answer among the options, whose score (s) is the highest. Score (s) is defined by the author as follows:

$$s = a_1.a_2 + b_1.b_2 + w_1(b_2 - a_2).(b_1 - a_1) + w_2(b_2 - b_1).(a_2 - a_1)$$

As mentioned by the authors, the appropriate values of w_1 and w_2 are optimized separately for each system using grid search, to achieve the best performance. We use accuracy as the evaluation metric. The last row of Table 7 presents the comparison of accuracies (best for each model) obtained using different embeddings portraying the same observation that combination of GloVe and DT embeddings leads to better performance compared to GloVe and DT embeddings when used separately. Note that, the optimized values of (w_1, w_2) are (0.2,0.2), (0.8,0.6), (6,0.6) for GloVe, DT embedding, combined representation of GloVe and DT embeddings, respectively, for the analogy task.

5 Conclusion

In this paper we showed that both dense count based model (GloVe) and predictive model (Word2vec) lead to improved word representation when they are combined with word representation learned using network embedding methods on Distributional Thesaurus (DT) network. We tried with various network embedding models among which *node2vec* proved to be the best in our experimental setup. We also tried with different methodologies to combine vector representations and PCA turned out to be the best among them. The combined vector representation of words yielded the better performance for most

of the similarity and relatedness datasets as compared to the performance of GloVe and Word2vec representation individually. Further we observed that we could use the information from DT as a proxy of WordNet in order to improve the state-of-the-art vector representation as we were getting comparable performances for most of the datasets. Similarly, for synonym detection task and analogy detection task, the same trend of combined vector representation continued, showing the superiority of the combined representation over state-of-the-art embeddings. All the datasets used in our experiments which are not under any copyright protection, along with the DT embeddings are made publicly available[2].

In future we plan to investigate the effectiveness of the joint representation on other NLP tasks like text classification, sentence completion challenge, evaluation of common sense stories etc. The overall aim is to prepare a better generalized representation of words which can be used across languages in different NLP tasks.

References

Clémentine Adam, Cécile Fabre, and Philippe Muller. 2013. Évaluer et améliorer une ressource distributionnelle. *Traitement Automatique des Langues* 54(1):71–97.

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 19–27.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL (1)*. pages 238–247.

Chris Biemann and Martin Riedl. 2013. Text: Now in 2d! a framework for lexical expansion with contextual similarity. *Journal of Language Modelling* 1(1):55–95.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. AcM, pages 1247–1250.

[2] https://tinyurl.com/yamrutrm

Fabian Both, Steffen Thoma, and Achim Rettinger. 2017. Cross-modal knowledge transfer: Improving the word embedding of apple by looking at oranges. In *K-CAP2017, The 9th International Conference on Knowledge Capture*. International Conference on Knowledge Capture, ACM.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *J. Artif. Intell. Res.(JAIR)* 49(2014):1–47.

Alexander Budanitsky and Graeme Hirst. 2006. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics* 32(1):13–47.

James R Curran and Marc Moens. 2002. Improvements in automatic thesaurus extraction. In *Proceedings of the ACL-02 workshop on Unsupervised lexical acquisition-Volume 9*. Association for Computational Linguistics, pages 59–66.

Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of NAACL*.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. Association for Computational Linguistics.

Olivier Ferret. 2017. Turning distributional thesauri into word vectors for synonym extraction and expansion. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. volume 1, pages 273–283.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*. ACM, pages 406–414.

John R Firth. 1957. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis* .

Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. 2013. Devise: A deep visual-semantic embedding model. In *Advances in neural information processing systems*. pages 2121–2129.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In *HLT-NAACL*. pages 758–764.

Josu Goikoetxea, Eneko Agirre, and Aitor Soroa. 2016. Single or multiple? combining word representations independently learned from text and wordnet. In *AAAI*. pages 2608–2614.

Josu Goikoetxea, Aitor Soroa, Eneko Agirre, and Basque Country Donostia. 2015. Random walks and neural network language models on knowledge bases. In *HLT-NAACL*. pages 1434–1439.

Yoav Goldberg and Jon Orwant. 2013. A dataset of syntactic-ngrams over time from a very large corpus of english books. In *Second Joint Conference on Lexical and Computational Semantics (* SEM)*. volume 1, pages 241–247.

Gregory Grefenstette. 2012. *Explorations in automatic thesaurus discovery*, volume 278. Springer Science & Business Media.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 855–864.

Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. 2012. Large-scale learning of word relatedness with constraints. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 1406–1414.

Per Christian Hansen. 1987. The truncatedsvd as a method for regularization. *BIT Numerical Mathematics* 27(4):534–553.

Felix Hill, Roi Reichart, and Anna Korhonen. 2016. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics* .

Ian T Jolliffe. 1986. Principal component analysis and factor analysis. In *Principal component analysis*, Springer, pages 115–128.

Douwe Kiela and Léon Bottou. 2014. Learning image embeddings using convolutional neural networks for improved multi-modal semantics. In *EMNLP*. pages 36–45.

Adam Kilgarriff, Pavel Rychly, Pavel Smrz, and David Tugwell. 2004. Itri-04-08 the sketch engine. *Information Technology* 105:116.

Thomas K Landauer and Susan T Dumais. 1997. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review* 104(2):211.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*. pages 2177–2185.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics* 3:211–225.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th international conference on Computational linguistics-Volume 2*. Association for Computational Linguistics, pages 768–774.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

George A Miller and Walter G Charles. 1991. Contextual correlates of semantic similarity. *Language and cognitive processes* 6(1):1–28.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pages 1532–1543.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, KDD '14, pages 701–710. https://doi.org/10.1145/2623330.2623732.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th international conference on World wide web*. ACM, pages 337–346.

Pushpendre Rastogi, Benjamin Van Durme, and Raman Arora. 2015. Multiview lsa: Representation learning via generalized cca. In *HLT-NAACL*. pages 556–566.

Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. 2017. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pages 385–394.

Martin Riedl and Chris Biemann. 2013. Scaling to large3 data: An efficient and effective method to compute distributional thesauri. In *EMNLP*. pages 884–890.

Herbert Rubenstein and John B Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM* 8(10):627–633.

Robert Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI*. pages 4444–4451.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, pages 1067–1077.

Steffen Thoma, Achim Rettinger, and Fabian Both. 2017. Knowledge fusion via embeddings from text, knowledge graphs, and images. *arXiv preprint arXiv:1704.06084* .

Fei Tian, Bin Gao, En-Hong Chen, and Tie-Yan Liu. 2016. Learning better word embedding by asymmetric low-rank projection of knowledge graph. *Journal of Computer Science and Technology* 31(3):624–634. https://doi.org/10.1007/s11390-016-1651-5.

Peter Turney. 2001. Mining the web for synonyms: Pmi-ir versus lsa on toefl. *Machine Learning: ECML 2001* pages 491–502.

Peter D Turney and Jeffrey Bigham. 2003. Combining independent modules to solve multiple-choice synonym and analogy. In *Problems. Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP-03*. Citeseer.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph and text jointly embedding. In *EMNLP*. volume 14, pages 1591–1601.

Deep Neural Models of Semantic Shift

Alex Rosenfeld
The University of Texas at Austin
Department of Linguistics
alexbrosenfeld@gmail.com

Katrin Erk
The University of Texas at Austin
Department of Linguistics
katrin.erk@mail.utexas.edu

Abstract

Diachronic distributional models track changes in word use over time. In this paper, we propose a deep neural network diachronic distributional model. Instead of modeling lexical change via a time series as is done in previous work, we represent time as a continuous variable and model a word's usage as a function of time. Additionally, we have created a novel synthetic task, which quantitatively measures how well a model captures the semantic trajectory of a word over time. Finally, we explore how well the derivatives of our model can be used to measure the speed of lexical change.

1 Introduction

Diachronic distributional models have provided interesting insights into how words change meaning. Generally, they are used to explore how specific words have changed meaning over time (Sagi et al., 2011; Gulordava and Baroni, 2011; Jatowt and Duh, 2014; Kim et al., 2014; Kulkarni et al., 2015; Bamler and Mandt, 2017; Hellrich and Hahn, 2017), but they have also been used to explore historical linguistic theories (Xu and Kemp, 2015; Hamilton et al., 2016a,b), to predict the emergence of novel senses (Bamman and Crane, 2011; Rohrdantz et al., 2011; Cook et al., 2013, 2014), and to predict world events (Kutuzov et al., 2017a,b).

Diachronic distributional models are distributional models where the vector for a word changes over time. Thus, we can calculate the cosine similarity between the vectors for a word at two different time points to measure how much that word has changed over time and we can perform a nearest neighbor analysis to understand in what direction a word is changing. For example, diachronic distributional models can detect that the word *gay* has greatly changed by comparing the word vector for *gay* across different time points. They can also be used to discover that gay has shifted its meaning from *happy* to *homosexual* by analyzing when those words show up as nearest neighbors to *gay*.

Previous research in diachronic distributional semantics has used models where data is partitioned into time bins and a synchronic model is trained on each bin. A synchronic model is a vanilla, time-independent distributional model, such as skip-gram. However, there are several technical issues associated with data binning. For example, if the bins are too large, you can only achieve extremely coarse grained representations of lexical change over time. However, if the bins are too small, the synchronic models get trained on insufficient data.

In this paper, we have built the first diachronic distributional model that represents time as a continuous variable instead of employing data binning. There are several advantages to treating time as continuous. The first advantage is that it is more realistic. Large scale change in the meaning of a word is the result of change happening one person at a time. Thus, semantic change must be a gradual process. By treating time as a continuous variable, we can capture this gradual shift. The second advantage is that it allows a greater representation of the underlying causes behind lexical change. Words change usage in reaction to real world events and multiple words can be affected by the same event. For example, the usage of *gay* and *lesbian* have changed in similar ways due to changing perceptions of homosexuality in society. By associating time with a vector and having word representations be a function of that vector, we can model a single underlying cause affecting multiple words similarly.

Proceedings of NAACL-HLT 2018, pages 474–484
New Orleans, Louisiana, June 1 - 6, 2018.

It is difficult to evaluate diachronic distributional models in their ability to capture semantic shift as it is extremely difficult to acquire gold data. Distributional models are traditionally evaluated with word similarity judgments, which we cannot obtain for word usage in the past. Thus, evaluation of diachronic distributional models is a focus of research, such as work done by Hellrich and Hahn (2016) and Dubossarsky et al. (2017). Our approach is to create a synthetic task to measure how well a model captures gradual semantic shifts.

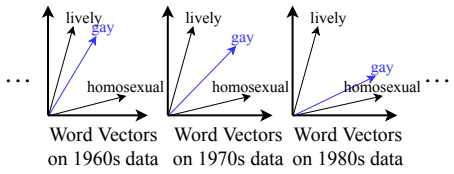
We will also explore how we can use our model to predict the speed at which a word changes. Our model is differentiable with respect to time, which gives us a natural way to measure the velocity, and thus speed, of a word at a given time. We explore the capabilities and limitations of this approach.

In short, our paper provides the following contributions:

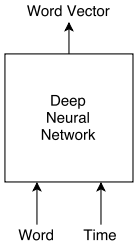
- We have developed the first continuous diachronic distributional model. This is also the first diachronic distributional model using a deep neural network.
- We have designed an evaluation of a model's ability to capture semantic shift that tracks gradual change.
- We have used the derivatives of our model as a natural way to measure the speed of word use change.

2 Related work

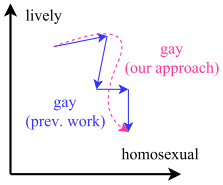
Previous research in diachronic distributional models has applied a binning approach. In this approach, researchers partition the data into bins based on time and train a synchronic distributional model on that bin's data (See Figure 1). Several authors have used large bin models in their research, such as using five year sized bins (Kulkarni et al., 2015), decade sized bins (Gulordava and Baroni, 2011; Xu and Kemp, 2015; Jatowt and Duh, 2014; Hamilton et al., 2016a,b; Hellrich and Hahn, 2016, 2017), and era sized bins (Sagi et al., 2009, 2011). The synchronic model for each time bin was trained independently of the others. In order to get a fine grained representation of semantic shift, several authors have used small bins. Kim et al. (2014) trained a synchronic model for each time bin. To mitigate data issues, Kim et al. preinitialized a time bin's synchronic model with the



(a) Previous work



(b) Our approach



(c) Difference in trajectories

Figure 1: Difference between our approach and previous work. Previous work in diachronic distributional models (a) has trained synchronic distributional models on consecutive time bins. In our work (b), a neural network takes word and time as input and produces a time specific word vector. In (c), we sketch that previous work produces a jagged semantic trajectory (blue, solid curve) whereas our model produces a smooth semantic trajectory (pink, dotted curve).

model from the previous time bin. Bamler and Mandt (2017) developed a small bin probabilistic approach that used transition probabilities to lessen data issues. They have two versions of their method. The first version trains the distribution in each bin iteratively and the second version trains a joint distribution over all bins. In this paper, we only explore the first version as the second version does not scale well to large vocabulary sizes. Following Bamler and Mandt (2017), we compare to models used by Hamilton et al. (2016b), Kim et al. (2014), and the first version of Bamler and Mandt's's model.

There have been other models of lexical change beside distributional ones. Topic modeling has been used to see how topics associated to a word have changed over time (Wijaya and Yeniterzi, 2011; Frermann and Lapata, 2016). Sentiment analysis has been applied to determine how sentiments associated to a word have changed over time (Jatowt and Duh, 2014).

As mentioned in the introduction, it is difficult to quantitatively evaluate diachronic distributional models due to the lack of gold data. Thus, previous research has attempted alternative routes to quantitatively evaluate their models. One route

is to use intrinsic evaluations, such as measuring a trajectory's smoothness (Bamler and Mandt, 2017). However, intrinsic measures do not directly measure semantic shift, which is the main use of diachronic distributional models. Hamilton et al. (2016b) use attested shifts generated by historical linguists. However, outside of first attestations, it is a difficult task for historical linguists themselves to accurately detail semantic shifts (Deo, 2015). Additionally, the task used by Hamilton et al. is unusable for model comparison as all but one model had a 100% accuracy in this task. Kulkarni et al. (2015) used a synthetic task to evaluate how well diachronic distributional models can detect semantic shift. They took 20 copies of wikipedia where each is a synthetic version of a time bin and changed several words in the last 10 copies. Models were then evaluated on their ability to detect when those words changed. Our evaluation improves upon this one by having the test data be from a diachronic corpus and we model lexical change as a gradual process rather than searching for a single change point.

3 Models

In this section, we describe the four diachronic distributional models that we analyze in our current work. Three will be from previous research to be used as benchmarks. Each of the four models we analyze are based on skip-gram with negative sampling (**SGNS**). The difference between the four diachronic distributional models we analyze is how they apply SGNS to changes over time.

Skip-gram with negative sampling (SGNS) is a word embedding model that learns a latent representation of word usage (Mikolov et al., 2013). For target words w and context words c, vector representations $\vec{w}$ and $\vec{c}$ are learned to best predict if c will be in context of w in a corpus. k negative contexts are randomly sampled for each positive context. Vector representations are computed by optimizing the following loss function:

$$\sum_{(w,c)\in D} [log(\sigma(\vec{w}\cdot\vec{c}))+ \sum_{c_1,...c_k\sim P_D} log(1-\sigma(\vec{w}\cdot\vec{c_i}))] \quad (1)$$

where D is a list of target-context pairs extracted from the corpus, P_D is the unigram distribution on the corpus, σ is the sigmoid function, and k is the number of negative samples.

3.1 Binning by Decade

The first diachronic distributional model we will consider is a large time bin model proposed by Hamilton et al. (2016b). Here, time is partitioned into decades and an SGNS model is trained on each decade's worth of data. We label this model **LargeBin**.

3.2 Preinitialization

The second diachronic distributional model we will consider is a small time bin model proposed by Kim et al. (2014). Here, time is partitioned into years and an SGNS model is trained on each year's worth of data. Data issues are mitigated by preinitializing the model[1] for a given time bin with the vectors of the preceding time bin (Kim et al., 2014). We label this model **SmallBinPreInit**.

3.3 Prior and Transition Probabilities

The third diachronic distributional model we will consider comes from Bamler and Mandt (2017). Bamler and Mandt take a probabilistic approach to modeling semantic change over time. The idea is to transform the SGNS loss function into a probability distribution over the target and context vectors. Then, to create a better diachronic distributional model, they apply priors to this distribution.

The first two priors are Gaussian distributions with mean zero on the vector variables to discourage the vectors from growing too large (Barkan, 2017). More formally:

$$\begin{aligned} P_1(\vec{w}) &= \mathcal{N}(0, \alpha_1 I) \\ P_2(\vec{c}) &= \mathcal{N}(0, \alpha_1 I) \end{aligned} \quad (2)$$

where α_1 is a hyperparameter.

The last two priors are also Gaussian distributions on the vector variables. The means are the vector representation from the previous bin. The goal of this prior is to discourage a vector variable from deviating from the previous bin's vectors.

$$\begin{aligned} P_3(\vec{w}) &= \mathcal{N}(\overrightarrow{w_{prev}}, \alpha_2 I) \\ P_4(\vec{c}) &= \mathcal{N}(\overrightarrow{c_{prev}}, \alpha_2 I) \end{aligned} \quad (3)$$

where α_2 is a hyperparameter and $\overrightarrow{w_{prev}}$ and $\overrightarrow{c_{prev}}$ are the vectors from the previous time bin.

We are only exploring point models, thus we take the maximum a posteriori estimate of the

[1] We do not perform preinitialization in LargeBin as large bin models are less susceptible to data issues.

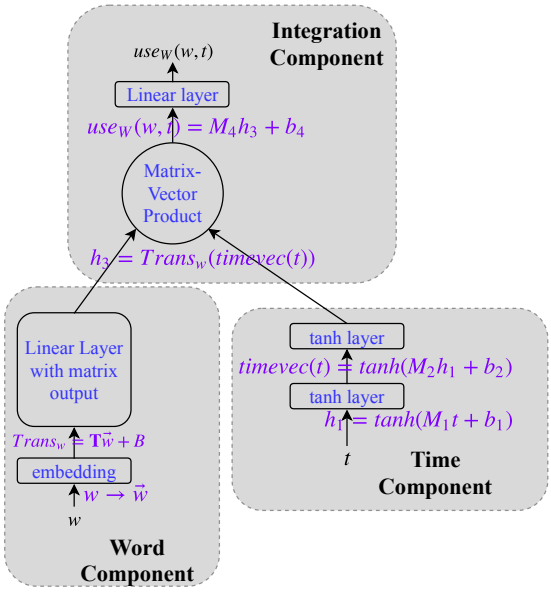


Figure 2: Diagram of DiffTime. $timevec(t)$ encodes temporal information as a vector. M_W encodes lexical information as a matrix. The target vector for w at time t, $use_W(w,t)$, is found by combining $Trans_w$ and $timevec(t)$. Context version $use_C(c,t)$ is the same except that it has its own embedding layer.

joint distribution to recover the vectors for each time bin. We apply a logarithm in constructing the estimate, which transforms the joint probability into the SGNS loss function with four regularizers (each one corresponding to a prior distribution). The prior distribution P_1 becomes $\sum_{w \in W} \frac{\alpha_1}{2}||w||$. The prior distribution P_2 becomes $\sum_{c \in C} \frac{\alpha_1}{2}||c||$. The prior distribution P_3 becomes $\sum_{w \in W} \frac{\alpha_2}{2}||\vec{w} - \overrightarrow{w_{prev}}||$. The prior distribution P_4 becomes $\sum_{c \in C} \frac{\alpha_2}{2}||\vec{c} - \overrightarrow{c_{prev}}||$. W and C are the sets of target and context words. We label this model **SmallBinReg**.

3.4 DiffTime Model

Our model is a modification of the SGNS algorithm to accommodate a continuous time variable. The original SGNS algorithm produces a target embedding $\vec{w}$ for target word w and a context embedding $\vec{c}$ for context word c. Instead, we produce a differentiable function $use_W(w,t)$ that returns a target embedding for target word w at time t and a differentiable function $use_C(c,t)$ that produces a context embedding for context word c at time t.

Our model consists of three components. One component takes time as its input and produces an embedding that characterizes that point in time (lower right). The second component (lower left) takes a word as its input and produces a time-independent word embedding, which is then reshaped into a set of parameters that can modify the time embedding. The third component (top) combines the time embedding and the word embedding.

The first component of our model is a two-layer feed-forward neural network with tanh activation functions. These layers take a time t as input and produces a time embedding $timevec(t)$ as output of those layers:

$$\begin{aligned} h_1 &= tanh(M_1t + b_1) \\ timevec(t) &= tanh(M_2h_1 + b_2) \end{aligned} \tag{4}$$

where M_1 and M_2 are the weights of the first two layers and b_1 and b_2 are the biases. To produce the input value t, a timepoint is scaled to a value between 0 and 1, where 0 corresponds to the year 1900, and 1 corresponds to 2009, the last year for which our corpus has data.

The second component incorporates word-specific information into our model. For $use_W(w,t)$, each target word w has a target vector representation $\vec{w}$. The vector $\vec{w}$ is then transformed into a linear transformation $Trans_w$, which in the third component is applied to the time embedding $timevec(t)$. We do this via a modified linear layer where the weights are a three dimensional tensor, the biases are a matrix and the output is a matrix:

$$Trans_w = \mathbf{T}\vec{w} + B \tag{5}$$

where $\mathbf{T}$ is the tensor acting as the weights and B is the matrix acting as the biases.

The third component combines the word-independent time embedding $timevec(t)$ and the time-independent linear transformation $Trans_w$ together to produce the final result. First, $Trans_w$ is applied to $timevec(t)$:

$$h_3 = Trans_w(timevec(t)) \tag{6}$$

Then, an additional linear layer is used as the output layer, taking h_3 as input:

$$use_W(w,t) = M_4h_3 + b_4 \tag{7}$$

where M_4 and b_4 are the weights and biases of the output layer.

The above details the architecture of $use_W(w,t)$. The corresponding function $use_C(c,t)$ for context words has the same architecture as $use_W(w,t)$ and shares weights with $use_W(w,t)$. The only exception is that $use_C(c,t)$ uses a separate set of vectors $\vec{c}$ in the second component instead of sharing the target vectors $\vec{w}$ with $use_W(w,t)$.

We train our model using a modified version of the SGNS loss function. In particular, our positive samples are now triples (w,c,t) where w is a target word, c is a context word, and t is a time, instead of pairs (w,c) which are typically used in SGNS. For each positive sample (w,c,t), we sample k negative contexts from the unigram distribution, P_D. P_D is trained from all contexts in the entire corpus and is time-independent. Explicitly, the loss function is:

$$\sum_{(w,c,t)\in D} log(\sigma(use_W(w,t)\cdot use_C(c,t)))+ kE_{c_N\sim P_D}[log(\sigma(-use_W(w,t)\cdot use_C(c_N,t)))] \quad (8)$$

3.5 Training

All models are trained on the same training data. We used the English Fiction section of the Google Books ngram corpus (Lin et al., 2012). We use the English fiction specifically, because it is less unbalanced than the full English section and less influenced by technical texts (Pechenick et al., 2015). We only use the years 1900 to 2009 as there is limited data before 1900.

We converted the ngram data for this corpus into a set of (target word, context word, year, frequency) tuples. The frequency is the expected number of times the target word-context word pair is sampled from that year's data using skip-gram. Following Hamilton et al. (2016b), we use subsampling with $t = 10^{-5}$. As the number of texts published since 1900 has increased five fold, we weigh the frequencies so that the sums across each year are equal.

For the binned models, we train each bin's synchronic model using the subset of the training data corresponding to that time bin. For our model, we sample (training word, context word, year) triples from the entire training data as the year is an input to our function.

4 Evaluation

4.1 Synchronic Accuracy

Method	Time	Spearman's ρ
LargeBin	1990s bin	0.615
SmallBinPreInit	1995 bin	0.489
SmallBinReg	1995 bin	0.564
DiffTime	start of 1995	**0.694**

Table 1: Synchronic accuracy of the methods. Time is the point of time we use as our synchronic model.

Before we can evaluate the methods as models of diachronic semantics, we must first ensure that the methods model semantics accurately. To do this, we follow Hamilton et al. (2016b) by performing the MEN word similarity task on vectors extracted from a fixed time point (Bruni et al., 2012). The hope is that the word similarity predictions of a model at that point in time highly correlate with word similarity judgments in the MEN dataset. For the binned models, we used the vectors from the bin best corresponding to 1995 to reflect the 1990s bin chosen by Hamilton et al. (2016b). DiffTime represents time as a continuous variable, so we chose a time t that corresponds to the start of 1995.

The results of MEN word similarity tasks is in Table 1. All of the Spearman's ρ values are comparable to those found in Levy and Goldberg (2014) and Hamilton et al. (2016b). Thus, all of these models reflect human judgments comparable to synchronic models. Thus, the predictions of the models correlate with human judgments.

4.2 Synthetic Task

The goal of creating diachronic distributional models is to help us understand how words change meaning over time. To that end, we have created a synthetic task to compare models by how accurately they track semantic change.

Our task creates synthetic words that change between two senses over time via a sigmoidal path. A sigmoidal path will allow us to emulate a word starting from one sense, shifting gradually to a second sense, then stabilizing on that second sense. By using sigmoidal paths, we can explore how well a model can track words that have switched senses over time such as *gay* (lively to homosexual) and *broadcast* (scattering seeds to televising shows). A similar task is used to evaluate word

sense disambiguation (Gale et al., 1992; Schütze, 1992).

The synthetic words are formed by a combination of two real words, e.g. *banana* and *lobster* are combined together to form $banana{\circ}lobster$. The real words are randomly sampled from two distinct semantic classes from the BLESS dataset (Baroni and Lenci, 2011). We use BLESS classes so that we can capture how semantically similar a synthetic word is to its component words by comparing to other words in the same BLESS classes as the component word. For example, we can capture how similar $banana{\circ}lobster$ is to $banana$ by comparing $banana{\circ}lobster$ to words in the fruit BLESS class. See Appendix B for preprocessing details. We denote the synthetic words with $r_1{\circ}r_2$ where r_1 and r_2 are the component real words.

We also randomly generate the sigmoidal path by which a synthetic word changes from one sense to another. For real words r_1 and r_2, this path will be denoted $\textit{shift}(t; r_1{\circ}r_2)$ and is defined by the following equation:

$$\textit{shift}(t; r_1{\circ}r_2) = \sigma(s(t-m)) \tag{9}$$

The value s is uniformly sampled from $(\frac{1.0}{110}, \frac{10.0}{110})$ and represents the steepness of the sigmoidal path. The value m is uniformly sampled from $\{1930, \ldots, 1980\}$ and represents the point where the synthetic word is equally both senses. For our example synthetic word $banana{\circ}lobster$, $banana{\circ}lobster$ can transition from meaning banana to meaning lobster via the sigmoidal path $\sigma(0.05(t-1957))$ where 1957 is the time where $banana{\circ}lobster$ is equally banana and lobster and 0.05 represents how gradually $banana{\circ}lobster$ shifts senses.

We then use $\textit{shift}(t; r_1{\circ}r_2)$ to integrate $r_1{\circ}r_2$ into the real diachronic corpus data. Our training data is a set of (target word, context word, year, frequency) tuples extracted from a diachronic corpus (see 3.5). For every tuple where r_1 is the target word, we replace the target word with $r_1{\circ}r_2$ and we multiply the frequency by $\textit{shift}(t; r_1{\circ}r_2)$. For every tuple where r_2 is the target word, we replace the target word with $r_1{\circ}r_2$ and we multiply the frequency by $1 - \textit{shift}(t; r_1{\circ}r_2)$. In other words, in the modified corpus, $r_1{\circ}r_2$ has $\textit{shift}(t; r_1{\circ}r_2)$ percent of r_1's contexts at time t and $1 - \textit{shift}(t; r_1{\circ}r_2)$ percent of r_2's contexts at time t.

We train a model mod on this modified training data. This provides a representation for $r_1{\circ}r_2$ over time. We can capture how much a model predicts $r_1{\circ}r_2$ is more semantically similar to r_1 than r_2 by comparing mod's representation of $r_1{\circ}r_2$ to words in the same semantic category as r_1 and r_2. We use BLESS classes as our notion of semantic category. If cls_1 is the BLESS class of r_1 and cls_2 is the BLESS class of r_2, then mod's prediction for how much more similar $r_1{\circ}r_2$ is to r_1 than r_2, $rec(t; r_1{\circ}r_2, mod)$, is defined as follows:

$$\begin{aligned} rec(t; r_1{\circ}r_2, mod) = \\ \frac{1}{|cls_1|} \sum_{r_1' \in cls_1} sim^{mod}(r_1{\circ}r_2, r_1', t) \\ - \frac{1}{|cls_2|} \sum_{r_2' \in cls_2} sim^{mod}(r_1{\circ}r_2, r_2', t) \end{aligned} \tag{10}$$

$sim^{mod}(r_1{\circ}r_2, r_1', t)$ is the cosine similarity between mod's word vector for $r_1{\circ}r_2$ at time t and mod's word vector for r_1' at time t.

Method	AMSE 1900–2009	AMSE 1950–2009
LargeBin	62.52	51.71
SmallBinPreInit	171.43	49.88
SmallBinReg	106.79	42.67
DiffTime	**25.67**	**11.48**

Table 2: Model performance under the synthetic evaluation. The values are the mean sum of squares error (MSSE) for each method. Lower value is better. The first column is MSSE using all times. The second column is MSSE using years 1950 to 2009.

To evaluate a model in its ability to capture semantic shift, we use the mean sum of squares error (MSSE) between $rec(t; r_1{\circ}r_2, mod)$ and $\textit{shift}(t; r_1{\circ}r_2)$ across all synthetic words. The function $rec(t; r_1{\circ}r_2, mod)$ is model mod's prediction of how much more similar $r_1{\circ}r_2$ is to r_1 than r_2. The gold value of $rec(t; r_1{\circ}r_2, mod)$ would then be the sigmoidal path that defines how $r_1{\circ}r_2$ semantically shifts from r_1 to r_2 over time, $\textit{shift}(t; r_1{\circ}r_2)$. To evaluate how accurately mod predicted the semantic trajectory of $r_1{\circ}r_2$, we calculate the mean squared error between $rec(t; r_1{\circ}r_2, mod)$ and $\textit{shift}(t; r_1{\circ}r_2)$ as follows:

$$\sum_{t=1900}^{2009} (rec(t; r_1 \circ r_2, mod) - \textit{shift}(t; r_1 \circ r_2))^2 \quad (11)$$

As $rec(t; r_1 \circ r_2, mod)$ and $\textit{shift}(t; r_1 \circ r_2)$ have different scales, we Z-scale both the $rec(t; r_1 \circ r_2, mod)$ values and the $\textit{shift}(t; r_1 \circ r_2)$ values before calculating the mean squared error.

We use three sets of 15 synthetic words and the average is calculated over all 45 words. The synthetic words and BLESS classes we used are contained in the supplementary material. The results are in Table 2. The column AMSE is MSSE when all years are taken into account. Kim et al. (2014) noted that small bin models require an initialization period, so the column AMSE (1950-) is MSSE when only years 1950 to 2009 are taken into account and the years 1900 to 1949 are used as the initialization period. From the table, we see our model outperforms the three benchmark models in both cases. Using a paired t-test, we found that the reduction in MSSE between our model and the benchmark models are statistically significant.

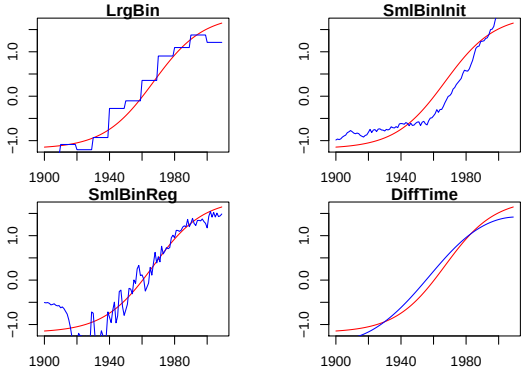


Figure 3: Graph Comparisons between $\textit{shift}(t; r_1 \circ r_2)$ (red) and $rec(t; r_1 \circ r_2,)$ (blue) for the synthetic word $pistol \circ elm$. The x-axis are the years and the y-axis are the values of $\textit{shift}(t; r_1 \circ r_2)$. $rec(t; r_1 \circ r_2, mod)$ and $\textit{shift}(t; r_1 \circ r_2)$ have been Z-scaled.

In Figure 3, we plot $\textit{shift}(t; r_1 \circ r_2)$ and $rec(t; r_1 \circ r_2, mod)$ for the synthetic word $pistol \circ elm$. Each method has a subgraph. The predictions of the large bin model LargeBin appear as a step function with large steps (top left graph). These large steps seem to cause the predicted shift (blue curve) to poorly correlate with the gold shift (red curve). Next, we consider the small bin models SmallBinPreInit (top right graph) and SmallBinReg (bottom left graph). Both predicted shifts have an initial portion that poorly fits the generated shift (between 1900 and 1950). From Kim et al. (2014), it takes several iterations for small bin models to stabilize due to each bin being fed limited data. Additionally, there are fluctuations in the graphs of the predicted shift, which we attribute to the high variance of data per bin. In contrast to the other models, our predicted shift tightly fits the gold shift (bottom right graph).

Although this evaluation provides useful information on the quality of an diachronic distributional model, it has some weaknesses. The first is that it is a synthetic task that operates on synthetic words. Thus, we have limited ability to understand how well a model will perform on real world data. Second, we only generate words that shift from one sense to another. This fails to account for other common changes, such as gaining/losing senses and narrowing/broadening. Finally, by using a sigmoidal function to generate how words change meaning, we may have privileged continuous models that incorporate a sigmoidal function in their architecture. We are working towards improving this evaluation to remove these issues.

4.3 Speed of word use change

In this section, we evaluate our model's ability to measure the speed at which a word is changing. Our model is differentiable with respect to time. Thus, we can get the derivative of $use_W(w, t)$ with respect to t to model how word w is changing usage at time t. We $l2$-normalize $use_W(w, t)$ beforehand to reduce frequency effects. We then get the magnitude of this normalized derivative to model the speed at which a word is changing at a given time.

We explore the connection between speed and the nearest neighbors to a word in Figure 4. First, we use *apple* as a baseline for discussion. We chose *apple*, because the meaning of the word has remained relatively stable throughout the 1900s. With *apple*, we see a low speed over time and a consistency in the cosine similarity to *apple*'s nearest neighbors. While it is true that *apple* has other meanings beyond the fruit, such as referring to Apple Inc., those meanings are much rarer, especially in the fiction corpus we use.

In contrast to *apple*, the word *gay* has a very high speed and a drastic change for *gay*'s nearest

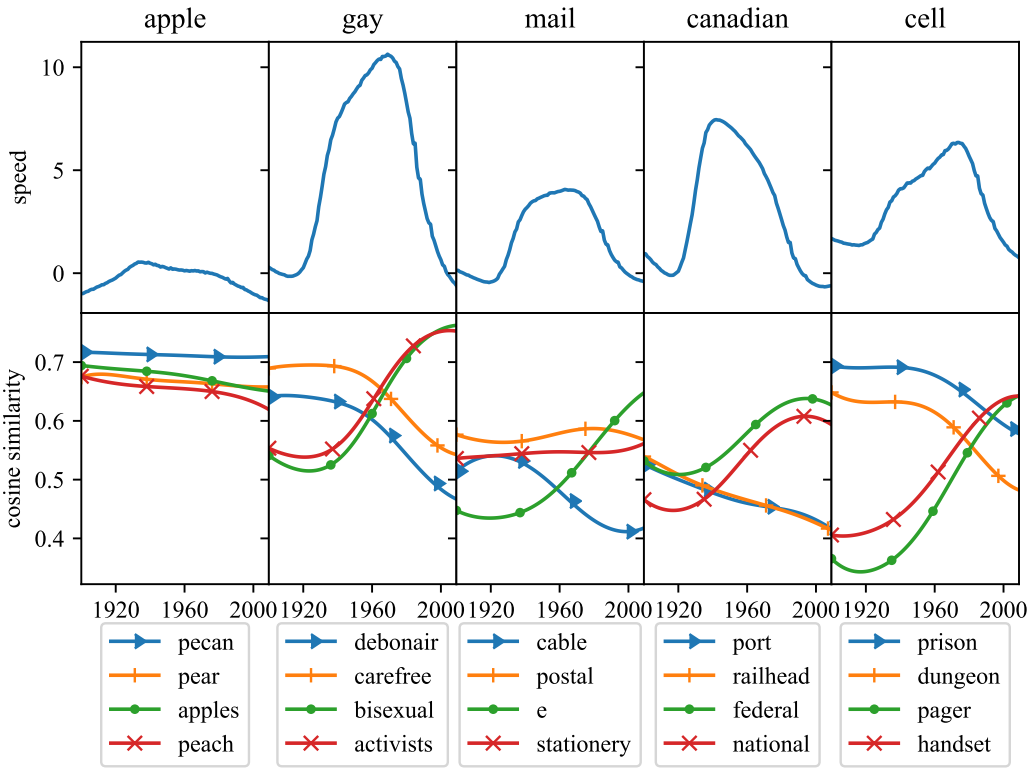


Figure 4: Speed and nearest neighbors over time of selected words. The top graphs as the speed at which a word changes usage according to our model. The bottom graphs are selected nearest neighbors for those words. Each of the chosen nearest neighbors appear as a top 10 nearest neighbor to the word at some year.

neighbors. This makes sense as *gay* is well established to have experienced a drastic sense change in the mid to late 1900s (Harper, 2014).

Next, we explore the word *mail*. The word *mail* has a moderately high speed. This may be reflective of the fact that there have been incredible changes in the medium by which we send mail, e.g. changing from cables to email. A possible reason for the speed only being moderately high is that, even though the medium by which we send mail has changed, many of the same uses of mail, e.g. sending, receiving, opening, etc., remain the same. We see this reflected in the nearest neighbors as well as *mail* shifts from a high similarity to *cable* to a high similarity to *e* (as in email), yet *mail* is consistently similar to *postal* and *stationery*.

The next word we will explore is the word *canadian*. We chose this word as we were surprised to find that *canadian* has one of the fastest speeds in the 1930s to 1940s. The nearest neighbors to *canadian* have shifted from geographic terms like *port* and *railhead* to civil terms like *federal* and *national*. In further analysis, we discovered that this may be reflective of a larger push to form a Canadian identity in the early 1900s (Francis, 1997). The nearest neighbors to *canadian* may reflect the change from being a part of the British Empire to having its own unique national identity.

The final word we will explore is *cell*. The word *cell* also has a high speed over time. However, there is a spike in the speed during the 1980s. Analyzing the nearest neighbors we see a rapid rise in similarity to *pager* and *handset*, which indicates that this spike may be related to the rapid rise of cell phone use. Additionally, this example demonstrates a weakness in our approach. Our graph shows that our model predicts that the word *cell* gradually changed meaning over time and that *cell* started changing meaning much earlier than expected. This prediction error comes from the smoothing out of the output caused by representing time as a continuous variable.

Even though we are able to extract interesting insights from the speed of word use change, Figure 4 also exhibits some limitations. In particu-

lar, most words have a sharp rise in speed in the 1930s and a steep decline in speed in the 1980s. We believe this is an artifact of our representation of word use as a function of time as there is a single time vector that influences all words. In the future, we will explore model variants to address this.

4.4 Automatic extraction of time periods

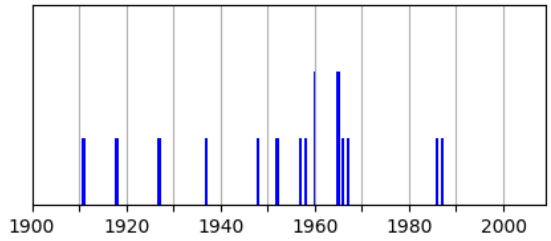


Figure 5: Distribution of time points where a node in h_1 is zero. We could interpret these points as barriers between time periods.

We can inspect h_1, the first layer in the time subnetwork, to gain further understanding of what our model is doing. We do this by analyzing the time points where a node in h_1 is zero.

As the activation function in h_1 is tanh, a node in h_1 switches from positive to negative (or vice versa) at the time points where it is zero. Thus, the time points where a node is zero should indicate barriers between time periods.

We visualize the time points where a node is zero in Figure 5. We see that we have a fairly even distribution of points until the 1940s, a large burst of points in the 1950s-1960s, and two points in the 1980s. Thus, there are many time periods before the 1940s (which may be caused by noisiness of the data in the first half of the century), a big transition between time periods in the 1950s-1960s, and a transition between time periods in the 1980s. Thus, these are time points that the model perceives as having increased semantic change.

However, there is a weakness to this analysis. Only 16% of the 100 nodes in h_1 are zero for time points between 1900 and 2009. Thus, a vast majority of nodes do not correspond to transitions between time periods.

5 Conclusion

Diachronic distributional models are a helpful tool in studying semantic shift. In this paper, we introduced our model of diachronic distributional semantics. Our model incorporates two hypotheses that better help the model capture how words change usage over time. The first hypothesis is that semantic change is gradual and the second hypothesis is that words can change usage due to common causes.

Additionally, we have developed a novel synthetic task to evaluate how accurately a model tracks the semantic shift of a word across time. This task directly measures semantic shift, is quantifiable, allows model comparison, and focuses on the trajectory of a word over time.

We have also used the fact that our model is differentiable to create a measure of the speed at which a word is changing. We then explored this measure's capabilities and limitations.

Acknowledgments

We would like to thank the University of Texas Natural Language Learning reading group as well as the reviewers for their helpful suggestions. This research was supported by the NSF grant IIS 1523637, and by a grant from the Morris Memorial Trust Fund of the New York Community Trust. We acknowledge the Texas Advanced Computing Center for providing grid resources that contributed to these results.

References

Robert Bamler and Stephan Mandt. 2017. Dynamic word embeddings. In *International Conference on Machine Learning*. pages 380–389.

David Bamman and Gregory Crane. 2011. Measuring historical word sense variation. In *Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries*. ACM, pages 1–10.

Oren Barkan. 2017. Bayesian neural word embedding. In *AAAI*. pages 3135–3143.

Marco Baroni and Alessandro Lenci. 2011. How we BLESSed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*. Association for Computational Linguistics, pages 1–10.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics, pages 136–145.

Paul Cook, Jey Han Lau, Diana McCarthy, and Timothy Baldwin. 2014. Novel word-sense identification. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. pages 1624–1635.

Paul Cook, Jey Han Lau, Michael Rundell, Diana McCarthy, and Timothy Baldwin. 2013. A lexicographic appraisal of an automatic approach for detecting new word senses. *Proceedings of eLex* pages 49–65.

Ashwini Deo. 2015. Diachronic semantics. *Annu. Rev. Linguist.* 1(1):179–197.

Haim Dubossarsky, Daphna Weinshall, and Eitan Grossman. 2017. Outta control: Laws of semantic change and inherent biases in word representation models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 1136–1145.

Daniel Francis. 1997. *National dreams: Myth, memory, and Canadian history*. Arsenal Pulp Press.

Lea Frermann and Mirella Lapata. 2016. A Bayesian Model of Diachronic Meaning Change. *Transactions of the Association for Computational Linguistics* 4:31–45.

William A Gale, Kenneth W Church, and David Yarowsky. 1992. Work on statistical methods for word sense disambiguation. In *Working Notes of the AAAI Fall Symposium on Probabilistic Approaches to Natural Language*. volume 54, page 60.

Kristina Gulordava and Marco Baroni. 2011. A distributional similarity approach to the detection of semantic change in the Google Books Ngram corpus. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*. Association for Computational Linguistics, pages 67–71.

William L Hamilton, Jure Leskovec, and Dan Jurafsky. 2016a. Cultural shift or linguistic drift? Comparing two computational measures of semantic change. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*. NIH Public Access, volume 2016, page 2116.

William L Hamilton, Jure Leskovec, and Dan Jurafsky. 2016b. Diachronic word embeddings reveal statistical laws of semantic change. *arXiv preprint arXiv:1605.09096* .

Douglas Harper. 2014. gay. In *Online Etymology Dictionary*. http://www.etymonline.com/word/gay.

Johannes Hellrich and Udo Hahn. 2016. Bad company-neighborhoods in neural embedding spaces considered harmful. In *COLING*. pages 2785–2796.

Johannes Hellrich and Udo Hahn. 2017. Exploring diachronic lexical semantics with JESEME. *Proceedings of ACL 2017, System Demonstrations* pages 31–36.

Adam Jatowt and Kevin Duh. 2014. A framework for analyzing semantic change of words across time. In *Proceedings of the 14th ACM/IEEE-CS Joint Conference on Digital Libraries*. IEEE Press, pages 229–238.

Yoon Kim, Yi-I Chiu, Kentaro Hanaki, Darshan Hegde, and Slav Petrov. 2014. Temporal analysis of language through neural language models. *arXiv preprint arXiv:1405.3515* .

Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2015. Statistically significant detection of linguistic change. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, pages 625–635.

Andrey Kutuzov, Erik Velldal, and Lilja Øvrelid. 2017a. Temporal dynamics of semantic relations in word embeddings: An application to predicting armed conflict participants. *arXiv preprint arXiv:1707.08660* .

Andrey Kutuzov, Erik Velldal, and Lilja Øvrelid. 2017b. Tracing armed conflicts with diachronic word embedding models. In *Proceedings of the Events and Stories in the News Workshop*. pages 31–36.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*. pages 2177–2185.

Yuri Lin, Jean-Baptiste Michel, Erez Lieberman Aiden, Jon Orwant, Will Brockman, and Slav Petrov. 2012. Syntactic annotations for the Google Books Ngram corpus. In *Proceedings of the ACL 2012 system demonstrations*. Association for Computational Linguistics, pages 169–174.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Eitan Adam Pechenick, Christopher M Danforth, and Peter Sheridan Dodds. 2015. Characterizing the Google Books corpus: Strong limits to inferences of socio-cultural and linguistic evolution. *PloS one* 10(10):e0137041.

Christian Rohrdantz, Annette Hautli, Thomas Mayer, Miriam Butt, Daniel A Keim, and Frans Plank. 2011. Towards tracking semantic change by visual analytics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*. Association for Computational Linguistics, pages 305–310.

Eyal Sagi, Stefan Kaufmann, and Brady Clark. 2009. Semantic density analysis: Comparing word meaning across time and phonetic space. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*. Association for Computational Linguistics, pages 104–111.

Eyal Sagi, Stefan Kaufmann, and Brady Clark. 2011. Tracing semantic change with latent semantic analysis. *Current methods in historical semantics* pages 161–183.

Hinrich Schütze. 1992. Context space. In *Working Notes of the AAAI Fall Symposium on Probabilistic Approaches to Natural Language*. pages 113–120.

Derry Tanti Wijaya and Reyyan Yeniterzi. 2011. Understanding semantic change of words over centuries. In *Proceedings of the 2011 international workshop on DETecting and Exploiting Cultural diversiTy on the social web*. ACM, pages 35–40.

Yang Xu and Charles Kemp. 2015. A computational evaluation of two laws of semantic change. In *CogSci*.

A Hyperparameters and preprocessing details

We used data from the English Fiction section of the Google Books ngram corpus (Lin et al., 2012). We use the English fiction specifically, because it is less unbalanced than the full English section and less influenced by technical texts (Pechenick et al., 2015). We only use the years 1900 to 2009 as there is limited data before 1900. Both the set of target words and the set of context words are the top 100,000 words by average frequency across the decades as generated by Hamilton et al. (2016b). We take a sampling approach to generating word vectors, so the corpus was converted into a list of (target word, context word, year, frequency) tuples. Frequency is the expected number of times the target word is in context of the context word that year. As the number of texts published since 1900 has increased five fold, we weigh the the frequencies so that the sums across each year are equal.

For every model, the representation of a word's use at time t is a 300 dimensional vector. For $SmallBinReg$, α_1 is set to 1000 and α_2 is set to 1. This choice of hyperparameters comes from Bamler and Mandt (2017). For $DiffTime$, every hidden layer is 100 dimensional, except for $embed_W(w)$ which is 300 dimensional.

We trained each method using random minibatching with 10,000 samples each iteration and 990 epochs total. For LargeBin, since our study spans 11 decades (1900-2009), the synchronic model for each decade is trained for 99 epochs. For SmallBinPreInit and SmallBinReg, since our study spans 110 years, the synchronic model for each year is trained for 9 epochs.

B BLESS class preprocessing

BLESS Class	Size
bird	10
building	7
clothing	10
fruit	7
furniture	8
ground_mammal	17
tool	12
tree	6
vehicle	6
weapon	7

Table 3: BLESS classes with the number of elements in each class after our preprocessing.

In this section, we discuss the BLESS preprocessing details. In the original dataset, there are 200 words categorized into 17 classes. However, we remove words that do not rank in the top 20,000 by frequency in any decade in our training data to ensure that the synthetic words do not lack context words at a given time. We then remove BLESS classes with less than 6 members to ensure that there are a sufficient number of words in each class. See Table 3 for the resulting list of BLESS classes and the number of members of each class.

Distributional Inclusion Vector Embedding for Unsupervised Hypernymy Detection

Haw-Shiuan Chang[1], **ZiYun Wang**[2], **Luke Vilnis**[1], **Andrew McCallum**[1]
[1]University of Massachusetts, Amherst, USA
[2]Tsinghua University, Beijing, China
hschang@cs.umass.edu, wang-zy14@mails.tsinghua.edu.cn,
{luke, mccallum}@cs.umass.edu

Abstract

Modeling hypernymy, such as poodle is-a dog, is an important generalization aid to many NLP tasks, such as entailment, coreference, relation extraction, and question answering. Supervised learning from labeled hypernym sources, such as WordNet, limits the coverage of these models, which can be addressed by learning hypernyms from unlabeled text. Existing unsupervised methods either do not scale to large vocabularies or yield unacceptably poor accuracy. This paper introduces *distributional inclusion vector embedding (DIVE)*, a simple-to-implement unsupervised method of hypernym discovery via per-word non-negative vector embeddings which preserve the inclusion property of word contexts in a low-dimensional and interpretable space. In experimental evaluations more comprehensive than any previous literature of which we are aware—evaluating on 11 datasets using multiple existing as well as newly proposed scoring functions—we find that our method provides up to double the precision of previous unsupervised embeddings, and the highest average performance, using a much more compact word representation, and yielding many new state-of-the-art results.

1 Introduction

Numerous applications benefit from compactly representing context distributions, which assign meaning to objects under the rubric of *distributional semantics*. In natural language processing, distributional semantics has long been used to assign meanings to words (that is, to *lexemes* in the dictionary, not individual instances of word tokens). The meaning of a word in the distributional sense is often taken to be the set of textual contexts (nearby tokens) in which that word appears, represented as a large sparse bag of words (SBOW). Without any supervision, Word2Vec (Mikolov et al., 2013), among other approaches based on matrix factorization (Levy et al., 2015a), successfully compress the SBOW into a much lower dimensional embedding space, increasing the scalability and applicability of the embeddings while preserving (or even improving) the correlation of geometric embedding similarities with human word similarity judgments.

While embedding models have achieved impressive results, context distributions capture more semantic information than just word similarity. The *distributional inclusion hypothesis* (DIH) (Weeds and Weir, 2003; Geffet and Dagan, 2005; Cimiano et al., 2005) posits that the context set of a word tends to be a subset of the contexts of its hypernyms. For a concrete example, most adjectives that can be applied to poodle can also be applied to dog, because dog is a hypernym of poodle (e.g. both can be obedient). However, the converse is not necessarily true — a dog can be straight-haired but a poodle cannot. Therefore, dog tends to have a broader context set than poodle. Many asymmetric scoring functions comparing SBOW features based on DIH have been developed for hypernymy detection (Weeds and Weir, 2003; Geffet and Dagan, 2005; Shwartz et al., 2017).

Hypernymy detection plays a key role in many challenging NLP tasks, such as textual entailment (Sammons et al., 2011), coreference (Ponzetto and Strube, 2006), relation extraction (Demeester et al., 2016) and question answering (Huang et al., 2008). Leveraging the variety of contexts and inclusion properties in context distributions can greatly increase the ability to discover taxonomic structure among words (Shwartz et al., 2017). The inability to preserve these features limits the semantic representation power and downstream applicability of some popular unsupervised learning approaches such as Word2Vec.

Several recently proposed methods aim to en-

Proceedings of NAACL-HLT 2018, pages 485–495
New Orleans, Louisiana, June 1 - 6, 2018.

code hypernym relations between words in dense embeddings, such as Gaussian embedding (Vilnis and McCallum, 2015; Athiwaratkun and Wilson, 2017), Boolean Distributional Semantic Model (Kruszewski et al., 2015), order embedding (Vendrov et al., 2016), H-feature detector (Roller and Erk, 2016), HyperVec (Nguyen et al., 2017), dual tensor (Glavaš and Ponzetto, 2017), Poincaré embedding (Nickel and Kiela, 2017), and LEAR (Vulić and Mrkšić, 2017). However, the methods focus on supervised or semi-supervised settings where a massive amount of hypernym annotations are available (Vendrov et al., 2016; Roller and Erk, 2016; Nguyen et al., 2017; Glavaš and Ponzetto, 2017; Vulić and Mrkšić, 2017), do not learn from raw text (Nickel and Kiela, 2017) or lack comprehensive experiments on the hypernym detection task (Vilnis and McCallum, 2015; Athiwaratkun and Wilson, 2017).

Recent studies (Levy et al., 2015b; Shwartz et al., 2017) have underscored the difficulty of generalizing supervised hypernymy annotations to unseen pairs — classifiers often effectively memorize prototypical hypernyms ('general' words) and ignore relations between words. These findings motivate us to develop more accurate and scalable unsupervised embeddings to detect hypernymy and propose several scoring functions to analyze the embeddings from different perspectives.

1.1 Contributions

- A novel unsupervised low-dimensional embedding method via performing non-negative matrix factorization (NMF) on a weighted PMI matrix, which can be efficiently optimized using modified skip-grams.
- Theoretical and qualitative analysis illustrate that the proposed embedding can intuitively and interpretably preserve inclusion relations among word contexts.
- Extensive experiments on 11 hypernym detection datasets demonstrate that the learned embeddings dominate previous low-dimensional unsupervised embedding approaches, achieving similar or better performance than SBOW, on both existing and newly proposed asymmetric scoring functions, while requiring much less memory and compute.

2 Method

The *distributional inclusion hypothesis* (DIH) suggests that the context set of a hypernym tends to contain the context set of its hyponyms. When representing a word as the counts of contextual co-occurrences, the count in every dimension of hypernym y tends to be larger than or equal to the corresponding count of its hyponym x:

$$x \preceq y \iff \forall c \in V,\ \#(x,c) \leq \#(y,c), \quad (1)$$

where $x \preceq y$ means y is a hypernym of x, V is the set of vocabulary, and $\#(x,c)$ indicates the number of times that word x and its context word c co-occur in a small window with size $|W|$ in the corpus of interest D. Notice that the concept of DIH could be applied to different context word representations. For example, Geffet and Dagan (2005) represent each word by the set of its co-occurred context words while discarding their counts. In this study, we define the inclusion property based on counts of context words in (1) because the counts are an effective and noise-robust feature for the hypernymy detection using only the context distribution of words (Clarke, 2009; Vulić et al., 2016; Shwartz et al., 2017).

Our goal is to produce lower-dimensional embeddings preserving the inclusion property that the embedding of hypernym y is larger than or equal to the embedding of its hyponym x in every dimension. Formally, the desired property can be written as

$$x \preceq y \iff \mathbf{x}[i] \leq \mathbf{y}[i]\ , \forall i \in \{1, ..., L\}, \quad (2)$$

where L is number of dimensions in the embedding space. We add additional non-negativity constraints, i.e. $\mathbf{x}[i] \geq 0, \mathbf{y}[i] \geq 0, \forall i$, in order to increase the interpretability of the embeddings (the reason will be explained later in this section).

This is a challenging task. In reality, there are a lot of noise and systematic biases that cause the violation of DIH in Equation (1) (i.e. $\#(x,c) > \#(y,c)$ for some neighboring word c), but the general trend can be discovered by processing thousands of neighboring words in SBOW together (Shwartz et al., 2017). After the compression, the same trend has to be estimated in a much smaller embedding space which discards most of the information in SBOW, so it is not surprising to see most of the unsupervised hypernymy detection studies focus on SBOW (Shwartz et al., 2017)

and the existing unsupervised embedding methods like Gaussian embedding have degraded accuracy (Vulić et al., 2016).

2.1 Inclusion Preserving Matrix Factorization

Popular methods of unsupervised word embedding are usually based on matrix factorization (Levy et al., 2015a). The approaches first compute a co-occurrence statistic between the wth word and the cth context word as the (w, c)th element of the matrix $M[w, c]$. Next, the matrix M is factorized such that $M[w, c] \approx \mathbf{w}^T\mathbf{c}$, where $\mathbf{w}$ is the low dimension embedding of wth word and $\mathbf{c}$ is the cth context embedding.

The statistic in $M[w, c]$ is usually related to pointwise mutual information (Levy et al., 2015a): $PMI(w, c) = \log(\frac{P(w,c)}{P(w)\cdot P(c)})$, where $P(w, c) = \frac{\#(w,c)}{|D|}$, $|D| = \sum_{w\in V}\sum_{c\in V} \#(w, c)$ is number of co-occurrence word pairs in the corpus, $P(w) = \frac{\#(w)}{|D|}$, $\#(w) = \sum_{c\in V} \#(w, c)$ is the frequency of the word w times the window size $|W|$, and similarly for $P(c)$. For example, $M[w, c]$ could be set as positive PMI (PPMI), $\max(PMI(w, c), 0)$, or shifted PMI, $PMI(w, c) - \log(k')$, which (Levy and Goldberg, 2014) demonstrate is connected to skip-grams with negative sampling (SGNS).

Intuitively, since $M[w, c] \approx \mathbf{w}^T\mathbf{c}$, larger embedding values of $\mathbf{w}$ at every dimension seems to imply larger $\mathbf{w}^T\mathbf{c}$, larger $M[w, c]$, larger $PMI(w, c)$, and thus larger co-occurrence count $\#(w, c)$. However, the derivation has two flaws: (1) $\mathbf{c}$ could contain negative values and (2) lower $\#(w, c)$ could still lead to larger $PMI(w, c)$ as long as the $\#(w)$ is small enough.

To preserve DIH, we propose a novel word embedding method, *distributional inclusion vector embedding (DIVE)*, which fixes the two flaws by performing non-negative factorization (NMF) (Lee and Seung, 2001) on the matrix M, where $M[w, c] =$

$$\log(\frac{P(w, c)}{P(w)\cdot P(c)}\cdot\frac{\#(w)}{k_I\cdot Z}) = \log(\frac{\#(w, c)|V|}{\#(c)k_I}), \tag{3}$$

where k_I is a constant which shifts PMI value like SGNS, $Z = \frac{|D|}{|V|}$ is the average word frequency, and $|V|$ is the vocabulary size. We call this weighting term $\frac{\#(w)}{Z}$ *inclusion shift*.

After applying the non-negativity constraint and inclusion shift, the inclusion property in DIVE (i.e. Equation (2)) implies that Equation (1) (DIH) holds if the matrix is reconstructed perfectly. The derivation is simple: If the embedding of hypernym $\mathbf{y}$ is greater than or equal to the embedding of its hyponym $\mathbf{x}$ in every dimension ($\mathbf{x}[i] \leq \mathbf{y}[i]\ , \forall i$), $\mathbf{x}^T\mathbf{c} \leq \mathbf{y}^T\mathbf{c}$ since context vector $\mathbf{c}$ is non-negative. Then, $M[x, c] \leq M[y, c]$ tends to be true because $\mathbf{w}^T\mathbf{c} \approx M[w, c]$. This leads to $\#(x, c) \leq \#(y, c)$ because $M[w, c] = \log(\frac{\#(w,c)|V|}{\#(c)k_I})$ and only $\#(w, c)$ changes with w.

2.2 Optimization

Due to its appealing scalability properties during training time (Levy et al., 2015a), we optimize our embedding based on the skip-gram with negative sampling (SGNS) (Mikolov et al., 2013). The objective function of SGNS is

$$l_{SGNS} = \sum_{w\in V}\sum_{c\in V} \#(w, c)\log\sigma(\mathbf{w}^T\mathbf{c}) + \sum_{w\in V} k'\sum_{c\in V}\#(w, c)\mathop{\mathbb{E}}_{c_N\sim P_D}[\log\sigma(-\mathbf{w}^T\mathbf{c_N})], \tag{4}$$

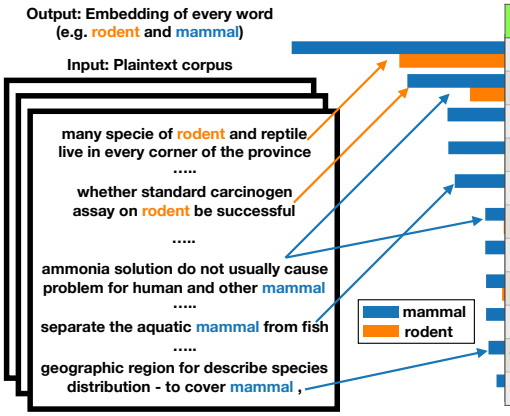
where $\mathbf{w} \in \mathbb{R}$, $\mathbf{c} \in \mathbb{R}$, $\mathbf{c_N} \in \mathbb{R}$, σ is the logistic sigmoid function, and k' is a constant hyper-parameter indicating the ratio between positive and negative samples.

Levy and Goldberg (2014) demonstrate SGNS is equivalent to factorizing a shifted PMI matrix M', where $M'[w, c] = \log(\frac{P(w,c)}{P(w)\cdot P(c)}\cdot\frac{1}{k'})$. By setting $k' = \frac{k_I\cdot Z}{\#(w)}$ and applying non-negativity constraints to the embeddings, DIVE can be optimized using the similar objective function:

$$l_{DIVE} = \sum_{w\in V}\sum_{c\in V} \#(w, c)\log\sigma(\mathbf{w}^T\mathbf{c}) + k_I\sum_{w\in V}\frac{Z}{\#(w)}\sum_{c\in V}\#(w, c)\mathop{\mathbb{E}}_{c_N\sim P_D}[\log\sigma(-\mathbf{w}^T\mathbf{c_N})], \tag{5}$$

where $\mathbf{w} \geq 0, \mathbf{c} \geq 0, \mathbf{c_N} \geq 0$, and k_I is a constant hyper-parameter. P_D is the distribution of negative samples, which we set to be the corpus word frequency distribution (not reducing the probability of drawing frequent words like SGNS) in this paper. Equation (5) is optimized by ADAM (Kingma and Ba, 2015), a variant of stochastic gradient descent (SGD). The non-negativity constraint is implemented by projection (Polyak, 1969) (i.e. clipping any embedding which crosses the zero boundary after an update).

The optimization process provides an alternative angle to explain how DIVE preserves DIH.



id	Top 1-5 words	Top 51-55 words
1	find, specie, species, animal, bird	hunt, terrestrial, lion, planet, shark
2	system, blood, vessel, artery, intestine	function, red, urinary, urine, tumor
3	head, leg, long, foot, hand	shoe, pack, food, short, right
4	may, cell, protein, gene, receptor	neuron, eukaryotic, immune, kinase, generally
5	sea, lake, river, area, water	terrain, southern, mediterranean, highland, shallow
6	cause, disease, effect, infection, increase	stress, problem, natural, earth, hazard
7	female, age, woman, male, household	spread, friend, son, city, infant
8	food, fruit, vegetable, meat, potato	fresh, flour, butter, leave, beverage
9	element, gas, atom, rock, carbon	light, dense, radioactive, composition, deposit
10	number, million, total, population, estimate	increase, less, capita, reach, male
11	industry, export, industrial, economy, company	centre, chemical, construction, fish, small

Figure 1: The embedding of the words rodent and mammal trained by the co-occurrence statistics of context words using DIVE. The index of dimensions is sorted by the embedding values of mammal and values smaller than 0.1 are neglected. The top 5 words (sorted by its embedding value of the dimension) tend to be more general or more representative on the topic than the top 51-105 words.

The gradients for the word embedding $\mathbf{w}$ is

$$\frac{dl_{DIVE}}{d\mathbf{w}} = \sum_{c \in V} \#(w,c)(1-\sigma(\mathbf{w}^T\mathbf{c}))\mathbf{c} - k_I \sum_{c_N \in V} \frac{\#(c_N)}{|V|}\sigma(\mathbf{w}^T\mathbf{c_N})\mathbf{c_N}. \quad (6)$$

Assume hyponym x and hypernym y satisfy DIH in Equation (1) and the embeddings $\mathbf{x}$ and $\mathbf{y}$ are the same at some point during the gradient ascent. At this point, the gradients coming from negative sampling (the second term) decrease the same amount of embedding values for both x and y. However, the embedding of hypernym $\mathbf{y}$ would get higher or equal positive gradients from the first term than $\mathbf{x}$ in every dimension because $\#(x,c) \leq \#(y,c)$. This means Equation (1) tends to imply Equation (2) because the hypernym has larger gradients everywhere in the embedding space.

Combining the analysis from the matrix factorization viewpoint, DIH in Equation (1) is approximately equivalent to the inclusion property in DIVE (i.e. Equation (2)).

2.3 PMI Filtering

For a frequent target word, there must be many neighboring words that incidentally appear near the target word without being semantically meaningful, especially when a large context window size is used. The unrelated context words cause noise in both the word vector and the context vector of DIVE. We address this issue by filtering out context words c for each target word w when the PMI of the co-occurring words is too small (i.e. $\log(\frac{P(w,c)}{P(w)\cdot P(c)}) < \log(k_f)$). That is, we set $\#(w,c) = 0$ in the objective function. This preprocessing step is similar to computing PPMI in SBOW (Bullinaria and Levy, 2007), where low PMI co-occurrences are removed from SBOW.

2.4 Interpretability

After applying the non-negativity constraint, we observe that each latent factor in the embedding is interpretable as previous findings suggest (Pauca et al., 2004; Murphy et al., 2012) (i.e. each dimension roughly corresponds to a topic). Furthermore, DIH suggests that a general word appears in more diverse contexts/topics. By preserving DIH using inclusion shift, the embedding of a general word (i.e. hypernym of many other words) tends to have larger values in these dimensions (topics). This gives rise to a natural and intuitive interpretation of our word embeddings: the word embeddings can be seen as unnormalized probability distributions over topics. In Figure 1, we visualize the unnormalized topical distribution of two words, rodent and mammal, as an example. Since rodent is a kind of mammal, the embedding (i.e. unnormalized topical distribution) of mammal includes the embedding of rodent when DIH holds. More examples are illustrated in our supplementary materials.

3 Unsupervised Embedding Comparison

In this section, we compare DIVE with other unsupervised hypernym detection methods. In this paper, unsupervised approaches refer to the methods that only train on plaintext corpus without using any hypernymy or lexicon annotation.

Dataset	BLESS	EVALution	LenciBenotto	Weeds	Medical	LEDS
Random	5.3	26.6	41.2	51.4	8.5	50.5
Word2Vec + C	9.2	25.4	40.8	51.6	11.2	71.8
GE + C	10.5	26.7	43.3	52.0	14.9	69.7
GE + KL	7.6	29.6	45.1	51.3	15.7	64.6 (80[3])
DIVE + C·ΔS	**16.3**	**33.0**	**50.4**	**65.5**	**19.2**	**83.5**

Dataset	TM14	Kotlerman 2010	HypeNet	WordNet	**Avg (10 datasets)**	HyperLex
Random	52.0	30.8	24.5	55.2	23.2	0
Word2Vec + C	52.1	**39.5**	20.7	**63.0**	25.3	16.3
GE + C	53.9	36.0	21.6	58.2	26.1	16.4
GE + KL	52.0	39.4	23.7	54.4	25.9	9.6 (20.6[3])
DIVE + C·ΔS	**57.2**	36.6	**32.0**	60.9	**32.7**	**32.8**

Table 1: Comparison with other unsupervised embedding methods. The scores are AP@all (%) for the first 10 datasets and Spearman ρ (%) for HyperLex. Avg (10 datasets) shows the micro-average AP of all datasets except HyperLex. Word2Vec+C scores word pairs using cosine similarity on skip-grams. GE+C and GE+KL compute cosine similarity and negative KL divergence on Gaussian embedding, respectively.

3.1 Experiment Setup

The embeddings are tested on 11 datasets. The first 4 datasets come from the recent review of Shwartz et al. (2017)[1]: BLESS (Baroni and Lenci, 2011), EVALution (Santus et al., 2015), Lenci/Benotto (Benotto, 2015), and Weeds (Weeds et al., 2014). The next 4 datasets are downloaded from the code repository of the H-feature detector (Roller and Erk, 2016)[2]: Medical (i.e., Levy 2014) (Levy et al., 2014), LEDS (also referred to as ENTAILMENT or Baroni 2012) (Baroni et al., 2012), TM14 (i.e., Turney 2014) (Turney and Mohammad, 2015), and Kotlerman 2010 (Kotlerman et al., 2010). In addition, the performance on the test set of HypeNet (Shwartz et al., 2016) (using the random train/test split), the test set of WordNet (Vendrov et al., 2016), and all pairs in HyperLex (Vulić et al., 2016) are also evaluated.

The F1 and accuracy measurements are sometimes very similar even though the quality of prediction varies, so we adopted average precision, AP@all (Zhu, 2004) (equivalent to the area under the precision-recall curve when the constant interpolation is used), as the main evaluation metric. The HyperLex dataset has a continuous score on each candidate word pair, so we adopt Spearman rank coefficient ρ (Fieller et al., 1957) as suggested by the review study of Vulić et al. (2016). Any OOV (out-of-vocabulary) word encountered in the testing data is pushed to the bottom of the prediction list (effectively assuming the word pair does not have hypernym relation).

[1] https://github.com/vered1986/UnsupervisedHypernymy

[2] https://github.com/stephenroller/emnlp2016/

We trained all methods on the first 51.2 million tokens of WaCkypedia corpus (Baroni et al., 2009) because DIH holds more often in this subset (i.e. SBOW works better) compared with that in the whole WaCkypedia corpus. The window size $|W|$ of DIVE and Gaussian embedding are set as 20 (left 10 words and right 10 words). The number of embedding dimensions in DIVE L is set to be 100. The other hyper-parameters of DIVE and Gaussian embedding are determined by the training set of HypeNet. Other experimental details are described in our supplementary materials.

3.2 Results

If a pair of words has hypernym relation, the words tend to be similar (sharing some context words) and the hypernym should be more general than the hyponym. Section 2.4 has shown that the embedding could be viewed as an unnormalized topic distribution of its context, so the embedding of hypernym should be similar to the embedding of its hyponym but having larger magnitude. As in HyperVec (Nguyen et al., 2017), we score the hypernym candidates by multiplying two factors corresponding to these properties. The C·ΔS (i.e. the cosine similarity multiply the difference of summation) scoring function is defined as

$$C \cdot \Delta S(\mathbf{w}_q \rightarrow \mathbf{w}_p) = \frac{\mathbf{w}_q^T \mathbf{w}_p}{||\mathbf{w}_q||_2 \cdot ||\mathbf{w}_p||_2} \cdot (\|\mathbf{w}_p\|_1 - \|\mathbf{w}_q\|_1), \quad (7)$$

where $\mathbf{w}_p$ is the embedding of hypernym and $\mathbf{w}_q$ is the embedding of hyponym.

As far as we know, Gaussian embedding (GE) (Vilnis and McCallum, 2015) is the state-of-the-art unsupervised embedding method which can capture the asymmetric relations between a hypernym and its hyponyms. Gaussian embedding

encodes the context distribution of each word as a multivariate Gaussian distribution, where the embeddings of hypernyms tend to have higher variance and overlap with the embedding of their hyponyms. In Table 1, we compare DIVE with Gaussian embedding[3] using the code implemented by Athiwaratkun and Wilson (2017)[4] and with word cosine similarity using skip-grams. The performances of random scores are also presented for reference. As we can see, DIVE is usually significantly better than other unsupervised embedding.

4 SBOW Comparison

Unlike Word2Vec, which only tries to preserve the similarity signal, the goals of DIVE cover preserving the capability of measuring not only the similarity but also whether one context distribution includes the other (inclusion signal) or being more general than the other (generality signal).

In this experiment, we perform a comprehensive comparison between SBOW and DIVE using multiple scoring functions to detect the hypernym relation between words based on different types of signal. The window size $|W|$ of SBOW is also set as 20, and experiment setups are the same as that described in Section 3.1. Notice that the comparison is inherently unfair because most of the information would be lost during the aggressive compression process of DIVE, and we would like to evaluate how well DIVE can preserve signals of interest using the number of dimensions which is several orders of magnitude less than that of SBOW.

4.1 Unsupervised Scoring Functions

After trying many existing and newly proposed functions which score a pair of words to detect hypernym relation between them, we find that good scoring functions for SBOW are also good scoring functions for DIVE. Thus, in addition to C·ΔS used in Section 3.2, we also present 4 other best performing or representative scoring functions in the experiment (see our supplementary materials for more details):

- **Inclusion**: CDE (Clarke, 2009) computes the summation of element-wise minimum over the magnitude of hyponym embedding (i.e. $\frac{||\min(\mathbf{w}_p, \mathbf{w}_q)||_1}{||\mathbf{w}_q||_1}$). CDE measures the degree of violation of equation (1). Equation (1) holds if and only if CDE is 1. Due to noise in SBOW, CDE is rarely exactly 1, but hypernym pairs usually have higher CDE. Despite its effectiveness, the good performance could mostly come from the magnitude of embeddings/features instead of inclusion properties among context distributions. To measure the inclusion properties between context distributions $\mathbf{d}_p$ and $\mathbf{d}_q$ ($\mathbf{w}_p$ and $\mathbf{w}_q$ after normalization, respectively), we use negative asymmetric L1 distance ($-AL_1$)[5] as one of our scoring function, where

$$AL_1 = \min_a \sum_c w_0 \cdot \max(a\mathbf{d}_q[c] - \mathbf{d}_p[c], 0) + \max(\mathbf{d}_p[c] - a\mathbf{d}_q[c], 0), \quad (8)$$

  and w_0 is a constant hyper-parameter.

- **Generality**: When the inclusion property in (2) holds, $||\mathbf{y}||_1 = \sum_i \mathbf{y}[i] \geq \sum_i \mathbf{x}[i] = ||\mathbf{x}||_1$. Thus, we use summation difference ($||\mathbf{w}_p||_1 - ||\mathbf{w}_q||_1$) as our score to measure generality signal (ΔS).

- **Similarity plus generality**: Computing cosine similarity on skip-grams (i.e. Word2Vec + C in Table 1) is a popular way to measure the similarity of two words, so we multiply the Word2Vec similarity with summation difference of DIVE or SBOW (W·ΔS) as an alternative of C·ΔS.

4.2 Baselines

- SBOW Freq: A word is represented by the frequency of its neighboring words. Applying PMI filter (set context feature to be 0 if its value is lower than $\log(k_f)$) to SBOW Freq only makes its performances closer to (but still much worse than) SBOW PPMI, so we omit the baseline.

- SBOW PPMI: SBOW which uses PPMI of its neighboring words as the features (Bullinaria and Levy, 2007). Applying PMI filter to SBOW PPMI usually makes the performances worse, especially when k_f is large. Similarly, a constant $\log(k')$ shifting to SBOW PPMI (i.e. $\max(PMI - \log(k'), 0)$) is not helpful, so we set both k_f and k' to be 1.

[3] Note that higher AP is reported for some models in previous literature: 80 (Vilnis and McCallum, 2015) in LEDS, 74.2 (Athiwaratkun and Wilson, 2017) in LEDS, and 20.6 (Vulić et al., 2016) in HyperLex. The difference could be caused by different train/test setup (e.g. How the hyperparameters are tuned, different training corpus, etc.). However, DIVE beats even these results.

[4] https://github.com/benathi/word2gm

[5] The meaning and efficient implementation of AL_1 are illustrated in our supplementary materials

AP@all (%)		BLESS					EVALution					Lenci/Benotto				
		CDE	AL_1	ΔS	W·ΔS	C·ΔS	CDE	AL_1	ΔS	W·ΔS	C·ΔS	CDE	AL_1	ΔS	W·ΔS	C·ΔS
SBOW	Freq	6.3	7.3	5.6	11.0	5.9	35.3	**32.6**	36.2	**33.0**	36.3	51.8	**47.6**	51.0	51.8	51.1
	PPMI	**13.6**	5.1	5.6	17.2	15.3	30.4	27.7	34.1	31.9	34.3	47.2	39.7	50.8	51.1	**52.0**
	PPMI w/ IS	6.2	5.0	5.5	12.4	5.8	**36.0**	27.5	**36.3**	32.9	**36.4**	**52.0**	43.1	50.9	**51.9**	50.7
	All wiki	12.1	5.2	6.9	12.5	13.4	28.5	27.1	30.3	29.9	31.0	47.1	39.9	48.5	48.7	51.1
DIVE	Full	9.3	**7.6**	6.0	**18.6**	**16.3**	30.0	27.5	34.9	32.3	33.0	46.7	43.2	**51.3**	51.5	50.4
	w/o PMI	7.8	6.9	5.6	16.7	7.1	32.8	32.2	35.7	32.5	35.4	47.6	44.9	50.9	51.6	49.7
	w/o IS	9.0	6.2	**7.3**	6.2	7.3	24.3	25.0	22.9	23.5	23.9	38.8	38.1	38.2	38.2	38.4
Kmean (Freq NMF)		6.5	7.3	5.6	10.9	5.8	33.7	27.2	36.2	**33.0**	36.2	49.6	42.5	51.0	51.8	51.2

AP@all (%)		Weeds					Micro Average (4 datasets)					Medical				
		CDE	AL_1	ΔS	W·ΔS	C·ΔS	CDE	AL_1	ΔS	W·ΔS	C·ΔS	CDE	AL_1	ΔS	W·ΔS	C·ΔS
SBOW	Freq	**69.5**	**58.0**	68.8	68.2	68.4	23.1	**21.8**	**22.9**	25.0	23.0	19.4	**19.2**	**14.1**	18.4	15.3
	PPMI	61.0	50.3	**70.3**	**69.2**	69.3	**24.7**	17.9	22.3	28.1	**27.8**	**23.4**	8.7	13.2	20.1	**24.4**
	PPMI w/ IS	67.6	52.2	69.4	68.7	67.7	23.2	18.2	**22.9**	25.8	22.9	22.8	10.6	13.7	18.6	17.0
	All wiki	61.3	48.6	70.0	68.5	**70.4**	23.4	17.7	21.7	24.6	25.8	22.3	8.9	12.2	17.6	21.1
DIVE	Full	59.2	55.0	69.7	68.6	65.5	22.1	19.8	22.8	**28.9**	27.6	11.7	9.3	13.7	**21.4**	19.2
	w/o PMI	60.4	56.4	69.3	68.6	64.8	22.2	21.0	22.7	28.0	23.1	10.7	8.4	13.3	19.8	16.2
	w/o IS	49.2	47.3	45.1	45.1	44.9	18.9	17.3	17.2	16.8	17.5	10.9	9.8	7.4	7.6	7.7
Kmean (Freq NMF)		69.4	51.1	68.8	68.2	68.9	22.5	19.3	**22.9**	24.9	23.0	12.6	10.9	14.0	18.1	14.6

AP@all (%)		LEDS					TM14					Kotlerman 2010				
		CDE	AL_1	ΔS	W·ΔS	C·ΔS	CDE	AL_1	ΔS	W·ΔS	C·ΔS	CDE	AL_1	ΔS	W·ΔS	C·ΔS
SBOW	Freq	82.7	70.4	70.7	83.3	73.3	55.6	53.2	54.9	55.7	55.0	35.9	**40.5**	**34.5**	37.0	35.4
	PPMI	**84.4**	50.2	72.2	**86.5**	**84.5**	56.2	52.3	54.4	57.0	**57.6**	**39.1**	30.9	33.0	37.0	36.3
	PPMI w/ IS	81.6	54.5	71.0	84.7	73.1	**57.1**	51.5	55.1	56.2	55.4	37.4	31.0	34.4	37.8	35.9
	All wiki	83.1	49.7	67.9	82.9	81.4	54.7	50.5	52.6	55.1	54.9	38.5	31.2	32.2	35.4	35.3
DIVE	Full	83.3	74.7	**72.7**	86.4	83.5	55.3	52.6	**55.2**	**57.3**	57.2	35.3	31.6	33.6	37.4	36.6
	w/o PMI	79.3	**74.8**	72.0	85.5	78.7	54.7	**53.9**	54.9	56.5	55.4	35.4	38.9	33.8	**37.8**	**36.7**
	w/o IS	64.6	55.4	43.2	44.3	46.1	51.9	51.2	50.4	52.0	51.8	32.9	33.4	28.1	30.2	29.7
Kmean (Freq NMF)		80.3	64.5	70.7	83.0	73.0	54.8	49.0	54.8	55.6	54.8	32.1	37.0	34.5	36.9	34.8

AP@all (%)		HypeNet					WordNet					**Micro Average (10 datasets)**				
		CDE	AL_1	ΔS	W·ΔS	C·ΔS	CDE	AL_1	ΔS	W·ΔS	C·ΔS	CDE	AL_1	ΔS	W·ΔS	C·ΔS
SBOW	Freq	37.5	**28.3**	46.9	**35.9**	43.4	56.6	55.2	55.5	56.2	55.6	31.1	**28.2**	31.5	31.6	31.2
	PPMI	23.8	24.0	47.0	32.5	33.1	57.7	53.9	55.6	56.8	57.2	30.1	23.0	31.1	32.9	**33.5**
	PPMI w/ IS	**38.5**	26.7	47.2	35.5	37.6	57.0	54.1	55.7	56.6	55.7	**31.8**	24.1	31.5	32.1	30.3
	All wiki	23.0	24.5	40.5	30.5	29.7	57.4	53.1	56.0	56.4	57.3	29.0	23.1	29.2	30.2	31.1
DIVE	Full	25.3	24.2	**49.3**	33.6	32.0	60.2	58.9	**58.4**	**61.1**	**60.9**	27.6	25.3	**32.1**	**34.1**	32.7
	w/o PMI	31.3	27.0	46.9	33.8	34.0	59.2	60.1	58.2	**61.1**	59.1	28.5	26.7	31.5	33.4	30.1
	w/o IS	20.1	21.7	20.3	21.8	22.0	**61.0**	56.3	51.3	55.7	54.7	22.3	20.7	19.1	19.6	19.9
Kmean (Freq NMF)		33.7	22.0	46.0	35.6	**45.2**	58.4	**60.2**	57.7	60.1	57.9	29.1	24.7	31.5	31.8	31.5

Table 2: AP@all (%) of 10 datasets. The box at lower right corner compares the micro average AP across all 10 datasets. Numbers in different rows come from different feature or embedding spaces. Numbers in different columns come from different datasets and unsupervised scoring functions. We also present the micro average AP across the first 4 datasets (BLESS, EVALution, Lenci/Benotto and Weeds), which are used as a benchmark for unsupervised hypernym detection (Shwartz et al., 2017). IS refers to inclusion shift on the shifted PMI matrix.

Spearman ρ (%)		HyperLex				
		CDE	AL_1	ΔS	W·ΔS	C·ΔS
SBOW	Freq	31.7	19.6	27.6	29.6	27.3
	PPMI	28.1	-2.3	**31.8**	**34.3**	**34.5**
	PPMI w/ IS	**32.4**	2.1	28.5	31.0	27.4
	All wiki	25.3	-2.2	28.0	30.5	31.0
DIVE	Full	28.9	18.7	31.2	33.3	32.8
	w/o PMI	29.2	**22.2**	29.5	31.9	29.2
	w/o IS	11.5	-0.9	-6.2	-10.0	-11.6
Kmean (Freq NMF)		30.6	3.3	27.5	29.5	27.6

Table 3: Spearman ρ (%) in HyperLex.

SBOW Freq	SBOW PPMI	DIVE
5799	3808	**20**

Table 4: The average number of non-zero dimensions across all testing words in 10 datasets.

- SBOW PPMI w/ IS (with additional inclusion shift): The matrix reconstructed by DIVE when $k_I = 1$. Specifically, $\mathbf{w}[c] = \max(\log(\frac{P(w,c)}{P(w)*P(c)*\frac{Z}{\#(w)}}), 0)$.
- SBOW all wiki: SBOW using PPMI features trained on the whole WaCkypedia.
- DIVE without the PMI filter (DIVE w/o PMI)
- NMF on shifted PMI: Non-negative matrix factorization (NMF) on the shifted PMI without inclusion shift for DIVE (DIVE w/o IS). This is the same as applying the non-negative constraint on the skip-gram model.

- K-means (Freq NMF): The method first uses Mini-batch k-means (Sculley, 2010) to cluster words in skip-gram embedding space into 100 topics, and hashes each frequency count in SBOW into the corresponding topic. If running k-means on skip-grams is viewed as an approximation of clustering the SBOW context vectors, the method can be viewed as a kind of NMF (Ding et al., 2005).

DIVE performs non-negative matrix factorization on PMI matrix after applying inclusion shift and PMI filtering. To demonstrate the effectiveness of each step, we show the performances of DIVE after removing PMI filtering (DIVE w/o PMI), removing inclusion shift (DIVE w/o IS), and removing matrix factorization (SBOW PPMI w/ IS, SBOW PPMI, and SBOW all wiki). The methods based on frequency matrix are also tested (SBOW Freq and Freq NMF).

4.3 Results and Discussions

In Table 2, we first confirm the finding of the previous review study of Shwartz et al. (2017): there is no single hypernymy scoring function which always outperforms others. One of the main reasons is that different datasets collect negative samples differently. For example, if negative samples come from random word pairs (e.g. WordNet dataset), a symmetric similarity measure is a good scoring function. On the other hand, negative samples come from related or similar words in HypeNet, EVALution, Lenci/Benotto, and Weeds, so only estimating generality difference leads to the best (or close to the best) performance. The negative samples in many datasets are composed of both random samples and similar words (such as BLESS), so the combination of similarity and generality difference yields the most stable results.

DIVE performs similar or better on most of the scoring functions compared with SBOW consistently across all datasets in Table 2 and Table 3, while using many fewer dimensions (see Table 4). This leads to 2-3 order of magnitude savings on both memory consumption and testing time. Furthermore, the low dimensional embedding makes the computational complexity independent of the vocabulary size, which drastically boosts the scalability of unsupervised hypernym detection especially with the help of GPU. It is surprising that we can achieve such aggressive compression while preserving the similarity, generality, and inclusion signal in various datasets with different types of negative samples. Its results on C·ΔS and W·ΔS outperform SBOW Freq. Meanwhile, its results on AL_1 outperform SBOW PPMI. The fact that W·ΔS or C·ΔS usually outperform generality functions suggests that only memorizing general words is not sufficient. The best average performance on 4 and 10 datasets are both produced by W·ΔS on DIVE.

SBOW PPMI improves the W·ΔS and C·ΔS from SBOW Freq but sacrifices AP on the inclusion functions. It generally hurts performance to directly include inclusion shift in PPMI (PPMI w/ IS) or compute SBOW PPMI on the whole WaCkypedia (all wiki) instead of the first 51.2 million tokens. The similar trend can also be seen in Table 3. Note that AL_1 completely fails in the HyperLex dataset using SBOW PPMI, which suggests that PPMI might not necessarily preserve the distributional inclusion property, even though it can have good performance on scoring functions combining similarity and generality signals.

Removing the PMI filter from DIVE slightly drops the overall precision while removing inclusion shift on shifted PMI (w/o IS) leads to poor performances. K-means (Freq NMF) produces similar AP compared with SBOW Freq but has worse AL_1 scores. Its best AP scores on different datasets are also significantly worse than the best AP of DIVE. This means that only making Word2Vec (skip-grams) non-negative or naively accumulating topic distribution in contexts cannot lead to satisfactory embeddings.

5 Related Work

Most previous unsupervised approaches focus on designing better hypernymy scoring functions for sparse bag of word (SBOW) features. They are well summarized in the recent study (Shwartz et al., 2017). Shwartz et al. (2017) also evaluate the influence of different contexts, such as changing the window size of contexts or incorporating dependency parsing information, but neglect scalability issues inherent to SBOW methods.

A notable exception is the Gaussian embedding model (Vilnis and McCallum, 2015), which represents each word as a Gaussian distribution. However, since a Gaussian distribution is normalized, it is difficult to retain frequency information during the embedding process, and experiments on HyperLex (Vulić et al., 2016) demonstrate that a sim-

ple baseline only relying on word frequency can achieve good results. Follow-up work models contexts by a mixture of Gaussians (Athiwaratkun and Wilson, 2017) relaxing the unimodality assumption but achieves little improvement on hypernym detection tasks.

Kiela et al. (2015) show that images retrieved by a search engine can be a useful source of information to determine the generality of lexicons, but the resources (e.g. pre-trained image classifier for the words of interest) might not be available in many domains.

Order embedding (Vendrov et al., 2016) is a supervised approach to encode many annotated hypernym pairs (e.g. all of the whole WordNet (Miller, 1995)) into a compact embedding space, where the embedding of a hypernym should be smaller than the embedding of its hyponym in every dimension. Our method learns embedding from raw text, where a hypernym embedding should be larger than the embedding of its hyponym in every dimension. Thus, DIVE can be viewed as an unsupervised and reversed form of order embedding.

Non-negative matrix factorization (NMF) has a long history in NLP, for example in the construction of topic models (Pauca et al., 2004). Non-negative sparse embedding (NNSE) (Murphy et al., 2012) and Faruqui et al. (2015) indicate that non-negativity can make embeddings more interpretable and improve word similarity evaluations. The sparse NMF is also shown to be effective in cross-lingual lexical entailment tasks but does not necessarily improve monolingual hypernymy detection (Vyas and Carpuat, 2016). In our study, we show that performing NMF on PMI matrix with inclusion shift can preserve DIH in SBOW, and the comprehensive experimental analysis demonstrates its state-of-the-art performances on unsupervised hypernymy detection.

6 Conclusions

Although large SBOW vectors consistently show the best all-around performance in unsupervised hypernym detection, it is challenging to compress them into a compact representation which preserves inclusion, generality, and similarity signals for this task. Our experiments suggest that the existing approaches and simple baselines such as Gaussian embedding, accumulating K-mean clusters, and non-negative skip-grams do not lead to satisfactory performance.

To achieve this goal, we propose an interpretable and scalable embedding method called *distributional inclusion vector embedding (DIVE)* by performing non-negative matrix factorization (NMF) on a weighted PMI matrix. We demonstrate that scoring functions which measure inclusion and generality properties in SBOW can also be applied to DIVE to detect hypernymy, and DIVE performs the best on average, slightly better than SBOW while using many fewer dimensions.

Our experiments also indicate that unsupervised scoring functions which combine similarity and generality measurements work the best in general, but no one scoring function dominates across all datasets. A combination of unsupervised DIVE with the proposed scoring functions produces new state-of-the-art performances on many datasets in the unsupervised regime.

7 Acknowledgement

This work was supported in part by the Center for Data Science and the Center for Intelligent Information Retrieval, in part by DARPA under agreement number FA8750-13-2-0020, in part by Defense Advanced Research Agency (DARPA) contract number HR0011-15-2-0036, in part by the National Science Foundation (NSF) grant numbers DMR-1534431 and IIS-1514053 and in part by the Chan Zuckerberg Initiative under the project Scientific Knowledge Base Construction. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, or the U.S. Government, or the other sponsors.

References

Ben Athiwaratkun and Andrew Gordon Wilson. 2017. Multimodal word distributions. In *ACL*.

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. In *EACL*.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The WaCky wide web: a collection of very large linguistically processed

web-crawled corpora. *Language resources and evaluation* 43(3):209–226.

Marco Baroni and Alessandro Lenci. 2011. How we BLESSed distributional semantic evaluation. In *Workshop on GEometrical Models of Natural Language Semantics (GEMS)*.

Giulia Benotto. 2015. Distributional models for semantic relations: A study on hyponymy and antonymy. *PhD Thesis, University of Pisa* .

John A Bullinaria and Joseph P Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior research methods* 39(3):510–526.

Philipp Cimiano, Andreas Hotho, and Steffen Staab. 2005. Learning concept hierarchies from text corpora using formal concept analysis. *J. Artif. Intell. Res.(JAIR)* 24(1):305–339.

Daoud Clarke. 2009. Context-theoretic semantics for natural language: an overview. In *workshop on geometrical models of natural language semantics*. pages 112–119.

Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. 2016. Lifted rule injection for relation embeddings. In *EMNLP*.

Chris Ding, Xiaofeng He, and Horst D Simon. 2005. On the equivalence of nonnegative matrix factorization and spectral clustering. In *ICDM*.

Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah Smith. 2015. Sparse overcomplete word vector representations. In *ACL*.

Edgar C Fieller, Herman O Hartley, and Egon S Pearson. 1957. Tests for rank correlation coefficients. i. *Biometrika* .

Maayan Geffet and Ido Dagan. 2005. The distributional inclusion hypotheses and lexical entailment. In *ACL*.

Goran Glavaš and Simone Paolo Ponzetto. 2017. Dual tensor model for detecting asymmetric lexico-semantic relations. In *EMNLP*.

Zhiheng Huang, Marcus Thint, and Zengchang Qin. 2008. Question classification using head words and their hypernyms. In *EMNLP*.

Douwe Kiela, Laura Rimell, Ivan Vulic, and Stephen Clark. 2015. Exploiting image generality for lexical entailment detection. In *ACL*.

Diederik Kingma and Jimmy Ba. 2015. ADAM: A method for stochastic optimization. In *ICLR*.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering* 16(4):359–389.

Germán Kruszewski, Denis Paperno, and Marco Baroni. 2015. Deriving boolean structures from distributional vectors. *TACL* 3:375–388. https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/616.

Daniel D Lee and H Sebastian Seung. 2001. Algorithms for non-negative matrix factorization. In *NIPS*.

Omer Levy, Ido Dagan, and Jacob Goldberger. 2014. Focused entailment graphs for open IE propositions. In *CoNLL*.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *NIPS*.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015a. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics* 3:211–225.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015b. Do supervised distributional methods really learn lexical inference relations? In *NAACL-HTL*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

George A. Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

Brian Murphy, Partha Talukdar, and Tom Mitchell. 2012. Learning effective and interpretable semantic models using non-negative sparse embedding. *COLING* pages 1933–1950.

Kim Anh Nguyen, Maximilian Köper, Sabine Schulte im Walde, and Ngoc Thang Vu. 2017. Hierarchical embeddings for hypernymy detection and directionality. In *EMNLP*.

Maximilian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations. In *NIPS*.

V. Paul Pauca, Farial Shahnaz, Michael W Berry, and Robert J. Plemmons. 2004. Text mining using non-negative matrix factorizations. In *ICDM*.

Boris Teodorovich Polyak. 1969. Minimization of unsmooth functionals. *USSR Computational Mathematics and Mathematical Physics* .

Simone Paolo Ponzetto and Michael Strube. 2006. Exploiting semantic role labeling, wordnet and wikipedia for coreference resolution. In *ACL*.

Stephen Roller and Katrin Erk. 2016. Relations such as hypernymy: Identifying and exploiting hearst patterns in distributional vectors for lexical entailment. In *EMNLP*.

Mark Sammons, V Vydiswaran, and Dan Roth. 2011. Recognizing textual entailment. *Multilingual Natural Language Applications: From Theory to Practice. Prentice Hall, Jun* .

Enrico Santus, Frances Yung, Alessandro Lenci, and Chu-Ren Huang. 2015. EVALution 1.0: an evolving semantic dataset for training and evaluation of distributional semantic models. In *Workshop on Linked Data in Linguistics (LDL)*.

David Sculley. 2010. Web-scale k-means clustering. In *WWW*.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In *ACL*.

Vered Shwartz, Enrico Santus, and Dominik Schlechtweg. 2017. Hypernyms under siege: Linguistically-motivated artillery for hypernymy detection. In *EACL*.

Peter D Turney and Saif M Mohammad. 2015. Experiments with three approaches to recognizing lexical entailment. *Natural Language Engineering* 21(3):437–476.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2016. Order-embeddings of images and language. In *ICLR*.

Luke Vilnis and Andrew McCallum. 2015. Word representations via gaussian embedding. In *ICLR*.

Ivan Vulić, Daniela Gerz, Douwe Kiela, Felix Hill, and Anna Korhonen. 2016. Hyperlex: A large-scale evaluation of graded lexical entailment. *arXiv preprint arXiv:1608.02117* .

Ivan Vulić and Nikola Mrkšić. 2017. Specialising word vectors for lexical entailment. *arXiv preprint arXiv:1710.06371* .

Yogarshi Vyas and Marine Carpuat. 2016. Sparse bilingual word representations for cross-lingual lexical entailment. In *HLT-NAACL*.

Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller. 2014. Learning to distinguish hypernyms and co-hyponyms. In *COLING*.

Julie Weeds and David Weir. 2003. A general framework for distributional similarity. In *EMNLP*.

Mu Zhu. 2004. Recall, precision and average precision. *Department of Statistics and Actuarial Science, University of Waterloo, Waterloo* 2:30.

Mining Possessions: Existence, Type and Temporal Anchors

Dhivya Chinnappa and **Eduardo Blanco**
Human Intelligence and Language Technologies Lab
University of North Texas
Denton, TX, 76203
dhivyainfantchinnappa@my.unt.edu, eduardo.blanco@unt.edu

Abstract

This paper presents a corpus and experiments to mine possession relations from text. Specifically, we target alienable and control possessions, and assign temporal anchors indicating when the possession holds between possessor and possessee. We present new annotations for this task, and experimental results using both traditional classifiers and neural networks. Results show that the three subtasks (predicting possession existence, possession type and temporal anchors) can be automated.

1 Introduction

Every language has a way of expressing possessive relationships (Aikhenvald and Dixon, 2012). Possession is an asymmetric semantic relation between two entities, where one entity (the possessee) *belongs* to the other entity (the possessor) (Stassen, 2009). When it comes to defining possession, *belongs* includes a wide range of relationships, including (hereafter, we use x to refer to the possessor, and y to refer to the possessee) kinship (e.g., *[my]$_x$ oldest [son]$_y$*), part-whole (e.g., *the [car]$_x$'s [dashboard]$_y$*), physical and temporary possession (e.g., *[I]$_x$ have John's [book]$_y$*), possession of something intangible (e.g., *[John]$_x$ got the [flu]$_y$ last year*) and proximity (e.g., *The [shelf]$_x$ has a [glass sculpture]$_y$*).

Possession relations can be divided into alienable (also referred to as acquired, transferable, non-intimate, etc.) and inalienable (also referred to as inherent, inseparable, intimate, etc.). Possessees that can be separated from their possessors are alienable, and possessees that cannot normally be separated from their possessors are inalienable (Heine, 1997). For example, *[John]$_x$'s [condo]$_y$* is alienable, and *[John]$_x$'s [arm]$_y$* is inalienable (some previous works would call the latter a part-whole relation instead). Tham (2004) defines control possession as a relation in which the possessor has temporary control of the possessee, but does not necessarily alienably possess it (e.g., *[John]$_x$ borrowed the [car]$_y$ for the weekend*). Following the aforecited works, possession goes beyond ownership of property.

Possession relations can be expressed in a wide variety of syntactic constructions, including noun phrases (e.g., *[John]$_x$'s [car]$_y$*) and clauses (e.g., *[John]$_x$ bought a [blue car]$_y$*). The subject of a verb can map to either the possessor as exemplified above, or to the possessee (e.g., *The [car]$_y$ belongs to [John]$_x$*) (Aikhenvald and Dixon, 2012).

Within computational linguistics, possession relationships have usually been studied as part of larger studies that target all relations between arguments connected with a syntactic pattern (e.g., possessive constructions, nominals). Additionally, previous efforts have mostly targeted alienable possession—or alternatively, ownership. The work presented here takes a different approach. We start by pairing people (plausible possessors) with physical objects (plausible possesses). Then, we determine whether a possession relationship exists, and if so, (a) determine the type (alienable or control) and (b) assign temporal anchors with respect to the event of which the possessor is the subject. We target all verbs, not only prototypical verbs of possession (e.g., *have*, *get*). Thus, our approach extracts possessions intuitive to humans when there is no specific possession cue (e.g., we extract a control possession from *The [computer]$_y$ at work was slow, [I]$_x$ didn't get anything done*).

The main contributions of this paper are: (a) deterministic procedure to pair plausible possessors and possessees; (b) corpus annotating possession existence, possession type and temporal anchors; (c) detailed corpus analysis per verb and type of possession; and (d) experimental results showing that the task can be automated.

Proceedings of NAACL-HLT 2018, pages 496–505
New Orleans, Louisiana, June 1 - 6, 2018.

2 Possession Relations

The literature has studied possession relations extensively from theoretical and conceptual points of views. Here, we succinctly present some of the most influential works in the area.

The very definition of possession is not set in stone. Aikhenvald (2013) distinguishes three core meanings for possessive noun phrases that occur across languages: ownership (of property), whole-part (often referred to as part-whole), and kinship. Following a cross-linguistic perspective, she discusses possessions and time (present and former possession relationships, e.g., *my tooth* vs. *my former axe*), temporary and permanent possession (e.g., *borrow* vs. *acquire*) and others. Heine (1997) classifies possession relationships depending on the possessor and possessee. First, he makes a distinction between human (e.g., *[I]$_x$ have a [house]$_y$*) and non-human possessors (e.g. *[This house]$_x$ has [two bedrooms]$_y$*). Second, he differentiates three kinds of possession depending on the possessee: concrete possession (e.g., *[I]$_x$ have [two cats]$_y$*), social possession (e.g., *[I]$_x$ have [two sisters]$_y$*), and abstract possession (e.g., *[I]$_x$ have [an idea]$_y$*). Miller and Johnson-Laird (1976) differentiate between three kinds of possession: inherent, accidental, and physical; and provide the following example: *He owns an umbrella (inherent), but she's borrowed it (accidental), though she doesn't have it with her (physical).*

Possession relationships have also been defined in terms of their parameters. Stassen (2009) consider two parameters: permanent contact and control. These parameters are binary, and four kinds of possessions emerge from combining them: alienable (permanent contact: +, control: +), inalienable (+, -), temporary (-, +), and abstract (-, -). Similarly, Heine (1997) defines five binary parameters: human possessor, concrete possessee, spatial proximity, temporal permanence, and control. Combining these parameters, he defines 7 kinds of possessions: alienable, physical, temporary, inalienable, abstract, inanimate inalienable and inanimate alienable possession.

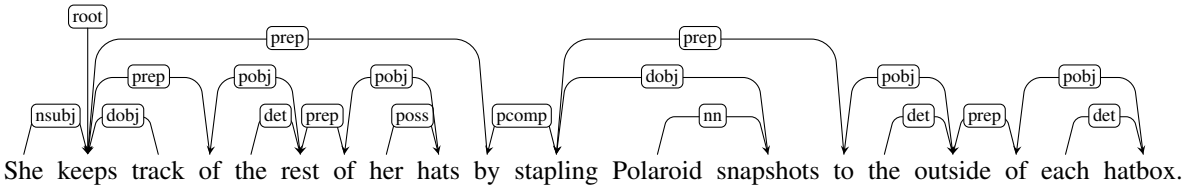
Most influential to the work presented here, Tham (2004) presents four types of possession: (a) inalienable (e.g., *John has a daughter*), (b) alienable (e.g., *John has a car*), (c) control (e.g., *John has the car (for the weekend)*), and (d) focus (e.g., *John has the window (to clean)*). In this paper, we target alienable and control possessions. We discard *inalienable* possessions because automated extraction has been studied before—at least partially, e.g., part-whole (Girju et al., 2006)—and *focus* possessions because they only occurred 5 times in the corpus we work with.

3 Previous Work

Within computational linguistics, possession relations have been mostly studied as one of the many relations encoded in a given syntactic construction. For example, Tratz and Hovy (2013) extract semantic relations within English possessives. They propose a set of 18 relations, e.g. *temporal* (e.g., *[today]$_x$'s [rates]$_y$*), *extent* (e.g., *[6 hours]$_y$' [drive]$_x$*). Their *controller / owner / user* relation (one relation with three aliases) is the closest relation to the alienable and control possessions we target in this paper. Unlike them, we distinguish between alienable and control possessions, and assign temporal anchors to possessions. Additionally, we are not restricted to possessive constructions. Instead, we start by pairing potential possessors and possessees within a sentence.

Extracting semantic relations between noun compounds (Nakov and Hearst, 2013; Tratz and Hovy, 2010) usually includes extracting possession relations, e.g., *[family]$_x$ [estate]$_y$*. Because they target noun compounds, they disregard numerous possessions encoded in text at the clause or sentence level. Although they do extract many relations from noun compounds beyond possessions, they do not distinguish between alienable and control possessions, or temporally anchor relations with respect to events in which the possessor participates.

To the best of our knowledge, the work by Banea et al. (2016) is the only one on extracting possession relations without imposing syntactic constraints. They build a dataset working with blog texts, but do not present results on automatic extraction. Their definition of possession includes alienable and control possessions, but they do not distinguish between them. Additionally, they only consider as possessors the author of a blog, and as possessees concrete nouns in the blog posts by the possessor. Regarding time, they annotate possessions at the time of the utterance. Unlike them, we distinguish between alienable and control possessions, and assign temporal anchors with respect to an event in which the possessor participates.



Pairs generated: (She, hats) and (She, hatbox)

Figure 2: Sample sentence, dependency tree and pairs (x, y) generated with the steps in Section 4.1.

antiquity.n.01, block.n.01, cone.n.01, container.n.01, covering.n.02, decker.n.01, device.n.01, fabric.n.01, fixture.n.01, float.n.01, furnishing.n.01, insert.n.01, layer.n.01, lemon.n.01, marker.n.01, plaything.n.01, ready-made.n.01, squeaker.n.01, strip.n.01, vehicle.n.01

Figure 1: WordNet synsets used to restrict possessees (y) when generating pairs (x, y). *lemma.n.y* indicates the yth synset of *n*oun *lemma*.

4 A Corpus of Possession Relations

We create a corpus[1] following two steps. First, we generate intrasentential pairs (x, y) of potential possessors (x) and possessees (y). Second, we annotate whether a possession exists, and if so, the type and temporal anchors. Generating pairs a priori proved more effective than giving annotators plain text and asking them to annotate possessions.

We add our annotations to OntoNotes (Hovy et al., 2006). Doing so has several advantages. First, OntoNotes contains texts from several domains and genres (e.g., conversational telephone speech, weblogs, broadcast), thus we not only work with newswire. Second, OntoNotes includes part-of-speech tags, named entities and parse trees, three annotation layers that allow us to streamline the corpus creation process.

4.1 Pairing Potential Possessors and Possessees

Our goal is to obtain pairs (x, y) such that it is plausible that x is the possessor of possessee y. To do so, we follow these steps:

1. Collect as potential possessors all PERSON named entities and personal pronouns (part-of-speech tag PRP) *I*, *he*, *she*, *we* and *they*.
2. Discard potential possessors that are not the nominal subject (*nsubj* syntactic dependency) of a verb. Let us name that verb $verb_x$.

[1] Available at http://www.cse.unt.edu/~blanco

possessee (y) is subsumed by	possessor (x) is a		
	pronoun	*person* NE	All
device.n.01	295	74	369
container.n.01	189	30	219
covering.n.02	138	54	192
vehicle.n.01	124	36	160
fabric.n.01	6	7	13
block.n.01	9	2	11
plaything.n.01	3	2	5
fixture.n.01	4	0	4
antiquity.n.01	2	0	2
other	4	0	4
All	774	205	979

Table 1: Counts of pairs (x, y) generated per type of potential possessor (x) and possessee (c).

3. For each possessor, collect as potential possessees all nouns reachable from $verb_x$ in the dependency tree and subsumed in WordNet (Miller, 1995) by the synsets in Figure 1.

Step (1) selects most people (not groups), and is inspired by Aikhenvald (2013, p. 11), who states that possessors are usually animate. Step (2) reduces the number of potential possessors, but note that we do not impose any restriction on $verb_x$, which may or may not be a verb of caused possession (Beavers, 2011). Finally, Step (3) restricts the kind of objects considered as possessees. The list of synsets was defined after analyzing the WordNet noun hierarchy and prior to generating pairs. Most of these synsets are children of *artifact.n.01*, other children of *artifact.n.01* were discarded because intuitively they cannot be possessees. For example, we discard *mystification.n.02: something designed to mystify or bewilder*.

Figure 2 shows a sample sentence and the pairs generated. Note that these pairs include distant possessor-possessee pairs, not only subject-object pairs. Nouns *track*, *rest*, *Polaroid* and *snapshot* are discarded as potential possessees because they are not subsumed by the synsets in Figure 1.

Labels	%	κ
yes, never, unk, inv	86.1	0.79
alienable, control	82.5	0.77
before yes, before no	83.6	0.68
during yes, during no	88.8	0.75
after yes, after no	83.6	0.59

Table 2: Inter-annotator agreements (raw percentage and Cohen's κ). κ values in the 0.60–0.80 range are considered *substantial*, over 0.80 would be *perfect* (Artstein and Poesio, 2008).

The total number of pairs generated after executing Steps (1–3) is 2,025. In order to reduce the annotation effort, we set to annotate 1,000 pairs. After trying several strategies, we reduce the number of pairs as follows. First, we discard pairs with verb$_x$ *see*, *think*, *believe*, *say* and *tell* because pilot annotations revealed that almost no possessions can be extracted from them (1,757 pairs left). Second, we discard pairs (x, y) such that verb$_x$ occurs five or less times (979 pairs left). Table 1 presents basic counts per type of possessor (named entity or personal pronoun) and possessee (WordNet synset) for the 979 pairs.

4.2 Annotating Possession Existence, Types and Temporal Information

After automatically generating pairs of potential possessors and possessees, annotators validate them manually. Annotations were done in-house, and the annotation interface showed the current sentence (with x, y and verb$_x$ highlighted), as well as the previous and next sentences.

The annotation process includes two major steps. First, annotators decide whether a possession relation exists between x and y based on the three sentences provided. More specifically, they choose from the following labels:

- `yes` if a possession exists at some point of time with respect to verb$_x$;
- `never` if a possession does not exist at any point of time with respect to verb$_x$;
- `unk` if it is sound to ask whether x is the possessor of possessee y, but there is not enough information to choose `yes` or `never`; and
- `inv` if either the potential possessor x is not animate, or the potential possessee y is nonsensical in the given context.

Second, annotators make two more decisions if the first label is `yes`:

- Possession type: whether the possession is `alienable` or `control`.

	Before		During		After	
	yes	*no*	*yes*	*no*	*yes*	*no*
Alienable	69.8	30.2	55.9	44.1	92.6	7.4
Control	28.8	71.2	85.3	14.7	33.3	66.7
All	52.0	48.0	68.7	31.3	67.8	32.2

Table 3: Percentage of alienable and control POSSESSION relations annotated *yes* and *no* per temporal anchor (before, during, after) with respect to verb$_x$ (i.e., the verb of which the possessor is the subject).

- Temporal anchors: whether the possession is true at some point of time `before`, `during`, and at some point of time `after` verb$_x$ takes place (three binary decisions).

Following the literature (Tham, 2004), we define *alienable* possession as a possessor owning a possessee, and *control* possession as a possessor having control of the possessee, but not necessarily ownership. Annotators were instructed to use world knowledge and fully interpret the sentences provided beyond what is explicitly stated. We present annotation examples in Section 5.1

Inter-Annotator Agreement. The annotations were done by two graduate students. Both of them annotated 35% of all pairs (possession existence, possession type and temporal anchors). We show inter-annotator agreements in Table 2. Cohen's κ for possession detection (labels `yes`, `never`, `unk` and `inv`) is 0.79, and 0.77 when including possession type (labels `alienable` and `control`). Answering whether the possession is true before, during or after verb$_x$ obtains lower coefficients: 0.68, 0.75 and 0.59 respectively. Not surprisingly, the agreement for `during` is higher. Note that κ coefficients in the range 0.60–0.80 are considered substantial, and coefficients over 0.80 are usually considered perfect (Artstein and Poesio, 2008). Given these high agreement, the rest of pairs (65%) were annotated once.

5 Corpus Analysis

Figure 3 presents percentages per label for all verbs and the top 10 most frequent verbs. Overall, 36.5% of pairs are validated (`alienable`: 20.6%, `control`: 15.9%), and only 5.8% of pairs are annotated `unk`. The relatively high percentage of `inv` label is mostly due to potential possessees that can only be possessed in certain contexts, e.g., compare *[They]$_x$ asked [regulators]$_y$ to suggest new ways to [...]* (`inv`) vs. *[They]$_x$ replaced the [regulators]$_y$ to control the flow of water* (`yes`).

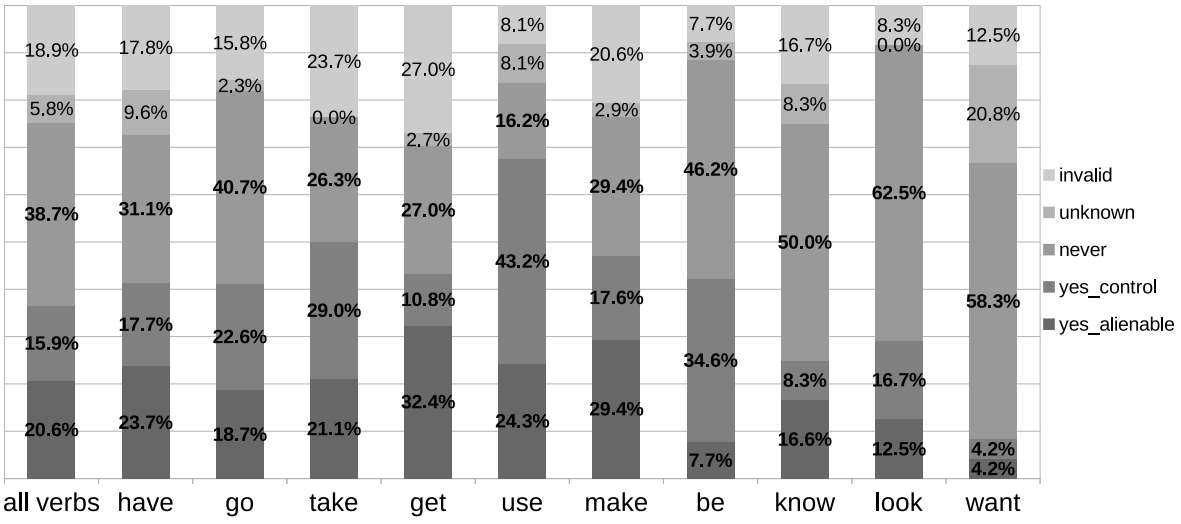


Figure 3: Percentage of labels for all verbs and the top 10 most frequent verbs of which the possessor is the subject. From left to right, they occurred 135, 44, 38, 37, 37, 34, 26, 24, 24, and 24 times respectively.

The percentage distributions depends heavily on the verb at hand. Note that several verbs with high `alienable` and `control` labels are not prototypical verbs of possession (e.g., *go*, *use*, *know*). When a possession holds, the type is most likely `control` for most verbs. The only exceptions are *have* (23.7% vs. 17.7%), *get* (32.4% vs. 10.8%), *make* (29.4% vs. 17.6%) and *know* (16.6% vs. 8.3%). The most productive verb as far as alienable possession is *get* (32.4%), and as far as `control` possessions, *use* (43.2%).

Labels per temporal anchor with respect to verb$_x$ (binary flags for before, during and after) and possession type are presented in Table 3. Alienable and control possessions show opposite trends for *before* and *after*, and substantially different distributions for *during*. The vast majority of control possessions are true during verb$_x$ (85.3% vs. 14.7%), as well as a more modest majority of alienable possessions (55.9% vs. 44.1%). Alienable and control possessions, however, have opposite temporal anchors for *before* and *after*. Specifically, most alienable possessions are true before and after verb$_x$ (69.8% and 92.6% respectively), and most control possessions are not true before and after verb$_x$ (71.2% and 66.7%).

5.1 Examples of Annotations

We present annotation examples using selected pairs of possessors and possessees in Table 4.

In Sentence (1), annotators interpreted that the relationship between *he* and *car* is an alienable possession. While not explicitly stated, annotators interpreted that *he* is an adult, and world knowledge tells us that most adults own the cars they drive unless a modifier indicates otherwise (e.g., *rental* car, my *father*'s car). Regarding temporal anchors, the possession between *he* and *car* is true before and during *died*, but not after.

Sentence (2) is a common example of alienable possession that is true after verb$_x$. The subject of a verb of creation (e.g., *make*, *build*) often becomes an alienable possessor of the direct object after the verb, but not before or during (because the object has not come into being yet).

Sentence (3) and (4) exemplify control possessions. In Sentence (3), *He* is borrowing *my father's car* for a period of time, and thus *He* has control over but does not own it. Regarding temporal anchors, nothing in the sentence indicates that *He* will have control over the car before or after *kept*. Note that our procedure to generate pairs would not generate the pair (*father*, *car*), but previous work has targeted possessives (Section 3).

In Example (4), verb$_x$ is *felt*, yet we extract a valid control possession. *I* is crew member of a warship and is describing his experience while on board. Annotators understood he had control over the ship (at least partially) before, during and after, as *felt* did not last long and there is no indication that *I* left the boat immediately before or after *felt*.

Sentences (5–7) present examples in which annotators did not annotate a possession relation (labels `never`, `unk`, and `inv`). In Sentence (5), the *mask* belongs to *Joseph*. There is no indication that a possession relation exists between *LaToya* and *mask*, although *LaToya* was in close spatial proximity of the *mask* worn by *Joseph*.

	Text (sentence of interest and context if relevant) and pair (x, y)	Label	B	D	A
1	But [he]$_x$ had extreme mood swings and [died]$_{verb_x}$ in a [car]$_y$ crash driving to work when I was five.	`alienable`	✓	✓	✗
2	After moving to the unoccupied zone, [Wang]$_x$ began [carving]$_{verb_x}$ [seals]$_y$ in his spare time to support himself.	`alienable`	✗	✗	✓
3	[He]$_y$ [kept]$_{verb_x}$ my father's [car]$_y$ for a year, without writing a confiscation order for it.	`control`	✗	✓	✗
4	The horror and the heroics of the [...] were relived this evening by one of the crew, Liutenant Ann Chamberlain. [I]$_x$ just [felt]$_{verb_x}$ a large—not so much a bang, but, to me, from where I was, it seemed like something had rammed the [ship]$_y$.	`control`	✓	✓	✓
5	[LaToya]$_x$ also [described]$_{verb_x}$ being awakened in the night by Joseph wearing a "monster [mask]$_y$."	`never`	n/a		
6	[They]$_x$ [asked]$_{verb_x}$ him to come to them immediately because the reported [car]$_y$ had been seized.	`unk`	n/a		
7	Will your political party straighten up and say, damn it, [we]$_x$ [have]$_{verb_x}$ to drop some of our ideological [baggage]$_y$?	`inv`	n/a		

Table 4: Example of annotations for selected pairs (x, y). Recall that temporal anchors (B: before, D: during, A: After) are annotated with respect to verb$_x$ and only if the main label is `yes`, i.e., if there is an alienable or control POSSESSION between x and y (Section 4.2).

In Sentence (6), it is the case that *They* have some knowledge about the *car* that was seized, and it appears that *him*—not *They*—may be the alienable possessor. It is unclear, however, whether *They* and *car* are related by a control possession, thus annotators chose label `unk`.

Finally, Sentence (7) exemplifies label `inv`. While *baggage* is most of the time a concrete object that passes the restrictions on potential possessees (Section 4), in this context, it is part of the metaphor *ideological baggage*. Since we only target concrete possessees, annotators chose `inv`.

6 Experiments and Results

We conduct experiments using Support Vector Machines and neural networks. Each pair (x, y) becomes an instance, and we create stratified train (80%) and test (20%) sets. We report results using the test set after tuning hyper parameters using 10-fold cross validation. More specifically, we train five classifiers and experiment with all instances but the ones annotated `inv`. The first classifier predicts possession existence (`yes`, `never` or `unk`). The second classifier predicts possession types, i.e., classifies pairs between which a possession holds (`yes`) into `alienable` or `control`. The third, fourth and fifth classifiers predict temporal anchors, i.e., classify pairs between which a possession holds—either alienable or control—into `before yes` or `before no`, `during yes` or `during no`, and `after yes` or `after no`.

6.1 Support Vector Machines

We trained the five classifiers using the SVM implementation in scikit-learn (Pedregosa et al., 2011). We tuned hyper-parameters C and γ using 10-fold cross validation, and used the features that are summarized in Table 5.

Verb features include the word and POS tag for the verb, previous and next tokens, as well as information regarding the outgoing and incoming dependencies. We also include a binary flag indicating whether the verb is a possession verb from the list collected by Viberg (2010, Table 1).

Possessor and *Possessee* features are very similar to *Verb* features, but we consider the concatenation of words and POS tags. *Possessee* features also include information derived from the WordNet hypernym paths to the root in the noun hierarchy, i.e., *entity.n.01*. More specifically, *WN synset* captures the synset from Figure 1 the possessee is subsumed by, and *WN path* are features capturing the top 6 synsets in the hypernym path from the possessee to *entity.n.01*. Finally, *Path* features include three syntactic paths (syntactic dependency types and up / down symbols): from the possessor to the verb, from the possessee to the verb, and from the possessor to the possessee. The feature set is heavily inspired in many previous works (e.g, (Gildea and Jurafsky, 2002)).

We experimented with SVMs to establish a strong supervised baseline using linguistic information, and to compare with neural networks that take as input only words along with information

	Feature	Description
Verb	word and tag	word form and part-of-speech tag of verb
	is_possession_verb	flag indicating whether *verb* is in the list of possession verbs
	previous, next tokens	word form and part-of-speech tags of the previous and next tokens
	dependency_out	outgoing syntactic dependency type
	dependencies_in	flags indicating the incoming syntactic dependencies
	left, right children	number of incoming syntactic dependencies to the left and right of *verb*
Possessor	words	concatenation of words
	pos tags	part-of-speech tag (full tag and first character, i.e., pronoun or noun)
	previous, next tokens	word form and part-of-speech tags of the previous and next tokens
	dependency_out	outgoing syntactic dependency type
	dependencies_in	flags indicating the incoming syntactic dependencies
Possessee	same as possessor	same features extracted for the possessor
	WN synset	WordNet synset from Figure 1 the possessee is subsumed by
	WN path	WordNet synsets from entity.n.01 to the possessee
Paths	possessor to verb	syntactic path between possessor and verb
	possessee to verb	syntactic path between possessee and verb
	possessor to possessee	syntactic path between possessor and possessee

Table 5: Feature set used to extract possession relations (existence, type and temporal anchors) with Support Vector Machines.) and possession type (`alienable` or `control`).

regarding who is the potential possessor, possessee and verb$_x$.

6.2 Neural Networks

We experiment with feedforward and Long Short-Term Memory networks, and use the implementations in Keras (Chollet et al., 2015) using TensorFlow backend (Abadi et al., 2015). All networks use GloVe embeddings with 100 dimensions (Pennington et al., 2014) and the Adam optimizer (Kingma and Ba, 2014). Regarding input, we experiment with the potential possessor *x*, possessee *y*, verb$_x$, and the rest of the sentence. The three architectures are depicted in Figure 4.

Feedforward Neural Network. The feedforward neural network takes as input the embeddings of the potential possessor *x*, possessee *y* and verb$_x$. It has a fully connected hidden layer with 50 neurons and uses softmax in the output layer of size 3 for predicting possession existence (`yes`, `never` and `unk`) or size 2 for predicting possession type (`alienable` and `control`) and temporal anchors (`yes` and `never` for before, during and after).

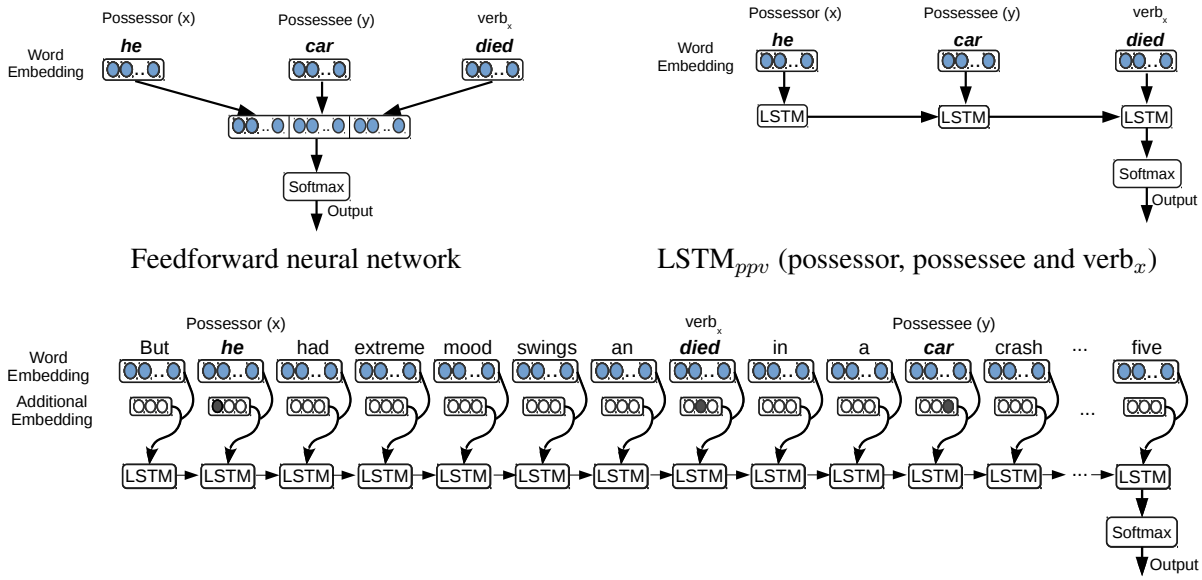
LSTM$_{ppv}$. The first Long Short-Term Memory network takes as input a fixed-length sequence consisting of the potential possessor *x*, possessee *y* and verb$_x$. We used 100 LSTM units (output dimension) and the output layer also uses softmax. While this LSTM has access to the same information than the feedforward network, we expect that the input, output and forget gates will learn to update the cell state to better solve our task.

LSTM$_{sent}$. The architecture of the second Long Short-Term Memory network is the same than LSTM$_{ppv}$, but the input is different. LSTM$_{sent}$ takes as input the sequence of words from which the potential possessor *x*, possessee *y* and verb$_x$ were extracted. Each element in the input is represented by the concatenation of its word embedding and an additional embedding indicating if the token is the potential possessor *x*, possessee *y*, verb$_x$, or none of them. Unlike the other two networks, LSTM$_{sent}$ has access to the full sentence, and we expect that the memory update mechanism (i.e., the input, output and forget gates) will learn the context most relevant for our task.

6.3 Results

Possession Existence and Type. Table 6 presents results obtained with the majority baseline (possession existence: always `never`, possession type: always `alienable`), SVMs and the three neural networks. All models outperform the majority baseline in both tasks (possession existence F1: 0.24, possession type F1: 0.40), and the three neural architectures outperform SVM (existence: 0.57–0.74 vs. 0.56, type: 0.61–0.67 vs. 0.58).

Regarding possession existence, the vanilla feedforward neural network alone performs similar to the SVM (F1: 0.57 vs. 0.56), indicating that word embeddings capture the kind of verbs and (potential) possessors and possessees more likely to have a possession relationship. Despite the small dataset (total: 979 pairs), the LSTMs out-



Feedforward neural network

LSTM$_{ppv}$ (possessor, possessee and verb$_x$)

LSTM$_{sent}$ (full sentence + additional embeddings for posessor, possessee and verb$_x$)

Figure 4: Neural network architectures: feedforward neural network (top left), LSTM$_{ppv}$ (top right), and LSTM$_{sent}$ (bottom). We exemplify the architectures with Example 1 from Table 4, *But [he]$_x$ had extreme mood swings and [died]$_{\text{verb}_x}$ in a [car]$_y$ crash driving to work when I was five* (only partially shown with LSTM$_{sent}$).

	Maj. baseline			SVM, all feats.			FFNN			LSTM$_{ppv}$			LSTM$_{sent}$		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
`yes`	.00	.00	.00	.57	.65	.61	.54	.53	.54	.65	.69	.67	.74	.72	.73
`never`	.41	1.0	.58	.61	.53	.56	.59	.59	.59	.71	.79	.75	.73	.82	.77
`unk`	.00	.00	.00	.44	.43	.44	.56	.59	.58	.75	.49	.59	.80	.65	.72
Weighted Avg.	.17	.41	.24	.56	.56	.56	.57	.57	.57	.70	.69	.69	.75	.75	.74
`alienable`	.56	1.0	.72	.64	.57	.61	.68	.75	.71	.67	.80	.73	.67	.60	.63
`control`	.00	.00	.00	.53	.59	.56	.64	.56	.60	.67	.50	.57	.56	.62	.59
Weighted Avg.	.32	.56	.40	.59	.58	.58	.66	.67	.66	.67	.67	.67	.62	.61	.61

Table 6: Results obtained using the majority baseline (possession existence: `never`, possession type: `alienable`), SVMs with the best feature combination (all features), and neural networks. Note that we report results for the possession existence (`yes`, `never` or `unk`) and possession type (`alienable` or `control`).

perform the feedforward neural network (0.74 vs. 0.57). LSTM$_{ppv}$ performs surprisingly well (F1: 0.69) even though it only has access to the possessor, possessee and verb$_x$. LSTM$_{sent}$ highly benefits from having access to the full sentence (F1: 0.74). This shows that context plays a vital role in deciding the existence of possession.

Regarding possession type, the feedforward neural network is comparable to LSTM$_{ppv}$. Intuitively, distinguishing between alienable and control possessions can be done mostly based on the possessor, possessee and verb$_x$, and the embeddings capture this kind of information. For example, verbs such as *use* and *rent* indicate a control possession, while *acquire* indicates alienable possession.

Temporal Anchors. Table 7 presents results obtained with SVMs and the best neural network architecture in this subtask. LSTM$_{ppv}$ performs similar to the SVM (before: 0.71 vs. 0.76, during: 0.75 vs. 0.72, after: 0.70 vs. 0.73). As expected, F1 scores are higher with the labels that occur more often: `yes` is more frequent than `never` with all temporal anchors, especially *during* and *after* (Table 3), and F1 scores for `yes` are higher than for `never` (before: 0.73 vs. 0.68, during: 0.82 vs. 0.59, after: 0.77 vs. 0.54).

7 Conclusions

Possession relations are present in all languages, and they can reflect relationships, values, concepts and cultural changes (Aikhenvald, 2013). In this

		Before			During			After		
		P	R	F1	P	R	F1	P	R	F1
SVM, all feats.	Yes	0.76	0.82	0.78	0.78	0.82	0.80	0.78	0.86	0.82
	No	0.77	0.71	0.74	0.57	0.52	0.55	0.61	0.48	0.54
	Weighted Avg.	0.76	0.76	0.76	0.72	0.72	0.72	0.72	0.74	0.73
LSTM_{ppv}	Yes	0.71	0.76	0.73	0.80	0.84	0.82	0.79	0.76	0.77
	No	0.71	0.65	0.68	0.62	0.57	0.59	0.52	0.57	0.54
	Weighted Avg.	0.71	0.71	0.71	0.74	0.75	0.75	0.70	0.69	0.70

Table 7: Results obtained using SVMs with the best feature combination (all features) and the best neural network architecture when predicting temporal anchors with respect to verb_y for a POSSESSION (both alienable and control).

paper, we mine possessions from text. Specifically, we extract alienable and control possessions, and specify temporal anchors with respect to the verb of which the possessor is the subject.

We have created the first corpus annotating types of possessions following two steps. First, we automatically pair potential possessors and possessees, resulting in 979 pairs. Second, we manually validate pairs by annotating possession existence (`yes`, `never`, `unk` and `inv`), types (`alienable` or `control`) and temporal anchors (`before yes` / `no`, `during yes` / `no`, `after yes` / `no`). Inter-annotator Cohen's κ coefficients show that the annotation task can be done reliably (Table 2). Experimental results show that the task can be automated, and that neural networks outperform SVMs trained with features extracted from linguistic structure although we experiment with a relatively small dataset.

Beyond fundamental research, we believe that mining possession types has several applications. For example, marketers may target people who do not alienably possess something, and certain skills may be inferred from the kind of objects people have control possessions over (e.g., an individual having a control possession of an 18-wheeler most likely knows how to drive large trucks and has a commercial driver's license).

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. `https://www.tensorflow.org/`.

A.Y Aikhenvald. 2013. Possession and ownership: a cross-linguistic perspective. In A.Y. Aikhenvald and R.M.W. Dixon, editors, *Possession and Ownership: A Cross-Linguistic Typology*, Oxford University Press, Oxford, chapter 1, pages 1–64.

A.Y. Aikhenvald and R.M.W. Dixon. 2012. *Possession and Ownership: A Cross-Linguistic Typology*. Explorations in Linguistic Typology. OUP Oxford.

Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Comput. Linguist.* 34(4):555–596.

Carmen Banea, Xi Chen, and Rada Mihalcea. 2016. Building a dataset for possessions identification in text. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association (ELRA), Paris, France.

J. Beavers. 2011. An aspectual analysis of ditransitive verbs of caused possession in english. *Journal of Semantics* 28(1):1–54.

François Chollet et al. 2015. Keras. `https://github.com/fchollet/keras`.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Comput. Linguist.* 28(3):245–288. `https://doi.org/10.1162/089120102760275983`.

Roxana Girju, Adriana Badulescu, and Dan Moldovan. 2006. Automatic discovery of part-whole relations. *Comput. Linguist.* 32(1):83–135. `https://doi.org/10.1162/coli.2006.32.1.83`.

B. Heine. 1997. *Possession: Cognitive Sources, Forces, and Grammaticalization*. Cambridge Studies in Linguistics. Cambridge University Press.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: the 90% Solution. In *NAACL '06: Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers on XX*. Association for Computational Linguistics, Morristown, NJ, USA, pages 57–60. http://portal.acm.org/citation.cfm?id=1614064.

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

G.A. Miller and P.N. Johnson-Laird. 1976. *Language and perception*. Belknap Press. Belknap Press of Harvard University Press.

George A. Miller. 1995. Wordnet: A lexical database for english. In *Communications of the ACM*. volume 38, pages 39–41.

Preslav I. Nakov and Marti A. Hearst. 2013. Semantic interpretation of noun compounds using verbal and other paraphrases. *ACM Trans. Speech Lang. Process.* 10(3):13:1–13:51.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

L. Stassen. 2009. *Predicative Possession*. Oxford Studies in Typology and Linguistic Theory. OUP Oxford.

Shiao Wei Tham. 2004. *Representing Possessive Predication: Semantic Dimensions and Pragmatic Bases*. Ph.D. thesis, Stanford University.

Stephen Tratz and Eduard Hovy. 2010. A taxonomy, dataset, and classifier for automatic noun compound interpretation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '10, pages 678–687.

Stephen Tratz and Eduard H. Hovy. 2013. Automatic interpretation of the english possessive. In *ACL (1)*. The Association for Computer Linguistics, pages 372–381.

Ake Viberg. 2010. Basic verbs of possession. *CogniTextes* 4.

Neural Tensor Networks with Diagonal Slice Matrices

Takahiro Ishihara[1] **Katsuhiko Hayashi**[2] **Hitoshi Manabe**[1]
Masahi Shimbo[1] **Masaaki Nagata**[3]
[1] Nara Institute of Science and Technology [2] Osaka University
[3] NTT Communication Science Laboratories
[1] {ishihara.takahiro.in0, manabe.hitoshi.me0, shimbo}@is.naist.jp
[2] khayashi0201@gmail.com [3] nagata.masaaki@lab.ntt.co.jp

Abstract

Although neural tensor networks (NTNs) have been successful in many natural language processing tasks, they require a large number of parameters to be estimated, which often results in overfitting and long training times. We address these issues by applying eigendecomposition to each slice matrix of a tensor to reduce the number of parameters. We evaluate our proposed NTN models in two tasks. First, the proposed models are evaluated in a knowledge graph completion task. Second, a recursive NTN (RNTN) extension of the proposed models is evaluated on a logical reasoning task. The experimental results show that our proposed models learn better and faster than the original (R)NTNs.

1 Introduction

Alongside the nonlinear activation functions, linear mapping by matrix multiplication is an essential component of neural network (NN) models, as it determines the feature interaction and thus the expressiveness of models. In addition to the matrix-based mapping, neural tensor networks (NTNs) (Socher et al., 2013a) employ a 3-dimensional tensor to capture direct interactions among input features. Due to the large expressive capacity of 3D tensors, NTNs have been successful in an array of natural language processing (NLP) and machine learning tasks, including knowledge graph completion (KGC) (Socher et al., 2013a), sentiment analysis (Socher et al., 2013b), and reasoning with logical semantics (Bowman et al., 2015). However, since a 3D tensor has a large number of parameters, NTNs need longer time to train than other NN models. Moreover, the millions of parameters often make the model suffer from overfitting (Yang et al., 2015).

To solve these problems, we propose two new parameter reduction techniques for NTNs. These techniques drastically decrease the number of parameters in an NTN without diminishing its expressiveness. We use the matrix decomposition techniques that are utilized for KGC in Yang et al. (2015) and Trouillon et al. (2016). Yang et al. (2015) imposed a constraint that a matrix in the bilinear term in their model had to be diagonal. As mentioned in a subsequent section, this is essentially equal to assuming that the matrix be symmetric and performing eigendecomposition. Trouillon et al. (2016) also applied eigendecomposition to a matrix by regarding it as the real part of a normal matrix. Following these studies, we perform simultaneous diagonalization on all slice matrices of a NTN tensor. As a result, mapping by a 3D ($n \times n \times k$) tensor is replaced with an array of k "triple inner products" of two input vectors and a weight vector. Thus, we obtain two new NTN models where the number of parameters is reduced from $O(n^2k)$ to $O(nk)$.

On a KGC task, these parameter-reduced NTNs (NTN-Diag and NTN-Comp) alleviate overfitting and outperform the original NTN. Moreover, our proposed NTNs can learn faster than the original NTN. We also show that our proposed models perform better and learn faster in a recursive setting by examining a logical reasoning task.

2 Background

We consider mapping in a neural network (NN) layer that takes two vectors as input, such as recursive neural networks. Recurrent neural networks also has this structure, with one input vector being the hidden state from the previous time step. As a mapping before activation in the NN layer, linear mapping (matrix multiplication) is commonly used:

$$\boldsymbol{W}_1\boldsymbol{x}_1 + \boldsymbol{W}_2\boldsymbol{x}_2 = [\boldsymbol{W}_1, \boldsymbol{W}_2] \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix} = \boldsymbol{W}\boldsymbol{x}.$$

Proceedings of NAACL-HLT 2018, pages 506–515
New Orleans, Louisiana, June 1 - 6, 2018.

Here, since $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^n, \boldsymbol{W}_1, \boldsymbol{W}_2 \in \mathbb{R}^{k \times n}$, this linear mapping is a transformation from $\mathbb{R}^{2n}$ to $\mathbb{R}^k$. Linear mapping, which is a standard component of NNs, has been applied successfully in many tasks. However, it cannot consider the interaction between different components of two input vectors, which renders it not ideal for modeling complex compositional structures such as trees and graphs.

To alleviate this problem, some models such as NTNs (Socher et al., 2013a) have explored 3D tensors to yield more expressive mapping:

$$\begin{aligned} \boldsymbol{x}_1^{\mathrm{T}} \boldsymbol{W}^{[1:k]} \boldsymbol{x}_2 &= \begin{bmatrix} \boldsymbol{x}_1^{\mathrm{T}} \boldsymbol{W}^{[1]} \boldsymbol{x}_2 \\ \boldsymbol{x}_1^{\mathrm{T}} \boldsymbol{W}^{[2]} \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_1^{\mathrm{T}} \boldsymbol{W}^{[k]} \boldsymbol{x}_2 \end{bmatrix} \\ &= \begin{bmatrix} \mathrm{sum}\left(\boldsymbol{W}^{[1]} \odot (\boldsymbol{x}_1 \otimes \boldsymbol{x}_2)\right) \\ \mathrm{sum}\left(\boldsymbol{W}^{[2]} \odot (\boldsymbol{x}_1 \otimes \boldsymbol{x}_2)\right) \\ \vdots \\ \mathrm{sum}\left(\boldsymbol{W}^{[k]} \odot (\boldsymbol{x}_1 \otimes \boldsymbol{x}_2)\right) \end{bmatrix} \end{aligned}$$

where $\boldsymbol{W}^{[1:k]} \in \mathbb{R}^{n \times n \times k}$. The output of this mapping is an array of k bilinear products in the form of $\boldsymbol{x}_1^{\mathrm{T}} \boldsymbol{W}^{[i]} \boldsymbol{x}_2$. Thus, this is also a transformation from $\mathbb{R}^{2n}$ to $\mathbb{R}^k$. Each element of the output of this mapping equals the sum of $\boldsymbol{W}^{[i]} \odot (\boldsymbol{x}_1 \otimes \boldsymbol{x}_2)$, where $\odot$ and $\otimes$ represent, respectively, the Hadamard and the outer products. Hence this mapping captures the direct interaction between different components (or "features") in two input vectors. Thanks to this expressiveness, NTNs are effective in tasks such as knowledge graph completion (Socher et al., 2013a), sentiment analysis (Socher et al., 2013b), and logical reasoning (Bowman et al., 2015).

Although mapping by a 3D tensor provides expressiveness, it has a large number ($O(n^2k)$) of parameters. Due to this, NTNs often suffer from overfitting and long training times.

3 Matrix Decomposition

3.1 Simple Matrix Decomposition (SMD)

To reduce the number of parameters of a slice matrix $\boldsymbol{W}^{[i]} \in \mathbb{R}^{n \times n}$ in a tensor, simple matrix decomposition (SMD) is commonly used (Bai et al., 2009). SMD factorizes $\boldsymbol{W}^{[i]}$ into a product of two low rank matrices $\boldsymbol{S}^{[i]} \in \mathbb{R}^{n \times m}$ and $\boldsymbol{T}^{[i]} \in \mathbb{R}^{m \times n}$ ($m \ll n$):

$$\boldsymbol{W}^{[i]} \simeq \boldsymbol{S}^{[i]} \boldsymbol{T}^{[i]}. \tag{1}$$

By plugging (1) into bilinear term $\boldsymbol{x}_1^{\mathrm{T}} \boldsymbol{W}^{[i]} \boldsymbol{x}_2$, we obtain the approximation $\boldsymbol{x}_1^{\mathrm{T}} \boldsymbol{S}^{[i]} \boldsymbol{T}^{[i]} \boldsymbol{x}_2$. SMD reduces the number of parameters of $\boldsymbol{W}^{[i]}$ from n^2 to $2nm$. However, the dimension m for $\boldsymbol{S}$ and $\boldsymbol{T}$ is a hyperparameter and must be determined prior to training.

3.2 Simultaneous Diagonalization

This section introduces two techniques that can simultaneously diagonalize all slice matrices $\boldsymbol{W}^{[1]}, \ldots, \boldsymbol{W}^{[i]}, \ldots, \boldsymbol{W}^{[k]} \in \mathbb{R}^{n \times n}$. As described in (Liu et al., 2017), we make use of the fact that if matrices $\boldsymbol{V}^{[1:k]}$ form a commuting family: i.e., $\boldsymbol{V}^{[i]} \boldsymbol{V}^{[j]} = \boldsymbol{V}^{[j]} \boldsymbol{V}^{[i]}, \forall i, j \in \{1, 2, \ldots, k\}$, they can be diagonalized by a shared orthogonal or unitary matrix. Both of the two techniques reduce the number of parameters of $\boldsymbol{W}^{[i]}$ to $O(n)$ from $O(n^2)$.

3.2.1 Orthogonal Diagonalization

Many NLP datasets contain symmetric patterns. For example, if binary relation *(Bob, is_relative_of, Alice)* holds in a knowledge graph, then *(Alice, is_relative_of, Bob)* should also hold in it. English phrases "dog and cat" and "cat and dog" have identical meaning. For symmetric structures, we can reasonably suppose that each slice matrix $\boldsymbol{W}^{[i]}$ of a 3D tensor is symmetric because $\boldsymbol{x}_1^{\mathrm{T}} \boldsymbol{W}^{[i]} \boldsymbol{x}_2$ must equal $\boldsymbol{x}_2^{\mathrm{T}} \boldsymbol{W}^{[i]} \boldsymbol{x}_1$.

When $\boldsymbol{W}^{[i]} \in \mathbb{R}^{n \times n}$ is symmetric, it can be diagonalized as:

$$\boldsymbol{W}^{[i]} = \boldsymbol{O}^{[i]} \boldsymbol{W}^{[i]\prime} \boldsymbol{O}^{[i]\mathrm{T}}$$

where $\boldsymbol{O}^{[i]} \in \mathbb{R}^{n \times n}$ is an orthogonal matrix and $\boldsymbol{W}^{[i]\prime} \in \mathbb{R}^{n \times n}$ is a diagonal matrix. Note that an orthogonal matrix $\boldsymbol{O}^{[i]}$ may not be equal to $\boldsymbol{O}^j$ if $i \neq j$. However, if all of the slice matrices $\boldsymbol{W}^{[1]}, \ldots, \boldsymbol{W}^{[i]}, \ldots, \boldsymbol{W}^{[k]} \in \mathbb{R}^{n \times n}$ are commuting, we can diagonalize every slice matrix with the same orthogonal matrix $\boldsymbol{O}$. By substituting $\boldsymbol{W}^{[i]}$ with $\boldsymbol{O} \boldsymbol{W}^{[i]\prime} \boldsymbol{O}^{\mathrm{T}}$ into bilinear term $\boldsymbol{x}_1^{\mathrm{T}} \boldsymbol{W}^{[i]} \boldsymbol{x}_2$, we can rewrite it as follows:

$$\begin{aligned} \boldsymbol{x}_1^{\mathrm{T}} \boldsymbol{W}^{[i]} \boldsymbol{x}_2 &= \boldsymbol{x}_1^{\mathrm{T}} \boldsymbol{O} \boldsymbol{W}^{[i]\prime} \boldsymbol{O}^{\mathrm{T}} \boldsymbol{x}_2 \\ &= \boldsymbol{y}_1^{\mathrm{T}} \boldsymbol{W}^{[i]\prime} \boldsymbol{y}_2 \\ &= \langle \boldsymbol{y}_1, \boldsymbol{w}^{[i]}, \boldsymbol{y}_2 \rangle \end{aligned} \tag{2}$$

where $\boldsymbol{y}_1 = \boldsymbol{O}^{\mathrm{T}} \boldsymbol{x}_1$, $\boldsymbol{y}_2 = \boldsymbol{O}^{\mathrm{T}} \boldsymbol{x}_2$, $\boldsymbol{w}^{[i]} = \mathrm{diag}(\boldsymbol{W}^{[i]\prime}) \in \mathbb{R}^n$ and $\langle \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \rangle$ denotes a "triple inner product" defined by $\langle \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \rangle = \sum_{l=1}^{n} a_l b_l c_l$. This reduces the number of parameters in a single slice matrix from n^2 to n.

3.2.2 Unitary Diagonalization

Since most of the structures in the NLP data are not symmetric, the symmetric matrix assumption is usually violated. To obtain more expressive diagonal matrix, we regard each slice matrix $\boldsymbol{W}^{[i]}$ as the real part of a complex matrix and consider its eigendecomposition.

For any real matrix $\boldsymbol{W}^{[i]}$, there exists a complex normal matrix $\boldsymbol{Z}^{[i]}$ whose real part is equal to it: $\boldsymbol{W}^{[i]} = \Re\left(\boldsymbol{Z}^{[i]}\right)$. $\Re(\cdot)$ represents an operation that takes the real part of a complex number, vector or matrix. Further, any complex normal matrix can be diagonalized by a unitary matrix. With these two properties, any real matrix $\boldsymbol{W}^{[i]}$ can be diagonalized as follows (Trouillon et al., 2016):

$$\boldsymbol{W}^{[i]} = \Re\left(\boldsymbol{Z}^{[i]}\right) = \Re\left(\boldsymbol{U}^{[i]}\boldsymbol{Z}^{[i]\prime}\boldsymbol{U}^{[i]*}\right).$$

Here, $\boldsymbol{U}^{[i]} \in \mathbb{C}^{n\times n}$ is a unitary matrix, $\boldsymbol{Z}^{[i]\prime} \in \mathbb{C}^{n\times n}$ is a diagonal matrix, and $\boldsymbol{U}^{[i]*}$ is the conjugate transpose of $\boldsymbol{U}^{[i]}$. To guarantee that every slice matrix can be diagonalized with the same unitary matrix $\boldsymbol{U}$ instead of $\boldsymbol{U}^{[i]}$, we assume all of the normal matrices $\boldsymbol{Z}^{[1]}, \ldots, \boldsymbol{Z}^{[i]}, \ldots, \boldsymbol{Z}^{[k]} \in \mathbb{C}^{n\times n}$ are commuting as in Section 3.2.1.

Substituting $\Re\left(\boldsymbol{U}\boldsymbol{Z}^{[i]\prime}\boldsymbol{U}^{*}\right)$ whose $\boldsymbol{U}$ is the same unitary matrix in all slice matrices, we can rewrite every bilinear term $\boldsymbol{x}_1^{\mathrm{T}}\boldsymbol{W}^{[i]}\boldsymbol{x}_2$ as follows:

$$\begin{aligned}
\boldsymbol{x}_1^{\mathrm{T}}\boldsymbol{W}^{[i]}\boldsymbol{x}_2 &= \Re\left(\langle \boldsymbol{y}_1, \boldsymbol{w}^{[i]}, \overline{\boldsymbol{y}_2}\rangle\right) \\
&= \langle \Re(\boldsymbol{y_1}), \Re(\boldsymbol{w}^{[i]}), \Re(\boldsymbol{y}_2)\rangle \\
&\quad + \langle \Re(\boldsymbol{y_1}), \Im(\boldsymbol{w}^{[i]}), \Im(\boldsymbol{y}_2)\rangle \\
&\quad + \langle \Im(\boldsymbol{y_1}), \Re(\boldsymbol{w}^{[i]}), \Im(\boldsymbol{y}_2)\rangle \\
&\quad - \langle \Im(\boldsymbol{y_1}), \Im(\boldsymbol{w}^{[i]}), \Re(\boldsymbol{y}_2)\rangle, \quad (3)
\end{aligned}$$

where $\boldsymbol{y}_1 = \boldsymbol{U}^{\mathrm{T}}\boldsymbol{x}_1$, $\overline{\boldsymbol{y}_2} = \boldsymbol{U}^{*}\boldsymbol{x}_2$, $\boldsymbol{w}^{[i]} = \mathrm{diag}(\boldsymbol{Z}^{[i]\prime}) \in \mathbb{C}^n$, and $\langle \boldsymbol{y}_1, \boldsymbol{w}^{[i]}, \overline{\boldsymbol{y}_2}\rangle$ is the triple Hermitian inner product of $\boldsymbol{y}_1$, $\boldsymbol{w}^{[i]}$ and $\boldsymbol{y}_2$ defined by $\langle \boldsymbol{a}, \boldsymbol{b}, \overline{\boldsymbol{c}}\rangle = \sum_{l=1}^{n} a_l b_l \overline{c_l}$. This technique reduces the number of parameters of the matrices from n^2 to $2n$. As shown in the right-hand side of Eq. (3), $\Re\left(\langle \boldsymbol{y}_1, \boldsymbol{w}^{[i]}, \overline{\boldsymbol{y}_2}\rangle\right)$ can be replaced with three additions and a subtraction of the triple inner product of real vectors.

4 Neural Network Models

This section introduces the baseline and our proposed models. After describing them, we explain how to extend them for handling compositional structures like binary trees.

Model	# of Parameters
NN	$(2n+1)k$
NTN	$(n^2+2n+1)k$
NTN-SMD	$(2mn+2n+1)k$
NTN-Diag	$(3n+1)k$
NTN-Comp	$(6n+1)k$

Table 1: Comparison of the number of parameters among the models

4.1 Baseline Models

Neural Network (NN)

First, we describe a standard single layer neural network (NN) model for two vectors $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^n$. The model uses linear mapping $\boldsymbol{V} \in \mathbb{R}^{k\times 2n}$ to combine two input vectors:

$$f(\boldsymbol{V}\begin{bmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{bmatrix} + \boldsymbol{b})$$

where $\boldsymbol{b} \in \mathbb{R}^k$ is a bias term and f is a non-linear activation function. The NN model has only $(2n+1)k$ parameters, and does not consider the direct interactions between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$.

Neural Tensor Network (NTN)

Socher et al. (2013a) proposed a neural tensor network (NTN) model that uses a 3D tensor $\boldsymbol{W}^{[1:k]} \in \mathbb{R}^{n\times n\times k}$ to combine two input vectors:

$$f(\boldsymbol{x}_1^{\mathrm{T}}\boldsymbol{W}^{[1:k]}\boldsymbol{x}_2 + \boldsymbol{V}\begin{bmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{bmatrix} + \boldsymbol{b}).$$

Unlike the standard NN model, NTN can directly relate two input vectors using a tensor. However, it has too many parameters; $(n^2+2n+1)k$.

NTN-SMD

Although the NTN model has tremendous expressive power, it is extremely time-consuming to compute, since a naive 3D tensor product incur $O(n^2k)$ computation time. To overcome this weakness, Zhao et al. (2015) and Liu et al. (2015) independently introduced simple matrix decomposition (SMD) to the NTN model by replacing each slice matrix $\boldsymbol{W}^{[i]}$ with its factorized approximation given by Eq. (1):

$$f(\boldsymbol{x}_1^{\mathrm{T}}\boldsymbol{S}^{[1:k]}\boldsymbol{T}^{[1:k]}\boldsymbol{x}_2 + \boldsymbol{V}\begin{bmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{bmatrix} + \boldsymbol{b})$$

where $\boldsymbol{S}^{[1:k]} \in \mathbb{R}^{n\times m\times k}, \boldsymbol{T}^{[1:k]} \in \mathbb{R}^{m\times n\times k}$. When $m \ll n$, the NTN-SMD model drastically reduces the number of parameters compared to the original NTN model; i.e., from $(n^2+2n+1)k$ to $(2mn+2n+1)k$.

4.2 NTNs with Diagonal Slice Matrices

In this paper, we introduce two new NTN models: NTN-Diag and NTN-Comp, both of which reduce the number of parameters in a 3D tensor more than NTN-SMD with little loss in the model's generalization performance. Table 1 summarizes the number of parameters in each model.

NTN-Diag

We replace all slice matrices $\boldsymbol{W}^{[i]}$ of $\boldsymbol{W}^{[1:k]}$ with the triple inner product formulation of Eq. (2) by assuming that they are symmetric and commuting. As a result, we derive the following new NTN formulation:

$$f\left(\begin{bmatrix} \langle \boldsymbol{x}_1, \boldsymbol{w}^{[1]}, \boldsymbol{x}_2 \rangle \\ \vdots \\ \langle \boldsymbol{x}_1, \boldsymbol{w}^{[k]}, \boldsymbol{x}_2 \rangle \end{bmatrix} + \boldsymbol{V} \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix} + \boldsymbol{b}\right)$$

where $\boldsymbol{w}^{[i]} \in \mathbb{R}^n, \forall i \in \{1, 2, \ldots, k\}$. Thus, under the symmetric and commuting matrix constraints, we regard mapping by a 3D tensor as an array of k triple inner products. The total number of parameters is just $(3n + 1)k$.

NTN-Comp

By assuming that $\boldsymbol{W}^{[1]}, \ldots, \boldsymbol{W}^{[i]}, \ldots, \boldsymbol{W}^{[k]}$ are real parts of normal matrices forming a commuting family, we can replace each slice matrix of a tensor term in NTN with the triple Hermitian inner product shown in Eq. (3):

$$f\left(\begin{bmatrix} \Re\left(\langle \boldsymbol{x}_1, \boldsymbol{w}^{[1]}, \overline{\boldsymbol{x}_2} \rangle\right) \\ \vdots \\ \Re\left(\langle \boldsymbol{x}_1, \boldsymbol{w}^{[k]}, \overline{\boldsymbol{x}_2} \rangle\right) \end{bmatrix} + \Re\left(\boldsymbol{V} \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix}\right) + \boldsymbol{b}\right)$$

where $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{C}^n$, $\boldsymbol{V} \in \mathbb{C}^{n \times n}$ and $\boldsymbol{w}^{[i]} \in \mathbb{C}^n$, $\forall i \in \{1, 2, \ldots, k\}$. Similar to NTN-Diag, we regard mapping by a 3D tensor as an array of k triple Hermitian inner products. The total number of parameters is just $(6n + 1)k$. As is clear of its form, NTN-Diag is a special case of NTN-Comp whose vectors $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ and $\boldsymbol{w}^{[i]}$ are constrained to be real.

4.3 Recursive Neural Tensor Networks

We extend the above NTN models to handle compositional structures. As a representative of compositional structures, we consider a binary tree where each NTN layer computes a vector representation for a node by combining two vectors from its child nodes in the lower layer. Except for NTN-Comp, the models implement mappings $\mathbb{R}^n \to \mathbb{R}^k$ so that each of their layers can receive its lower layer's output directly, if k equals to n. Thus, the models do not have to be modified for them. However, NTN-Comp cannot receive its lower layer's output as it is because NTN-Comp is a mapping from $\mathbb{C}^n$ to $\mathbb{R}^k$. To solve this problem, we set k to $2n$ and treat the output $\boldsymbol{y}' \in \mathbb{R}^{2n}$ as the concatenation of vectors representing the real and imaginary parts of $\boldsymbol{y} \in \mathbb{C}^n$:

$$\Re(\boldsymbol{y}) = (y'_1, \cdots, y'_n),\ \Im(\boldsymbol{y}) = (y'_{n+1}, \cdots, y'_{2n}).$$

Note that this approach is valid since Eq. (3) can actually be defined in real vector space by transforming the complex vectors in $\mathbb{C}^n$ into real vectors in $\mathbb{R}^{2n}$.

5 Related Work

Knowledge Graph Completion

In KGC, researchers usually design scoring function Φ for the given triplet (s, r, o) to judge whether it is a fact or not. Here (s, r, o) denotes that entity s is linked to entity o by relation r. RESCAL (Nickel et al., 2011) uses $\boldsymbol{e}_s^{\mathrm{T}} \boldsymbol{W}_r \boldsymbol{e}_o$ as Φ, where $\boldsymbol{e}_s, \boldsymbol{e}_o$ are entity embedding vectors and $\boldsymbol{W}_r$ is an embedding matrix of relation r. This bilinear operation is effective for the task, but its computational cost is high and it suffers from overfitting. To overcome these problems, DistMult (Yang et al., 2015) adopts the triple inner product $\langle \boldsymbol{e}_s, \boldsymbol{w}_r, \boldsymbol{e}_o \rangle$ as Φ, where $\boldsymbol{w}_r$ is an embedding vector of relation r. This solves those problems, but it degrades the model's ability to capture directionality of relations, because the scoring function of DistMult is symmetric with respect to s and o; i.e., $\langle \boldsymbol{e}_s, \boldsymbol{w}_r, \boldsymbol{e}_o \rangle = \langle \boldsymbol{e}_o, \boldsymbol{w}_r, \boldsymbol{e}_s \rangle$. To reconcile the complexity and expressiveness of a model, ComplEx (Trouillon et al., 2016) uses complex vectors for entity and relation embeddings. As scoring function Φ, they adopted the triple Hermitian inner product $\Re\left(\langle \boldsymbol{e}_s, \boldsymbol{w}_r, \overline{\boldsymbol{e}_o} \rangle\right)$, where $\overline{\boldsymbol{e}_o}$ denotes the complex conjugate of $\boldsymbol{e}_o$. Since $\Re\left(\langle \boldsymbol{e}_s, \boldsymbol{w}_r, \overline{\boldsymbol{e}_o} \rangle\right) \neq \Re\left(\langle \boldsymbol{e}_o, \boldsymbol{w}_r, \overline{\boldsymbol{e}_s} \rangle\right)$, ComplEx solves the expressiveness problem of DistMult without full matrices as relation embeddings. We can regard DistMult as a special case of RESCAL with a symmetric matrix constraint on $\boldsymbol{W}_r$. ComplEx is also a RESCAL variant with $\boldsymbol{W}_r$ as the real part of a normal matrix. Our research is based on these works, but to the best of our knowledge, no previous work applied this ap-

proach to reduce the number of parameters in a tensor.

NN Architectures

To give additional expressiveness power to standard (R)NNs, many architectures have been proposed, such as LSTM (Hochreiter and Schmidhuber, 1997), GRU (Cho et al., 2014), and CNN (LeCun et al., 1998). NTN (Socher et al., 2013a) and RNTN (Socher et al., 2013b) are other such architectures. However, (R)NTNs differ in that they only add 3D tensor mapping to standard neural networks. Thus, they can also be regarded as a powerful basic component of NNs because 3D tensor mapping can be applied to more complicated architectures such as those examples.

Parameter Reduction in NN

Several researchers reduced the number of parameters of NNs by using specific parameter sharing mechanisms. Cheng et al. (2015) used circulant matrix mapping instead of conventional linear mapping and improved the time complexity of the matrix-vector product by using Fast Fourier Transformation (FFT). Circulant matrix

$$C(\boldsymbol{w}) = \begin{bmatrix} w_1 & w_n & \dots & w_3 & w_2 \\ w_2 & w_1 & \dots & w_4 & w_3 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ w_{n-1} & w_{n-2} & \dots & w_1 & w_n \\ w_n & w_{n-1} & \dots & w_2 & w_1 \end{bmatrix}$$

for $\boldsymbol{w}^{\mathrm{T}} = (w_1, \dots, w_n)$ can be factorized into $\mathfrak{F}^{-1}\,\mathrm{diag}(\mathfrak{F}\,\boldsymbol{w})\,\mathfrak{F}$ with the Fourier matrix $\mathfrak{F}$. By assuming each slice matrix $\boldsymbol{W}^{[i]}$ of $\boldsymbol{W}^{[1:k]}$ is circulant, we get the same scoring function as that in Eq. (3); $\boldsymbol{x}_1^{\mathrm{T}}\boldsymbol{W}^{[i]}\boldsymbol{x}_2 = \boldsymbol{x}_1^{\mathrm{T}}\,\mathfrak{F}^{-1}\,\mathrm{diag}(\mathfrak{F}\,\boldsymbol{w}^{[i]})\,\mathfrak{F}\,\boldsymbol{x}_2 = \Re(\langle \boldsymbol{x}_1', \boldsymbol{w}^{[i]\prime}, \overline{\boldsymbol{x}_2}' \rangle)$ where $\boldsymbol{x}_1' = \overline{\mathfrak{F}\,\boldsymbol{x}_1}$, $\boldsymbol{x}_2' = \overline{\mathfrak{F}\,\boldsymbol{x}_2}$, and $\boldsymbol{w}^{[i]\prime} = \frac{1}{n}\,\mathrm{diag}(\mathfrak{F}\,\boldsymbol{w}^{[i]})$ are complex vectors in $\mathbb{C}^n$. In this sense, NTN-Comp is equivalent to NTN where slice matrices of the 3D tensor are restricted to be circulant. Hayashi and Shimbo (2017) established a more detailed proof of the equivalence. Lu et al. (2016) employed a Toeplitz-like structured matrix, reducing parameters of LSTM. Chen et al. (2015) used a feature hashing technique to reduce parameters in RNN. Although these techniques can also be extended to reduce the number of tensor-related parameters in NTN, the former needs FFT operations; i.e., $O(n \log n)$ computation time, and the latter's contribution is only a reduction in memory consumption.

Dataset	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	#Train	#Valid	#Test
FB15k	14,951	1,345	483,142	50,000	59,701
WN18	40,943	18	141,442	5,000	5,000

Table 2: Dataset statistics.

6 Experiment

6.1 Knowledge Graph Completion

To evaluate their performance for link prediction on knowledge graphs, we compared our proposed methods (NTN-Diag and NTN-Comp) to baseline methods (NTN (Socher et al., 2013a) and NTN-SMD).

Task

Let $\mathcal{E}$ and $\mathcal{R}$ denote entities and relations, respectively. A *relational triplet*, or simply a *triplet*, (s, r, o) is a triple with $s, o \in \mathcal{E}$ and $r \in \mathcal{R}$. It represents a proposition that relation r holds between subject entity s and object entity o. A triplet is called a fact if the proposition it denote is true. A *knowledge graph* is a collection of knowledge triplets, with the understanding that all its member triplets are facts. It is called a graph because each triplet can be regarded as an edge in a directed graph; the vertices in this graph represent entities in $\mathcal{E}$, and each edge is labeled by a relation in $\mathcal{R}$. Let $\mathcal{G}$ be a knowledge graph, viewed as a collection of facts. *Knowledge graph completion* (KGC) is the task of predicting whether unknown triplet $(s', r', o') \notin \mathcal{G}$ such that $s', o' \in \mathcal{E}, r' \in \mathcal{R}$ is a fact or not.

Models and Loss Function

The standard approach to KGC is to design a score function $\Phi : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \to \mathbb{R}$ that assigns a large value when a triplet seems to be a fact. Socher et al. (2013a) defined it as follows.

$$\boldsymbol{u}_r^{\mathrm{T}} f\left(\boldsymbol{e}_s^{\mathrm{T}}\boldsymbol{W}_r^{[1:k]}\boldsymbol{e}_o + \boldsymbol{V}_r \begin{bmatrix} \boldsymbol{e}_s \\ \boldsymbol{e}_o \end{bmatrix} + \boldsymbol{b}_r\right)$$

Here, $\boldsymbol{e}_s, \boldsymbol{e}_o \in \mathbb{R}^n$ are entity embeddings and $\boldsymbol{W}_r, \boldsymbol{V}_r, \boldsymbol{b}_r, \boldsymbol{u}_r$ are parameters for each relation r. $\boldsymbol{u}_r$ is a k-dimensional vector to map f's output $\mathbb{R}^k$ to $\mathbb{R}$ which indicates a score. f is the hyperbolic tangent. To compare the performances of the baselines and proposed models, we change the mapping before an activation. For NTN-SMD, we change term $\boldsymbol{e}_s^{\mathrm{T}}\boldsymbol{W}_r^{[1:k]}\boldsymbol{e}_o$ to $\boldsymbol{e}_s^{\mathrm{T}}\boldsymbol{S}_r^{[1:k]}\boldsymbol{T}_r^{[1:k]}\boldsymbol{e}_o$. To apply NTN-Diag and NTN-Comp in this model,

	WN18					FB15K				
	MRR		Hits@			MRR		Hits@		
model	Filter	Raw	1	3	10	Filter	Raw	1	3	10
NN	0.111	0.106	7.0	11.7	18.3	0.259	0.165	17.9	28.1	41.7
NTN ($k = 1$)	0.740	0.512	67.6	78.4	85.2	0.347	0.188	24.1	39.3	55.2
NTN ($k = 4$)	0.754	0.530	69.3	79.5	86.3	0.380	0.198	27.1	43.0	59.2
NTN-SMD ($m = 1$)	0.243	0.216	15.9	26.1	40.9	0.278	0.172	19.3	30.1	44.7
NTN-SMD ($m = 2$)	0.224	0.199	15.1	23.8	37.2	0.298	0.177	20.7	32.7	47.8
NTN-SMD ($m = 3$)	0.299	0.255	20.4	32.4	49.2	0.312	0.183	21.7	34.5	49.9
NTN-SMD ($m = 10$)	0.533	0.413	42.2	59.4	74.5	0.333	0.188	22.8	37.5	53.8
NTN-SMD ($m = 25$)	0.618	0.463	52.1	67.8	80.0	0.341	0.187	23.2	38.6	55.5
NTN-Diag	0.824	0.590	74.8	89.6	92.7	0.443	0.238	31.5	51.2	68.5
NTN-Comp	**0.857**	**0.610**	**80.1**	**90.9**	**93.1**	**0.490**	**0.246**	**36.3**	**56.7**	**71.9**
DistMult*	0.822	0.532	72.8	91.4	93.6	0.654	0.242	54.6	73.3	82.4
ComplEx*	0.941	0.587	93.6	94.5	94.7	0.692	0.242	59.9	75.9	84.0

Table 3: Mean Reciprocal Rank (MRR) and Hits@n for the models tested on WN18 and FB15k. MRR is reported in the raw and filtered settings. Hits@n metrics are percentages of test examples that lie in the top n ranked results. We report Hits@n in the filtered setting. *Results are those in (Trouillon et al., 2016)

we assume all slice matrices of tensors among relations form a commuting family. The loss function used to train the models is shown below:

$$\sum_{i=1}^{N}\sum_{c=1}^{C}\max\left(0, 1-\Phi\left(T^{(i)}\right)+\Phi\left(T_c^{(i)}\right)\right) + \lambda\|\mathbf{\Omega}\|_2^2,$$

where $\lambda\|\mathbf{\Omega}\|_2^2$ is an L2 regularization term, $T^{(i)}$ denotes the i-th example of training data of size N, and $T_c^{(i)}$ is one of C randomly sampled negative examples for the i-th training example. We generated negative samples of a triplet (s, r, o) by corrupting its subject or object entity.

Experimental Setup

We used the Wordnet (WN18) and Freebase (FB15k) datasets to verify the benefits of our proposed methods. The dataset statistics are given in Table 2. We selected hyper-parameters based on Socher et al. (2013a) and Yang et al. (2015): For all of the models, the size of mini-batches was set to 1000, the dimensionality of the entity vector to $d = 100$, and the regularization parameter to 0.0001; the tensor slice size was set to $k = 4$ for all models, except NTN for which we also tested with $k = 1$ to see the influence of the slice size on the performance. We performed 300 epochs of training for Wordnet and 100 on Freebase using Adagrad (Duchi et al., 2011) with the initial learning rate set to 0.1.

For evaluation, we removed the subject or object entity of each test example and then replaced it with all the entities in $\mathcal{E}$. We computed the scores of these corrupted triplets and ranked them in descending order of scores. We here report the results collected in filtered and raw settings. In the filtered setting, given test example (s, r, o), we remove from the ranking all the other positive triplets that appear in either training, validation, or test dataset, whereas the raw metrics do not remove these triplets.

Result

Experimental results are shown in Table 3. We observe the following:

- The performance of NN and NTNs differs considerably; Apparently, NN is inadequate for this task.
- By comparing the results of NTNs with different slice sizes, we see that $k = 4$ performs better than $k = 1$.
- NTN-SMDs perform better than NN, but are all inferior to NTNs, although their results improved as m (the rank of decomposed matrices) is increased.
- NTN-Diag achieved better results than NTN, although it has far fewer parameters than NTN and the datasets contain many unsymmetrical triplets. This demonstrates that NTN-Diag solves the overfitting problem of NTN without sacrificing the expressiveness power. NTN-Diag also has fewer parameters than the smallest ($m = 1$) NTN-SMD. Thus,

Conjunctive normal form	$\bigwedge_{i=1}^{m} \bigvee_{j=1}^{n_i} A_{ij}$
Disjunctive normal form	$\bigvee_{i=1}^{m} \bigwedge_{j=1}^{n_i} A_{ij}$

Table 4: Conjunctive and disjunctive normal forms in propositional logic. A_{ij} is a *literal*, which is a propositional variable or its negation. For example, p_1 and $\neg p_2$ are literal, but not $\neg\neg p_3$.

Name	Symbol	Set-theoretic definition
Entailment	$A \sqsubset B$	$A \subset B$
Reverse entailment	$A \sqsupset B$	$A \supset B$
Equivalence	$A \equiv B$	$A = B$
Alternation	$A \mid B$	$A \cap B = \emptyset \wedge A \cup B \neq \mathcal{D}$
Negation	$A \wedge B$	$A \cap B = \emptyset \wedge A \cup B = \mathcal{D}$
Cover	$A \smile B$	$A \cap B \neq \emptyset \wedge A \cup B = \mathcal{D}$
Independence	$A \# B$	else

Table 5: Natural logic relations over formula pairs. A and B denote a formula in propositional logic.

$not\ p_3$	$\wedge$	p_3
p_3	$\sqsubset$	$(p_3\ or\ p_2)$
$(p_1\ or (p_2\ or\ p_4)))$	$\sqsupset$	$(p_2\ and\ not\ p_4)$

Table 6: Short examples of type of formulas and their relations in datasets.

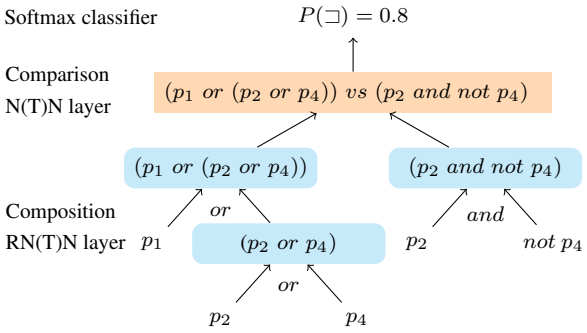


Figure 1: Comparison and composition layers. $not\ p_4$ is treated as an embedding.

we conclude that NTN-Diag is a better alternative of NTN than NTN-SMD is, in terms of both accuracy and computational cost.

- NTN-Comp outperformed NTN-Diag, showing that its flexible constraint on matrices yielded additional expressiveness. However, NTN-Diag and NTN-Comp do not exceed DistMult and ComplEx, respectively, in almost all measures.

Although not shown in the table, in this experiment, NTN-Diag and NTN-Comp was, respectively, 3 and 1.7 times as fast as NTN to train.

6.2 Logical Reasoning

To validate the performance of our proposed models in a recursive neural network setting, we experimentally tested them by having them solve a semantic compositionality problem in logic.

Task

This task definition basically follows Bowman et al. (2015): Given a pair of artificially generated propositional logic formulas, classify the relation between the formulas into one of the seven *basic semantic relations* of natural logic (MacCartney and Manning, 2009). Table 5 shows these seven relation types. The formulas consist of propositional variables, negation, and conjunction and disjunction connectives. Although Bowman et al. (2015) generated formulas with no constraint on its form, we restricted them to disjunctive normal form (DNF) or conjunctive normal form (CNF) (Table 4). Recall that any propositional formula can be transformed into these forms.

Models and Loss Function

Following Bowman et al. (2015), we constructed a model that infers the relations between formula pairs, as described in Table 6.

The model consists of two layers: composition and comparison layers (Figure 1). The composition layer outputs the embeddings of both left and right formulas by recursive neural networks. Subsequently, the comparison layer compares the two embeddings using a single layer neural network, and then a softmax classifier receives its output. In the composition layer, we set different parameters for *and* and *or* operations. As a loss function, we used cross entropy with L2 regularization and apply the NTNs in Section 4 to the comparison layer and uses RNTNs for as the composition layer.

Experimental Setup

In this experiment, an example is a pair of propositional formulas, and its class label is the seven relation types between the pair. We generated examples following the protocol described in Bowman et al. (2015), with the exception that the formulas are restricted to CNF or DNF, as mentioned above. We obtained 62,589 training examples, 13,413 validation examples, and 55,150 test examples. Each formula in the training and validation examples contains up to four logical operators, whereas those in the test examples have

Model	1	2	3	4	5	6	7	8	9	10	11	12	Avg.
Majority class	56.0	53.0	53.4	53.2	55.9	56.5	56.5	57.8	56.5	57.7	56.8	59.9	56.1
RNN	98.0	97.5	95.5	93.3	89.9	86.1	82.8	79.9	74.8	73.2	71.8	71.7	84.5
RNTN	**99.9**	**99.5**	98.2	95.7	92.7	88.5	84.7	81.2	78.1	77.5	74.4	74.4	87.0
RNTN-SMD ($m = 1$)	93.7	92.5	90.9	89.1	86.9	84.1	81.7	79.8	76.1	75.7	75.3	75.1	83.4
RNTN-SMD ($m = 2$)	93.0	93.4	91.7	90.3	88.2	85.5	82.7	81.4	77.6	77.0	75.4	**75.8**	84.3
RNTN-SMD ($m = 4$)	90.2	90.3	89.4	87.6	86.0	83.6	81.2	79.6	76.5	75.2	74.6	75.7	82.4
RNTN-SMD ($m = 8$)	86.8	84.9	83.5	82.5	81.1	79.1	76.6	75.6	72.4	71.3	70.9	71.2	77.9
RNTN-SMD ($m = 16$)	86.6	83.9	82.4	81.4	80.2	78.6	76.5	75.5	73.1	72.7	72.2	73.3	78.0
RNTN-Diag	**99.9**	98.9	**98.5**	**97.4**	**94.9**	**91.5**	**87.6**	**85.0**	**80.3**	**78.5**	**77.1**	75.2	**88.7**
RNTN-Comp	99.3	98.1	98.0	96.9	94.3	90.6	86.1	83.5	79.2	76.6	74.5	74.6	87.6

Table 7: Result of logical inference for Tests 1–12. Example in Test n has n logical operators in either or both left and right formulas. Each score is the average accuracy of five trials of the λ that achieved best performance on validation set. "Majority class" denotes the ratio of the majority class (relation "#", i.e., Independence; see Table 5).

Model	Accuracy	(Std. Dev.)
RNN	95.0	(0.8)
RNTN	97.2	(0.4)
RNTN-SMD ($m = 1$)	90.1	(3.4)
RNTN-SMD ($m = 2$)	91.4	(4.6)
RNTN-SMD ($m = 4$)	88.6	(7.1)
RNTN-SMD ($m = 8$)	82.7	(10.2)
RNTN-SMD ($m = 16$)	81.8	(11.7)
RNTN-Diag	98.1	(0.1)
RNTN-Comp	97.5	(0.1)

Table 8: Average accuracy and standard deviation on the validation dataset. The reported values are average over the best-performing model λ in each method.

up to 12 logical operators. Every formula consists of up to four variables taken from six propositional variables that are shared among all the examples. Hyperparameters and optimization are based on Bowman et al. (2015): Embedding size $d = 25$ (for RNN, $d = 45$) and the output size of comparison layer is $k = 75$, and we used AdaDelta (Zeiler, 2012) for an optimizer. We searched for the best coefficient λ of L2 regularization in $\lambda \in \{0.0001, 0.0003, 0.0005, 0.0007, 0.0009, 0.001\}$, whereas Bowman et al. (2015) set λ to 0.001 for RNN and 0.0003 for RNTN.

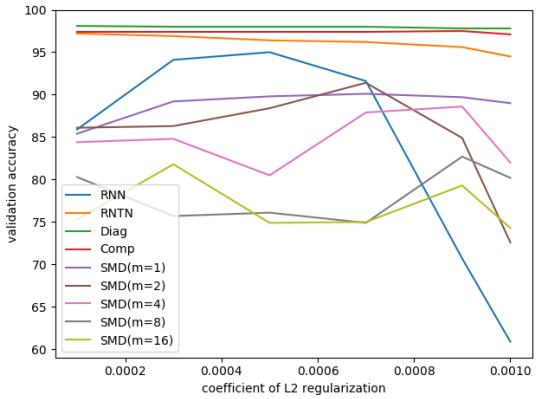
Result

The results are shown in Table 7. From the table, we observe the following:

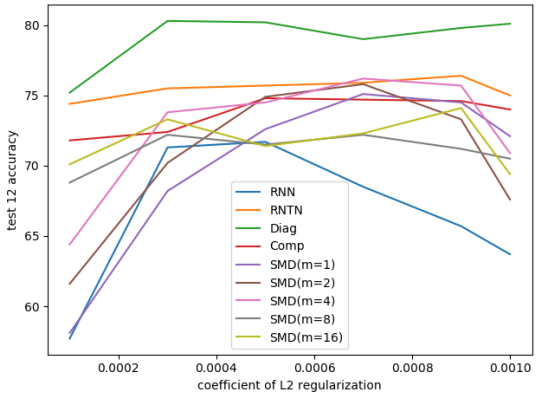
- As with KGC, the large difference in performance between RNN and RNTN suggests that this logical reasoning task requires feature interactions to be captured[1].
- RNTN-Diag achieved the best accuracy except for Tests 2 and 12 and outperformed RNTN except for Test 2. This is not surprising because both *and* and *or* are symmetric: p_1 *and* p_2 equals p_2 *and* p_1. This matches the tensor term in RNTN-Diag which is symmetric with respect to $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$.
- RNTN-Comp was the second best except for Tests 1–3 and 10–12. For all tests, its accuracy was comparable with or superior to that of RNTN.
- RNTN-SMD ($m = 1$) was inferior to RNTN for most test sets, although some good results were observed with $m = 1, 2, 3$ on Tests 11 and 12. Indeed, except for Tests 9–12, RNTN-SMD ($m = 1$) was inferior even to RNN despite the larger number of parameters in RNTN-SMD. RNTN-SMD ($m = 2$) obtained better results than $m = 1$, but it is still worse than RNTN except for Tests 10-12. Further increase in m ($m = 4, 8, 16$) worsened the accuracy despite an increase of the number of parameters.

We also evaluated the stability of the model over different trials and hyperparameters. Table 8 shows the best average accuracy for each compared model (among all the tested λ) on the validation set. The parenthesized figures (on the rightmost column) show the standard deviation over five independent trials used for computing the average, i.e., all five trials used the same λ value that achieved the best average accuracy. We see that RNTN-SMDs have larger standard deviations than

[1] Bowman (2016) also evaluated TreeLSTM, but its advantage over RNN was unclear in their experiment. For that reason, we did not test TreeLSTM in this paper.



(a) Validation set.



(b) Test 12.

Figure 2: Sensitivity of accuracy to λ.

RNTN, RNTN-Diag and RNTN-Comp. This indicates that RNTN-SMD is a less reliable model.

RNTN-SMDs are also unstable, not only within the same λ, but also between different λs. Figure 2 describes how accuracies are impacted by λs. The top graph shows validation accuracies between different λ values. RNTN, RNTN-Diag and RNTN-Comp are stable, whereas RNN and RNTN-SMDs have steep drops. The bottom one describes the accuracies for Test 12. This also shows that RNTN-SMDs are unstable and that RNTN-Diag achieves distinctive performances.

Finally, Figure 3 shows that training times increase quadratically with dimension for RNTN that has $O(n^2k)$ parameters, but not for our methods, which have only $O(nk)$ parameters.

7 Conclusion

We proposed two new parameter reduction methods for tensors in NTNs. The first method constrains the slice matrices to be symmetric, and the second assumes them to be normal matrices. In both methods, the number of a 3D tensor parameters is reduced from $O(n^2k)$ to $O(nk)$ after the constrained matrices are eigendecomposed. By removing the tensor's surplus parameters, our methods learn better and faster as was shown in experiments.[2] Future work will test the versatility of our proposals, RNTN-Diag and RNTN-Comp, in other tasks that deal with data sets exhibiting carious structures.

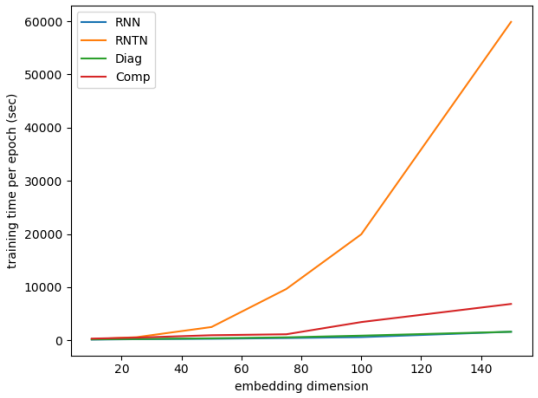


Figure 3: Training times of the models.

References

Bing Bai, Jason Weston, David Grangier, Ronan Collobert, Kunihiko Sadamasa, Yanjun Qi, Corinna Cortes, and Mehryar Mohri. 2009. Polynomial semantic indexing. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*. Curran Associates, Inc., pages 64–72.

Samuel R. Bowman. 2016. *Modeling natural language semantics in learned representations*. Ph.D. thesis, Stanford University.

Samuel R Bowman, Christopher Potts, and Christopher D Manning. 2015. Recursive neural networks can learn logical semantics. *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality* .

Wenlin Chen, James Wilson, Stephen Tyree, Kilian Weinberger, and Yixin Chen. 2015. Compressing neural networks with the hashing trick. In *Proceedings of the 32nd International Conference on Machine Learning*. pages 2285–2294.

Yu Cheng, Felix X Yu, Rogerio S Feris, Sanjiv Kumar, Alok Choudhary, and Shi-Fu Chang. 2015. An exploration of parameter redundancy in deep networks with circulant projections. In *Proceedings of the IEEE International Conference on Computer Vision*. pages 2857–2865.

[2] Code of the two experiments will be available at https://github.com/tkhrshhr

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation pages 1724–1734.

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

Katsuhiko Hayashi and Masashi Shimbo. 2017. On the equivalence of holographic and complex embeddings for link prediction. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics* pages 554–559.

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

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Hanxiao Liu, Yuexin Wu, and Yiming Yang. 2017. Analogical inference for multi-relational embeddings. In *Proceedings of the 34th International Conference on Machine Learning*. pages 2168–2178.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2015. Learning context-sensitive word embeddings with neural tensor skip-gram model. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*. pages 1284–1290.

Zhiyun Lu, Vikas Sindhwani, and Tara N Sainath. 2016. Learning compact recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, pages 5960–5964.

Bill MacCartney and Christopher D Manning. 2009. An extended model of natural logic. In *Proceedings of the eighth international conference on computational semantics*. Association for Computational Linguistics, pages 140–156.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th international conference on machine learning*. pages 809–816.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013a. Reasoning with neural tensor networks for knowledge base completion. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*. pages 926–934.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*. pages 1631–1642.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on Machine Learning*. pages 2071–2080.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. *International Conference on Learning Representations* .

Matthew D Zeiler. 2012. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

Yu Zhao, Zhiyuan Liu, and Maosong Sun. 2015. Phrase type sensitive tensor indexing model for semantic composition. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. pages 2195–2202.

Post-Specialisation: Retrofitting Vectors of Words Unseen in Lexical Resources

Ivan Vulić[1], **Goran Glavaš**[2], **Nikola Mrkšić**[3], **Anna Korhonen**[1]
[1] Language Technology Lab, University of Cambridge
[2] Data and Web Science Group, University of Mannheim
[3] PolyAI
{iv250,alk23}@cam.ac.uk
goran@informatik.uni-mannheim.de nikola@poly-ai.com

Abstract

Word vector *specialisation* (also known as *retrofitting*) is a portable, light-weight approach to fine-tuning arbitrary distributional word vector spaces by *injecting* external knowledge from rich lexical resources such as WordNet. By design, these *post-processing* methods only update the vectors of words occurring in external lexicons, leaving the representations of all *unseen* words intact. In this paper, we show that constraint-driven vector space specialisation can be extended to unseen words. We propose a novel *post-specialisation* method that: **a)** preserves the useful linguistic knowledge for seen words; while **b)** propagating this external signal to unseen words in order to improve their vector representations as well. Our post-specialisation approach explicits a non-linear specialisation function in the form of a deep neural network by learning to predict specialised vectors from their original distributional counterparts. The learned function is then used to specialise vectors of unseen words. This approach, applicable to any post-processing model, yields considerable gains over the initial specialisation models both in intrinsic word similarity tasks, and in two downstream tasks: *dialogue state tracking* and *lexical text simplification*. The positive effects persist across three languages, demonstrating the importance of specialising the full vocabulary of distributional word vector spaces.

1 Introduction

Word representation learning is a key research area in current Natural Language Processing (NLP), with its usefulness demonstrated across a range of tasks (Collobert et al., 2011; Chen and Manning, 2014; Melamud et al., 2016b). The standard techniques for inducing distributed word representations are grounded in the distributional hypothesis (Harris, 1954): they rely on co-occurrence information in large textual corpora (Mikolov et al., 2013b; Pennington et al., 2014; Levy and Goldberg, 2014; Levy et al., 2015; Bojanowski et al., 2017). As a result, these models tend to coalesce the notions of semantic similarity and (broader) conceptual relatedness, and cannot accurately distinguish antonyms from synonyms (Hill et al., 2015; Schwartz et al., 2015). Recently, we have witnessed a rise of interest in representation models that move beyond stand-alone unsupervised learning: they leverage external knowledge in human- and automatically-constructed lexical resources to enrich the semantic content of distributional word vectors, in a process termed *semantic specialisation*.

This is often done as a post-processing (sometimes referred to as *retrofitting*) step: input word vectors are *fine-tuned* to satisfy linguistic constraints extracted from lexical resources such as WordNet or BabelNet (Faruqui et al., 2015; Mrkšić et al., 2017). The use of external curated knowledge yields improved word vectors for the benefit of downstream applications (Faruqui, 2016). At the same time, this specialisation of the distributional space distinguishes between true similarity and relatedness, and supports language understanding tasks (Kiela et al., 2015; Mrkšić et al., 2017).

While there is consensus regarding their benefits and ease of use, one property of the post-processing specialisation methods slips under the radar: most existing post-processors update word embeddings only for words which are present (i.e., *seen*) in the external constraints, while vectors of all other (i.e., *unseen*) words remain unaffected. In this work, we propose a new approach that extends the specialisation framework to unseen words, relying on the transformation of the vector (sub)space of seen words. Our intuition is that the process of fine-tuning seen words provides implicit information on how to leverage the external knowledge to unseen words. The method should preserve the already injected knowledge for seen words, simultaneously

Proceedings of NAACL-HLT 2018, pages 516–527
New Orleans, Louisiana, June 1 - 6, 2018.

propagating the external signal to unseen words in order to improve their vectors.

The proposed *post-specialisation* method can be seen as a two-step process, illustrated in Fig. 1a: **1)** We use a state-of-the-art specialisation model to transform the subspace of seen words from the input distributional space into the specialised subspace; **2)** We learn a mapping function based on the transformation of the "seen subspace", and then apply it to the distributional subspace of unseen words. We allow the proposed post-specialisation model to learn from large external linguistic resources by implementing the mapping as a deep feed-forward neural network with non-linear activations. This allows the model to learn the generalisation of the fine-tuning steps taken by the initial specialisation model, itself based on a very large number (e.g., hundreds of thousands) of external linguistic constraints.

As indicated by the results on word similarity and two downstream tasks (dialogue state tracking and lexical text simplification) our post-specialisation method consistently outperforms state-of-the-art methods which specialise seen words only. We report improvements using three distinct input vector spaces for English and for three test languages (English, German, Italian), verifying the robustness of our approach.

2 Related Work and Motivation

Vector Space Specialisation A standard approach to incorporating external and background knowledge into word vector spaces is to pull the representations of similar words closer together and to push words in undesirable relations (e.g., antonyms) away from each other. Some models integrate such constraints into the training procedure and *jointly* optimize distributional and non-distributional objectives: they modify the prior or the regularisation (Yu and Dredze, 2014; Xu et al., 2014; Bian et al., 2014; Kiela et al., 2015), or use a variant of the SGNS-style objective (Liu et al., 2015; Ono et al., 2015; Osborne et al., 2016; Nguyen et al., 2017). In theory, word embeddings obtained by these joint models could be as good as representations produced by models which fine-tune input vector space. However, their performance falls behind that of fine-tuning methods (Wieting et al., 2015). Another disadvantage is that their architecture is tied to a specific underlying model (typically `word2vec` models).

In contrast, fine-tuning models inject external knowledge from available lexical resources (e.g., WordNet, PPDB) into pre-trained word vectors as a *post-processing step* (Faruqui et al., 2015; Rothe and Schütze, 2015; Wieting et al., 2015; Nguyen et al., 2016; Mrkšić et al., 2016; Cotterell et al., 2016; Mrkšić et al., 2017). Such post-processing models are popular because they offer a portable, flexible, and light-weight approach to incorporating external knowledge into *arbitrary* vector spaces, yielding state-of-the-art results on language understanding tasks (Faruqui et al., 2015; Mrkšić et al., 2016; Kim et al., 2016; Vulić et al., 2017b).

Existing post-processing models, however, suffer from a major limitation. Their *modus operandi* is to enrich the distributional information with external knowledge only if such knowledge is present in a lexical resource. This means that they update and improve only representations of words actually seen in external resources. Because such words constitute only a fraction of the whole vocabulary (see Sect. 4), most words, unseen in the constraints, retain their original vectors. The main goal of this work is to address this shortcoming by specialising *all* words from the initial distributional space.

3 Methodology: Post-Specialisation

Our starting point is the state-of-the-art specialisation model ATTRACT-REPEL (AR) (Mrkšić et al., 2017), outlined in Sect. 3.1. We opt for the AR model due to its strong performance and ease of use, but we note that the proposed post-specialisation approach for specialising unseen words, described in Sect. 3.2, is applicable to any post-processor, as empirically validated in Sect. 5.

3.1 Initial Specialisation Model: AR

Let $\mathcal{V}_s$ be the vocabulary, A the set of synonymous ATTRACT word pairs (e.g., *rich* and *wealthy*), and R the set of antonymous REPEL word pairs (e.g., *increase* and *decrease*). The ATTRACT-REPEL procedure operates over mini-batches of such pairs $\mathcal{B}_A$ and $\mathcal{B}_R$. Let each word pair (x_l, x_r) in these sets correspond to a vector pair $(\mathbf{x}_l, \mathbf{x}_r)$. A mini-batch of b_{att} attract word pairs is given by $\mathcal{B}_A = [(\mathbf{x}_l^1, \mathbf{x}_r^1), \ldots, (\mathbf{x}_l^{k_1}, \mathbf{x}_r^{k_1})]$ (analogously for $\mathcal{B}_R$, which consists of b_{rep} pairs).

Next, the sets of negative examples $T_A = [(\mathbf{t}_l^1, \mathbf{t}_r^1), \ldots, (\mathbf{t}_l^{k_1}, \mathbf{t}_r^{k_1})]$ and $T_R = [(\mathbf{t}_l^1, \mathbf{t}_r^1), \ldots, (\mathbf{t}_l^{k_2}, \mathbf{t}_r^{k_2})]$ are defined as pairs of *negative examples* for each A and R

pair in mini-batches $\mathcal{B}_A$ and $\mathcal{B}_R$. These negative examples are chosen from the word vectors present in $\mathcal{B}_A$ or $\mathcal{B}_R$ so that, for each A pair $(\mathbf{x}_l, \mathbf{x}_r)$, the negative example pair $(\mathbf{t}_l, \mathbf{t}_r)$ is chosen so that $\mathbf{t}_l$ is the vector closest (in terms of cosine distance) to $\mathbf{x}_l$ and $\mathbf{t}_r$ is closest to $\mathbf{x}_r$.[1] The negatives are used **1)** to force A pairs to be closer to each other than to their respective negative examples; and **2)** to force R pairs to be further away from each other than from their negative examples. The first term of the cost function pulls A pairs together:

$$Att(\mathcal{B}_A, T_A) = \sum_{i=1}^{b_{att}} [\, \tau\left(\delta_{att} + \mathbf{x}_l^i \mathbf{t}_l^i - \mathbf{x}_l^i \mathbf{x}_r^i\right) + \tau\left(\delta_{att} + \mathbf{x}_r^i \mathbf{t}_r^i - \mathbf{x}_l^i \mathbf{x}_r^i\right)] \quad (1)$$

where $\tau(z) = \max(0, z)$ is the standard rectifier function (Nair and Hinton, 2010) and δ_{att} is the attract margin: it determines how much closer these vectors should be to each other than to their respective negative examples. The second, REPEL term in the cost function is analogous: it pushes R word pairs away from each other by the margin δ_{rep}.

Finally, in addition to the A and R terms, a regularisation term is used to preserve the semantic content originally present in the distributional vector space, as long as this information does not contradict the injected external knowledge. Let $\mathcal{V}(\mathcal{B})$ be the set of all word vectors present in a mini-batch, the distributional regularisation term is then:

$$Reg(\mathcal{B}_A, \mathcal{B}_R) = \sum_{\mathbf{x}_i \in V(\mathcal{B}_A \cup \mathcal{B}_R)} \lambda_{reg} \left\| \widehat{\mathbf{x}}_i - \mathbf{x}_i \right\|_2 \quad (2)$$

where λ_{reg} is the L_2-regularisation constant and $\widehat{\mathbf{x}}_i$ denotes the original (distributional) word vector for word x_i. The full ATTRACT-REPEL cost function is finally constructed as the sum of all three terms.

3.2 Specialisation of Unseen Words

Problem Formulation The goal is to learn a *global transformation function* that generalises the perturbations of the initial vector space made by ATTRACT-REPEL (or any other specialisation procedure), as conditioned on the external constraints. The learned function propagates the signal coded in the input constraints to all the words unseen during the specialisation process. We seek a regression function $f : \mathbb{R}^{dim} \rightarrow \mathbb{R}^{dim}$, where dim is the vector space dimensionality. It maps word vectors from the initial vector space $\mathbf{X}$ to the specialised target space $\mathbf{X}'$. Let $\widehat{\mathbf{X}'} = f(\mathbf{X})$ refer to the predicted mapping of the vector space, while the mapping of a single word vector is denoted $\widehat{\mathbf{x}'_i} = f(\mathbf{x}_i)$.

An input distributional vector space $\mathbf{X}_d$ represents words from a vocabulary $\mathcal{V}_d$. $\mathcal{V}_d$ may be divided into two vocabulary subsets: $\mathcal{V}_d = \mathcal{V}_s \cup \mathcal{V}_u$, $\mathcal{V}_s \cap \mathcal{V}_u = \emptyset$, with the accompanying vector subspaces $\mathbf{X}_d = \mathbf{X}_s \sqcup \mathbf{X}_u$. $\mathcal{V}_s$ refers to the vocabulary of seen words: those that appear in the external linguistic constraints and have their embeddings changed in the specialisation process. $\mathcal{V}_u$ denotes the vocabulary of unseen words: those not present in the constraints and whose embeddings are unaffected by the specialisation procedure.

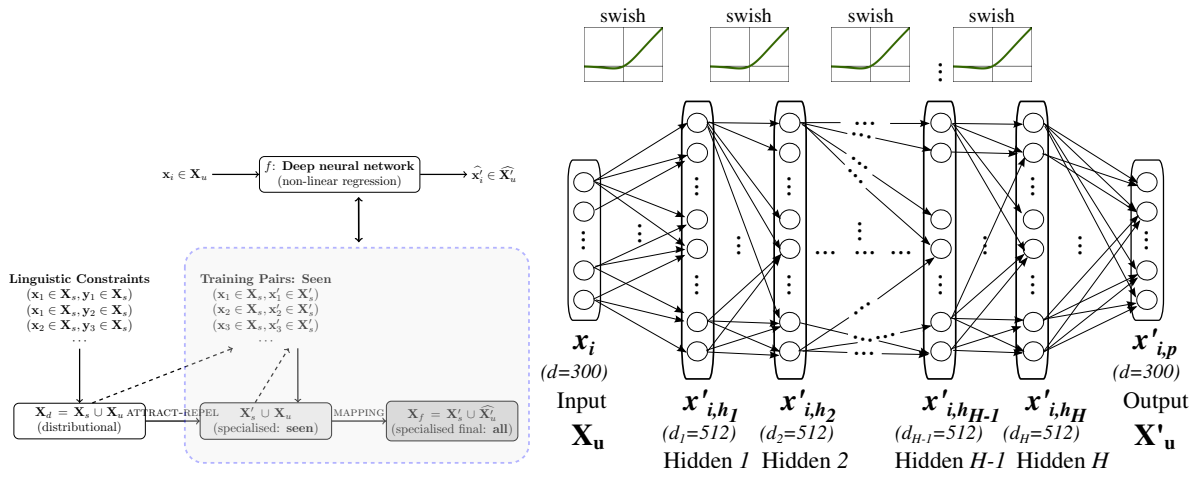
The AR specialisation process transforms only the subspace $\mathbf{X}_s$ into the specialised subspace $\mathbf{X}'_s$. All words $x_i \in \mathcal{V}_s$ may now be used as training examples for learning the explicit mapping function f from $\mathbf{X}_s$ into $\mathbf{X}'_s$. If $N = |\mathcal{V}_s|$, we in fact rely on N training pairs: $(\mathbf{x}_i, \mathbf{x}'_i) = \{\mathbf{x}_i \in \mathbf{X}_s, \mathbf{x}'_i \in \mathbf{X}'_s\}$. Function f can then be applied to unseen words $x \in \mathcal{V}_u$ to yield the specialised subspace $\widehat{\mathbf{X}'_u} = f(\mathbf{X}_u)$. The specialised space containing *all* words is then $\mathbf{X}_f = \mathbf{X}'_s \cup \widehat{\mathbf{X}'_u}$. The complete high-level post-specialisation procedure is outlined in Fig. 1a.

Note that another variant of the approach could obtain $\mathbf{X}_f$ as $\mathbf{X}_f = f(\mathbf{X}_d)$, that is, the entire distributional space is transformed by f. However, this variant seems counter-intuitive as it forgets the actual output of the initial specialisation procedure and replaces word vectors from $\mathbf{X}'_s$ with their approximations, i.e., f-mapped vectors.[2]

Objective Functions As mentioned, the N *seen* words $x_i \in \mathcal{V}_s$ in fact serve as our "pseudo-translation" pairs supporting the learning of a cross-space mapping function. In practice, in its high-level formulation, our mapping problem is equivalent to those encountered in the literature on cross-lingual word embeddings where the goal is to learn a shared cross-lingual space given monolingual vector spaces in two languages and N_1 translation pairs (Mikolov et al., 2013a; Lazaridou et al., 2015; Vulić and Korhonen, 2016b; Artetxe et al., 2016, 2017; Conneau et al., 2017; Ruder et al., 2017). In our setup, the standard objective based on L_2-penalised

[1] Similarly, for each R pair $(\mathbf{x}_l, \mathbf{x}_r)$, the negative pair $(\mathbf{t}_l, \mathbf{t}_r)$ is chosen from the in-batch vectors so that $\mathbf{t}_l$ is the vector furthest away from $\mathbf{x}_l$ and $\mathbf{t}_r$ is furthest from $\mathbf{x}_r$. All vectors are unit length (re)normalised after each epoch.

[2] We have empirically confirmed the intuition that the first variant is superior to this alternative. We do not report the actual quantitative comparison for brevity.



(a) High-level **illustration** (b) Low-level **implementation**: deep feed-forward neural network

Figure 1: **(a)** High-level illustration of the post-specialisation approach: the subspace $\mathbf{X}_s$ of the initial distributional vector space $\mathbf{X}_d = \mathbf{X}_s \cup \mathbf{X}_u$ is first specialised/fine-tuned by the ATTRACT-REPEL specialisation model (or any other post-processing model) to obtain the transformed subspace $\mathbf{X}'_s$. The words present (i.e., *seen*) in the input set of linguistic constraints are now assigned different representations in $\mathbf{X}_s$ (the original distributional vector) and $\mathbf{X}'_s$ (the specialised vector): they are therefore used as training examples to learn a non-linear cross-space mapping function. This function is then applied to all word vectors $\mathbf{x}_i \in \mathbf{X}_u$ representing words *unseen* in the constraints to yield a specialised subspace $\widehat{\mathbf{X}'_u}$. The final space is $\mathbf{X}_f = \mathbf{X}'_s \cup \widehat{\mathbf{X}'_u}$, and it contains transformed representations for *all* words from the initial space $\mathbf{X}_d$. **(b)** The actual implementation of the non-linear regression function which maps from $\mathbf{X}_u$ to $\widehat{\mathbf{X}'_u}$: a deep feed-forward fully-connected neural net with non-linearities and H hidden layers.

least squares may be formulated as follows:

$$f_{\text{MSE}} = \arg\min ||f(\mathbf{X}_s) - \mathbf{X}'_s||_F^2 \quad (3)$$

where $||\cdot||_F^2$ denotes the squared Frobenius norm. In the most common form $f(\mathbf{X}_s)$ is simply a linear map/matrix $\mathbf{W}_f \in \mathbb{R}^{dim \times dim}$ (Mikolov et al., 2013a) as follows: $f(\mathbf{X}) = \mathbf{W}_f\mathbf{X}$.

After learning f based on the $\mathbf{X}_s \rightarrow \mathbf{X}'_s$ transformation, one can simply apply f to unseen words: $\widehat{\mathbf{X}'_u} = f(\mathbf{X}_u)$. This linear mapping model, termed LINEAR-MSE, has an analytical solution (Artetxe et al., 2016), and has been proven to work well with cross-lingual embeddings. However, given that the specialisation model injects hundreds of thousands (or even millions) of linguistic constraints into the distributional space (see later in Sect. 4), we suspect that the assumption of linearity is too limiting and does not fully hold in this particular setup.

Using the same L_2-penalized least squares objective, we can thus replace the linear map with a *non-linear* function $f : \mathbb{R}^{dim} \rightarrow \mathbb{R}^{dim}$. The non-linear mapping, illustrated by Fig. 1b, is implemented as a deep feed-forward fully-connected neural network (DFFN) with H hidden layers and non-linear activations. This variant is called NONLINEAR-MSE.

Another variant objective is the contrastive margin-based ranking loss with negative sampling (MM) similar to the original ATTRACT-REPEL objective, used in other applications in prior work (e.g., for cross-modal mapping) (Weston et al., 2011; Frome et al., 2013; Lazaridou et al., 2015; Kummerfeld et al., 2015). Let $\widehat{\mathbf{x}'_i} = f(\mathbf{x_i})$ denote the predicted vector for the word $x_i \in \mathcal{V}_s$, and let $\mathbf{x}'_i$ refer to the "true" vector of x_i in the specialised space $\mathbf{X}'_s$ after the AR specialisation procedure. The MM loss is then defined as follows:

$$J_{\text{MM}} = \sum_{i=1}^{N}\sum_{j\neq i}^{k} \tau\Big(\delta_{mm} - cos\big(\widehat{\mathbf{x}'_i}, \mathbf{x}'_i\big) + cos\big(\widehat{\mathbf{x}'_i}, \mathbf{x}'_j\big)\Big)$$

where cos is the cosine similarity measure, δ_{mm} is the margin, and k is the number of negative samples. The objective tries to learn the mapping f so that each predicted vector $\widehat{\mathbf{x}'_i}$ is by the specified margin δ_{mm} closer to the correct target vector $\mathbf{x}'_i$ than to any other of k target vectors $\mathbf{x}'_j$ serving as negative examples.[3] Function f can again be either a simple linear map (LINEAR-MM), or implemented as a DFFN (NONLINEAR-MM, see Fig. 1b).

[3]We have also experimented with a simpler *hinge loss* function without negative examples, formulated as $J =$

4 Experimental Setup

Starting Word Embeddings ($\mathbf{X}_d = \mathbf{X}_s \cup \mathbf{X}_u$) To test the robustness of our approach, we experiment with three well-known, publicly available collections of English word vectors: **1)** Skip-Gram with Negative Sampling (SGNS-BOW2) (Mikolov et al., 2013b) trained on the Polyglot Wikipedia (Al-Rfou et al., 2013) by Levy and Goldberg (2014) using bag-of-words windows of size 2; **2)** GLOVE Common Crawl (Pennington et al., 2014); and **3)** FASTTEXT (Bojanowski et al., 2017), a SGNS variant which builds word vectors as the sum of their constituent character n-gram vectors. All word embeddings are 300-dimensional.[4]

AR Specialisation and Constraints ($\mathbf{X}_s \rightarrow \mathbf{X}'_s$) We experiment with linguistic constraints used before by (Mrkšić et al., 2017; Vulić et al., 2017a): they extracted monolingual synonymy/ATTRACT pairs from the Paraphrase Database (PPDB) (Ganitkevitch et al., 2013; Pavlick et al., 2015) (640,435 synonymy pairs in total), while their antonymy/REPEL constraints came from BabelNet (Navigli and Ponzetto, 2012) (11,939 pairs).[5]

The coverage of $\mathcal{V}_d$ vocabulary words in the constraints illustrates well the problem of unseen words with the fine-tuning specialisation models. For instance, the constraints cover only a small subset of the entire vocabulary $\mathcal{V}_d$ for SGNS-BOW2: 16.6%. They also cover only 15.3% of the top 200K most frequent $\mathcal{V}_d$ words from FASTTEXT.

Network Design and Parameters ($\mathbf{X}_u \rightarrow \widehat{\mathbf{X}'_u}$) The non-linear regression function $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$ is a DFFN with H hidden layers, each of dimensionality $d_1 = d_2 = \ldots = d_H = 512$ (see Fig. 1b). Non-linear activations are used in each layer and omitted only before the final output layer to enable full-range predictions (see Fig. 1b again).

The choices of non-linear activation and initialisation are guided by recent recommendations from the literature. First, we use *swish* (Ramachandran et al., 2017; Elfwing et al., 2017) as non-linearity, defined as $swish(x) = x \cdot \text{sigmoid}(\beta x)$. We fix $\beta = 1$ as suggested by Ramachandran et al. (2017).[6] Second, we use the HE normal initialisation (He et al., 2015), which is preferred over the XAVIER initialisation (Glorot and Bengio, 2010) for deep models (Mishkin and Matas, 2016; Li et al., 2016), although in our experiments we do not observe a significant difference in performance between the two alternatives. We set $H = 5$ in all experiments without any fine-tuning; we also analyse the impact of the network depth in Sect. 5.

Optimisation For the AR specialisation step, we adopt the original suggested model setup. Hyperparameter values are set to: $\delta_{att} = 0.6$, $\delta_{rep} = 0.0$, $\lambda_{reg} = 10^{-9}$ (Mrkšić et al., 2017). The models are trained for 5 epochs with Adagrad (Duchi et al., 2011), with batch sizes set to $b_{att} = b_{rep} = 50$, again as in the original work.

For training the non-linear mapping with DFFN (Fig. 1b), we use the Adam algorithm (Kingma and Ba, 2015) with default settings. The model is trained for 100 epochs with early stopping on a validation set. We reserve 10% of all available seen data (i.e., the words from $\mathcal{V}_s$ represented in $\mathbf{X}_s$ and $\mathbf{X}'_s$) for validation, the rest are used for training. For the MM objective, we set $\delta_{mm} = 0.6$ and $k = 25$ in all experiments without any fine-tuning.

5 Results and Discussion

5.1 Intrinsic Evaluation: Word Similarity

Evaluation Protocol The first set of experiments evaluates vector spaces with different specialisation procedures intrinsically on word similarity benchmarks: we use the SimLex-999 dataset (Hill et al., 2015), and SimVerb-3500 (Gerz et al., 2016), a recent verb pair similarity dataset providing similarity ratings for 3,500 verb pairs.[7] Spearman's ρ

$\sum_{i=1}^{N} \tau\left(\delta_{mm} - cos(\widehat{\mathbf{x}'_i}, \mathbf{x}'_i)\right)$. For instance, with $\delta_{mm} = 1.0$ the idea is to learn a mapping f that, for each x_i enforces the predicted vector and the correct target vector to have a maximum cosine similarity. We do not report the results with this variant as, although it outscores the MSE-style objective, it was consistently outperformed by the MM objective.

[4] For further details regarding the architectures and training setup of the used vector collections, we refer the reader to the original papers. Additional experiments with other word vectors, e.g., with CONTEXT2VEC (Melamud et al., 2016a) (which uses bidirectional LSTMs (Hochreiter and Schmidhuber, 1997) for context modeling), and with dependency-word based embeddings (Bansal et al., 2014; Melamud et al., 2016b) lead to similar results and same conclusions.

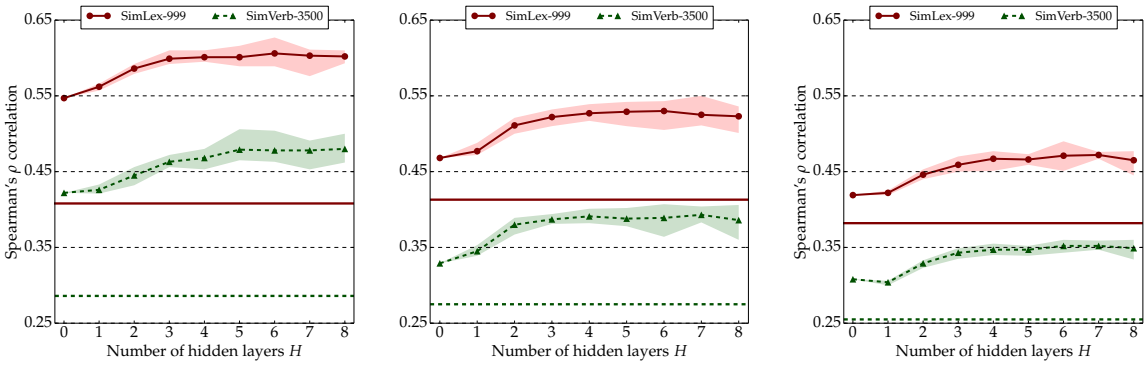
[5] We have experimented with another set of constraints used in prior work (Zhang et al., 2014; Ono et al., 2015), reaching similar conclusions: these were extracted from WordNet (Fellbaum, 1998) and Roget (Kipfer, 2009), and comprise 1,023,082 synonymy pairs and 380,873 antonymy pairs.

[6] According to Ramachandran et al. (2017), for deep networks *swish* has a slight edge over the family of LU/ReLU-related activations (Maas et al., 2013; He et al., 2015; Klambauer et al., 2017). We also observe a minor (and insignificant) difference in performance in favour of *swish*.

[7] While other gold standards such as WordSim-353 (Finkelstein et al., 2002) or MEN (Bruni et al., 2014) coalesce the notions of true semantic similarity and (more broad) conceptual relatedness, SimLex and SimVerb provide explicit guidelines

	Setup: *hold-out*						Setup: *all*					
	GLOVE		SGNS-BOW2		FASTTEXT		GLOVE		SGNS-BOW2		FASTTEXT	
	SL	SV	SL	SV	SL	SV	SL	SV	SL	SV	SL	SV
Distributional: $\mathbf{X}_d$	.408	.286	.414	.275	.383	.255	.408	.286	.414	.275	.383	.255
+AR specialisation: $\mathbf{X}'_s$	.408	.286	.414	.275	.383	.255	.690	.578	.658	.544	.629	.502
++Mapping unseen: $\mathbf{X}_f$												
LINEAR-MSE	.504	.384	.447	.309	.405	.285	.690	.578	.656	.551	.628	.502
NONLINEAR-MSE	.549	.407	.484	.344	.459	.329	.694	.586	.663	.556	.631	.506
LINEAR-MM	.548	.422	.468	.329	.419	.308	.697	.582	.663	.554	.628	.487
NONLINEAR-MM	**.603**	**.480**	**.531**	**.391**	**.471**	**.349**	**.705**	**.600**	**.667**	**.562**	**.638**	**.507**

Table 1: Spearman's ρ correlation scores for three word vector collections on two English word similarity datasets, SimLex-999 (SL) and SimVerb-3500 (SV), using different mapping variants, evaluation protocols, and word vector spaces: from the initial distributional space $\mathbf{X}_d$ to the fully specialised space $\mathbf{X}_f$. $H = 5$.



(a) GLOVE (b) SGNS-BOW2 (c) FASTTEXT

Figure 2: The results of the *hold-out* experiments on SimLex-999 and SimVerb-3500 after applying our non-linear vector space transformation with different depths (hidden layer size H, see Fig. 1b). The results are presented as averages over 20 runs with the NONLINEAR-MM variant, the shaded regions are spanned by the maximum and minimum scores obtained. Thick horizontal lines refer to Spearman's rank correlations achieved in the initial space $\mathbf{X}_d$. $H = 0$ denotes the standard *linear* regression model (Mikolov et al., 2013a; Lazaridou et al., 2015) (LINEAR-MM shown since it outperforms LINEAR-MSE).

rank correlation is used as the evaluation metric.

We evaluate word vectors in two settings. First, in a synthetic *hold-out* setting, we remove all linguistic constraints which contain words from the SimLex *and* SimVerb evaluation data, effectively forcing all SimLex and SimVerb words to be *unseen* by the AR specialisation model. The specialised vectors for these words are estimated by the learned non-linear DFFN mapping model. Second, the *all* setting is a standard "real-life" scenario where some test (SimLex/SimVerb) words do occur in the constraints, while the mapping is learned for the remaining words.

Results and Analysis The results with the three word vector collections are provided in Tab. 1. In addition, Fig. 2 plots the influence of the network depth H on the model's performance.

The results suggest that the mapping of unseen words is universally useful, as the highest correlation scores are obtained with the final fully specialised vector space $\mathbf{X}_f$ for all three input spaces. The results in the *hold-out* setup are particularly indicative of the improvement achieved by our post-specialisation method. For instance, it achieves a +0.2 correlation gain with GLOVE on both SimLex and SimVerb by specialising vector representations for words present in these datasets *without* seeing a single external constraint which contains any of these words. This suggests that the perturbation of the seen subspace $\mathbf{X}_s$ by ATTRACT-REPEL contains implicit knowledge that can be propagated to $\mathbf{X}_u$, learning better representations for unseen words. We observe small but consistent improvements across the board in the *all* setup. The smaller gains can be explained by the fact that a majority of

to discern between the two, so that related but non-similar words (e.g. *tiger* and *jungle*) have a low rating.

SimLex and SimVerb words are present in the external constraints (93.7% and 87.2%, respectively).

The scores also indicate that *both* non-linearity and the chosen objective function contribute to the quality of the learned mapping: largest gains are reported with the NONLINEAR-MM variant which **a)** employs non-linear activations and **b)** replaces the basic mean-squared-error objective with max-margin. The usefulness of the latter has been established in prior work on cross-space mapping learning (Lazaridou et al., 2015). The former indicates that the initial AR transformation is non-linear. It is guided by a large number of constraints; their effect cannot be captured by a simple linear map as in prior work on, e.g., cross-lingual word embeddings (Mikolov et al., 2013a; Ruder et al., 2017).

Finally, the analysis of the network depth H indicates that going deeper helps only to a certain extent. Adding more layers allows for a richer parametrisation of the network (which is beneficial given the number of linguistic constraints used by AR). This makes the model more expressive, but it seems to saturate with larger H values.

Post-Specialisation with Other Post-Processors We also verify that our post-specialisation approach is not tied to the ATTRACT-REPEL method, and is indeed applicable on top of any post-processing specialisation method. We analyse the impact of post-specialisation in the *hold-out* setting using the original *retrofitting* (*RFit*) model (Faruqui et al., 2015) and *counter-fitting* (*CFit*) (Mrkšić et al., 2016) in lieu of attract-repel. The results on word similarity with the best-performing NONLINEAR-MM variant are summarised in Tab. 2.

The scores again indicate the usefulness of post-specialisation. As expected, the gains are lower

Figure 3: DST labels (user goals given by slot-value pairs) in a multi-turn dialogue (Mrkšić et al., 2015).

	GLOVE		SGNS-BOW2		FASTTEXT	
	SL	SV	SL	SV	SL	SV
$\mathbf{X}_d$	.408	.286	.414	.275	.383	.255
$\mathbf{X}_f$: *RFit*	.493	.365	.412	.285	.413	.279
$\mathbf{X}_f$: *CFit*	.540	.401	.439	.318	.306	.441

Table 2: Post-specialisation applied to two other post-processing methods. SL: SimLex; SV: SimVerb. *Hold-out* setting. NONLINEAR-MM.

than with ATTRACT-REPEL. *RFit* falls short of *CFit* as by design it can leverage only synonymy (i.e., ATTRACT) external constraints.

5.2 Downstream Task I: DST

Next, we evaluate the usefulness of post-specialisation for two downstream tasks – dialogue state tracking and lexical text simplification – in which discerning semantic similarity from other types of semantic relatedness is crucial. We first evaluate the importance of post-specialisation for a downstream language understanding task of *dialogue state tracking* (DST) (Henderson et al., 2014; Williams et al., 2016), adopting the evaluation protocol and data of Mrkšić et al. (2017).

DST: Model and Evaluation The DST model is the first component of modern dialogue pipelines (Young, 2010), which captures the users' goals at each dialogue turn and then updates the dialogue state. Goals are represented as sets of constraints expressed as slot-value pairs (e.g., food=*Chinese*). The set of slots and the set of values for each slot constitute the ontology of a dialogue domain. The probability distribution over the possible states is the system's estimate of the user's goals, and it is used by the dialogue manager module to select the subsequent system response (Su et al., 2016). An example in Fig. 3 illustrates the DST pipeline.

For evaluation, we use the Neural Belief Tracker (NBT), a state-of-the-art DST model which was the first to reason purely over pre-trained word vectors (Mrkšić et al., 2017).[8] The NBT uses no hand-crafted semantic lexicons, instead composing word vectors into intermediate utterance and context representations.[9] For full model details, we refer the reader to the original paper. The importance of word vector specialisation for the DST task (e.g., distinguishing between synonyms and antonyms by pulling *northern* and *north* closer in

[8] https://github.com/nmrksic/neural-belief-tracker

[9] The NBT keeps word vectors *fixed* during training to enable generalisation for words unseen in DST training data.

ENGLISH	*hold-out*	*all*
Distributional: $\mathbf{X}_d$	.797	.797
+AR Spec.: $\mathbf{X}'_s \cup \mathbf{X}_u$	.797	.817
++Mapping: $\mathbf{X}_f = \mathbf{X}'_s \cup \widehat{\mathbf{X}'_u}$		
LINEAR-MM	.815	.818
NONLINEAR-MM	**.827**	**.835**

Table 3: DST results in two evaluation settings (*hold-out* and *all*) with different GLOVE variants.

the vector space while pushing *north* and *south* away) has been established (Mrkšić et al., 2017).

Again, as in prior work the DST evaluation is based on the Wizard-of-Oz (WOZ) v2.0 dataset (Wen et al., 2017; Mrkšić et al., 2017), comprising 1,200 dialogues split into training (600 dialogues), development (200), and test data (400). In all experiments, we report the standard DST performance measure: *joint goal accuracy*, and report scores as averages over 5 NBT training runs.

Results and Analysis We again evaluate word vectors in two settings: **1)** *hold-out*, where linguistic constraints with words appearing in the WOZ data are removed, making all WOZ words unseen by ATTRACT-REPEL; and **2)** *all*. The results for the English DST task with different GLOVE word vector variants are summarised in Tab. 3; similar trends in results are observed with two other word vector collections. The scores maintain conclusions established in the word similarity task. First, semantic specialisation with ATTRACT-REPEL is again beneficial, and discerning between synonyms and antonyms improves DST performance. However, specialising unseen words (the final $\mathbf{X}_u$ vector space) yields further improvements in both evaluation settings, supporting our claim that the specialisation signal can be propagated to unseen words.

This downstream evaluation again demonstrates the importance of non-linearity, as the peak scores are reported with the NONLINEAR-MM variant. More substantial gains in the *all* setup are observed in the DST task compared to the word similarity task. This stems from a lower coverage of the WOZ data in the AR constraints: 36.3% of all WOZ words are unseen words. Finally, the scores are higher on average in the *all* setup, since this setup uses more external constraints for AR, and consequently uses more training examples to learn the mapping.

Other Languages We test the portability of our framework to two other languages for which we have similar evaluation data: German (DE) and Italian (IT). SimLex-999 has been translated and rescored in the two languages by Leviant and Reichart (2015), and the WOZ data were translated and adapted by Mrkšić et al. (2017). Exactly the same setup is used as in our English experiments, without any additional language-specific fine-tuning. Linguistic constraints were extracted from the same sources: synonyms from the PPDB (135,868 in DE, 362,452 in IT), antonyms from BabelNet (4,124 in DE, and 16,854 in IT). Our starting distributional vector spaces are taken from prior work: IT vectors are from (Dinu et al., 2015), DE vectors are from (Vulić and Korhonen, 2016a). The results are summarised in Tab. 4.

Our post-specialisation approach yields consistent improvements over the initial distributional space and the AR specialisation model in both tasks and for both languages. We do not observe any gain on IT SimLex in the *all* setup since IT constraints have almost complete coverage of all IT SimLex words (99.3%; the coverage is 64.8% in German). As expected, the DST scores in the *all* setup are higher than in the *hold-out* setup due to a larger number of constraints and training examples.

Lower absolute scores for Italian and German compared to the ones reported for English are due to multiple factors, as discussed previously by Mrkšić et al. (2017): **1)** the AR model uses less linguistic constraints for DE and IT; **2)** distributional word vectors are induced from smaller corpora; **3)** linguistic phenomena (e.g., cases and compounding in DE) contribute to data sparsity and also make the DST task more challenging. However, it is important to stress the consistent gains over the vector space specialised by the state-of-the-art ATTRACT-REPEL model across all three test languages. This indicates that the proposed approach is language-agnostic and portable to multiple languages.

5.3 Downstream Task II: Lexical Simplification

In our second downstream task, we examine the effects of post-specialisation on lexical simplification (LS) in English. LS aims to substitute complex words (i.e., less commonly used words) with their simpler synonyms in the context. Simplified text must keep the meaning of the original text, which is discerning similarity from relatedness is important (e.g., in *"The automobile was set on fire"* the word *"automobile"* should be replaced with *"car"* or *"vehicle"* but not with *"wheel"* or *"driver"*).

	GERMAN				ITALIAN			
	SimLex (Similarity)		WOZ (DST)		SimLex (Similarity)		WOZ (DST)	
	hold-out	*all*	*hold-out*	*all*	*hold-out*	*all*	*hold-out*	*all*
Distributional: $\mathbf{X}_d$	.267	.267	.487	.487	.363	.363	.618	.618
+AR Spec.: $\mathbf{X}'_s \cup \mathbf{X}_u$	.267	.422	.487	.535	.363	**.616**	.618	.634
++Mapping: $\mathbf{X}_f$								
LINEAR-MM	.354	.449	.485	.533	.401	**.616**	.627	.633
NONLINEAR-MM	**.367**	**.466**	**.496**	**.538**	**.428**	**.616**	**.637**	**.647**

Table 4: Results on word similarity (Spearman's ρ) and DST (joint goal accuracy) for German and Italian.

Vectors	Specialisation	Acc.	Ch.
GLOVE	**Distributional:** $\mathbf{X}_d$	66.0	**94.0**
	+AR Spec.: $\mathbf{X}'_s \cup \mathbf{X}_u$	67.6	87.0
	++Mapping: $\mathbf{X}_f$	**72.3**	87.6
FASTTEXT	**Distributional:** $\mathbf{X}_d$	57.8	84.0
	+AR Spec.: $\mathbf{X}'_s \cup \mathbf{X}_u$	69.8	**89.4**
	++Mapping: $\mathbf{X}_f$	**74.3**	88.8
SGNS-BOW2	**Distributional:** $\mathbf{X}_d$	56.0	79.1
	+AR Spec.: $\mathbf{X}'_s \cup \mathbf{X}_u$	64.4	86.7
	++Mapping: $\mathbf{X}_f$	**70.9**	**86.8**

Table 5: Lexical simplification performance with post-specialisation applied on three input spaces.

We employ LIGHT-LS (Glavaš and Štajner, 2015), a lexical simplification algorithm that: 1) makes substitutions based on word similarities in a semantic vector space, and 2) can be provided an arbitrary embedding space as input.[10] For a complex word, LIGHT-LS considers the most similar words from the vector space as simplification candidates. Candidates are ranked according to several features, indicating simplicity and fitness for the context (semantic relatedness to the context of the complex word). The substitution is made if the best candidate is simpler than the original word. By providing vector spaces post-specialised for semantic similarity to LIGHT-LS, we expect to more often replace complex words with their true synonyms.

We evaluate LIGHT-LS performance in the *all* setup on the LS benchmark compiled by Horn et al. (2014), who crowdsourced 50 manual simplifications for each complex word. As in prior work, we evaluate performance with the following metrics: 1) *Accurracy* (Acc.) is the number of correct simplifications made (i.e., the system made the simplification and its substitution is found in the list of crowdsourced substitutions), divided by the total number of indicated complex words; 2) *Changed* (Ch.) is the percentage of indicated complex words that were replaced by the system (whether or not the replacement was correct).

LS results are summarised in Tab. 5. Post-specialised vector spaces consistently yield 5-6% gain in *Accuracy* compared to respective distributional vectors and embeddings specialised with the state-of-the-art ATTRACT-REPEL model. Similar to DST evaluation, improvements over ATTRACT-REPEL demonstrate the importance of specialising the vectors of the entire vocabulary and not only the vectors of words from the external constraints.

6 Conclusion and Future Work

We have presented a novel post-processing model, termed *post-specialisation*, that specialises word vectors for the full vocabulary of the input vector space. Previous post-processing specialisation models fine-tune word vectors only for words occurring in external lexical resources. In this work, we have demonstrated that the specialisation of the subspace of seen words can be leveraged to learn a mapping function which specialises vectors for all other words, unseen in the external resources. Our results across word similarity and downstream language understanding tasks show consistent improvements over the state-of-the-art specialisation method for all three test languages.

In future work, we plan to extend our approach to specialisation for asymmetric relations such as hypernymy or meronymy (Glavaš and Ponzetto, 2017; Nickel and Kiela, 2017; Vulić and Mrkšić, 2018). We will also investigate more sophisticated non-linear functions. The code is available at: https://github.com/cambridgeltl/post-specialisation/.

Acknowledgments

We thank the three anonymous reviewers for their insightful suggestions. This work is supported by the ERC Consolidator Grant LEXICAL: Lexical Acquisition Across Languages (no 648909).

[10] https://github.com/codogogo/lightls

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual NLP. In *Proceedings of CoNLL*, pages 183–192.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of EMNLP*, pages 2289–2294.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of ACL*, pages 451–462.

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proceedings of ACL*, pages 809–815.

Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Knowledge-powered deep learning for word embedding. In *Proceedings of ECML-PKDD*, pages 132–148.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the ACL*, 5:135–146.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49:1–47.

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

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *CoRR*, abs/1710.04087.

Ryan Cotterell, Hinrich Schütze, and Jason Eisner. 2016. Morphological smoothing and extrapolation of word embeddings. In *Proceedings of ACL*, pages 1651–1660.

Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. 2015. Improving zero-shot learning by mitigating the hubness problem. In *Proceedings of ICLR (Workshop Papers)*.

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

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. 2017. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *CoRR*, abs/1702.03118.

Manaal Faruqui. 2016. *Diverse Context for Learning Word Representations*. Ph.D. thesis, Carnegie Mellon University.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of NAACL-HLT*, pages 1606–1615.

Christiane Fellbaum. 1998. *WordNet*.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: The concept revisited. *ACM Transactions on Information Systems*, 20(1):116–131.

Andrea Frome, Gregory S. Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. 2013. DeViSE: A deep visual-semantic embedding model. In *Proceedings of NIPS*, pages 2121–2129.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *Proceedings of NAACL-HLT*, pages 758–764.

Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. 2016. SimVerb-3500: A large-scale evaluation set of verb similarity. In *Proceedings of EMNLP*, pages 2173–2182.

Goran Glavaš and Simone Paolo Ponzetto. 2017. Dual tensor model for detecting asymmetric lexico-semantic relations. In *Proceedings of EMNLP*, pages 1758–1768.

Goran Glavaš and Sanja Štajner. 2015. Simplifying lexical simplification: Do we need simplified corpora? In *Proceedings of ACL*, pages 63–68.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of AISTATS*, pages 249–256.

Zellig S. Harris. 1954. Distributional structure. *Word*, 10(23):146–162.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of ICCV*, pages 1026–1034.

Matthew Henderson, Blaise Thomson, and Jason D. Wiliams. 2014. The Second Dialog State Tracking Challenge. In *Proceedings of SIGDIAL*, pages 263–272.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. SimLex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Colby Horn, Cathryn Manduca, and David Kauchak. 2014. Learning a lexical simplifier using wikipedia. In *Proceedings of the ACL*, pages 458–463.

Douwe Kiela, Felix Hill, and Stephen Clark. 2015. Specializing word embeddings for similarity or relatedness. In *Proceedings of EMNLP*, pages 2044–2048.

Joo-Kyung Kim, Gokhan Tur, Asli Celikyilmaz, Bin Cao, and Ye-Yi Wang. 2016. Intent detection using semantically enriched word embeddings. In *Proceedings of SLT*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR (Conference Track)*.

Barbara Ann Kipfer. 2009. *Roget's 21st Century Thesaurus (3rd Edition)*. Philip Lief Group.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-normalizing neural networks. *CoRR*, abs/1706.02515.

Jonathan K. Kummerfeld, Taylor Berg-Kirkpatrick, and Dan Klein. 2015. An empirical analysis of optimization for max-margin NLP. In *Proceedings of EMNLP*, pages 273–279.

Angeliki Lazaridou, Georgiana Dinu, and Marco Baroni. 2015. Hubness and pollution: Delving into cross-space mapping for zero-shot learning. In *Proceedings of ACL*, pages 270–280.

Ira Leviant and Roi Reichart. 2015. Separated by an un-common language: Towards judgment language informed vector space modeling. *CoRR*, abs/1508.00106.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of ACL*, pages 302–308.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the ACL*, 3:211–225.

Sihan Li, Jiantao Jiao, Yanjun Han, and Tsachy Weissman. 2016. Demystifying ResNet. *CoRR*, abs/1611.01186.

Quan Liu, Hui Jiang, Si Wei, Zhen-Hua Ling, and Yu Hu. 2015. Learning semantic word embeddings based on ordinal knowledge constraints. In *Proceedings of ACL*, pages 1501–1511.

Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of ICML*.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016a. Context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of CoNLL*, pages 51–61.

Oren Melamud, David McClosky, Siddharth Patwardhan, and Mohit Bansal. 2016b. The role of context types and dimensionality in learning word embeddings. In *Proceedings of NAACL-HLT*, pages 1030–1040.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013a. Exploiting similarities among languages for machine translation. *arXiv preprint, CoRR*, abs/1309.4168.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119.

Dmytro Mishkin and Jiri Matas. 2016. All you need is a good init. In *Proceedings of ICLR (Conference Track)*.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of ACL*, pages 1777–1788.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2015. Multi-domain dialog state tracking using recurrent neural networks. In *Proceedings of ACL*, pages 794–799.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina Maria Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *Proceedings of NAACL-HLT*.

Nikola Mrkšić, Ivan Vulić, Diarmuid Ó Séaghdha, Ira Leviant, Roi Reichart, Milica Gašić, Anna Korhonen, and Steve Young. 2017. Semantic specialisation of distributional word vector spaces using monolingual and cross-lingual constraints. *Transactions of the ACL*, 5:309–324.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of ICML*, pages 807–814.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Kim Anh Nguyen, Maximilian Köper, Sabine Schulte im Walde, and Ngoc Thang Vu. 2017. Hierarchical embeddings for hypernymy detection and directionality. In *Proceedings of EMNLP*, pages 233–243.

Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. 2016. Integrating distributional lexical contrast into word embeddings for antonym-synonym distinction. In *Proceedings of ACL*, pages 454–459.

Maximilian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations. In *Proceedings of NIPS*.

Masataka Ono, Makoto Miwa, and Yutaka Sasaki. 2015. Word embedding-based antonym detection using thesauri and distributional information. In *Proceedings of NAACL-HLT*, pages 984–989.

Dominique Osborne, Shashi Narayan, and Shay Cohen. 2016. Encoding prior knowledge with eigenword embeddings. *Transactions of the ACL*, 4:417–430.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of ACL*, pages 425–430.

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

Prajit Ramachandran, Barret Zoph, and Quoc V. Le. 2017. Searching for activation functions. *CoRR*, abs/1710.05941.

Sascha Rothe and Hinrich Schütze. 2015. AutoExtend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of ACL*, pages 1793–1803.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2017. A survey of cross-lingual embedding models. *CoRR*, abs/1706.04902.

Roy Schwartz, Roi Reichart, and Ari Rappoport. 2015. Symmetric pattern based word embeddings for improved word similarity prediction. In *Proceedings of CoNLL*, pages 258–267.

Pei-Hao Su, Milica Gašić, Nikola Mrkšić, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Continuously learning neural dialogue management. In *arXiv preprint: 1606.02689*.

Ivan Vulić and Anna Korhonen. 2016a. Is "universal syntax" universally useful for learning distributed word representations? In *Proceedings of ACL*, pages 518–524.

Ivan Vulić and Anna Korhonen. 2016b. On the role of seed lexicons in learning bilingual word embeddings. In *Proceedings of ACL*, pages 247–257.

Ivan Vulić and Nikola Mrkšić. 2018. Specialising word vectors for lexical entailment. In *Proceedings of NAACL-HLT*.

Ivan Vulić, Nikola Mrkšić, and Anna Korhonen. 2017a. Cross-lingual induction and transfer of verb classes based on word vector space specialisation. In *Proceedings of EMNLP*, pages 2536–2548.

Ivan Vulić, Nikola Mrkšić, Roi Reichart, Diarmuid Ó Séaghdha, Steve Young, and Anna Korhonen. 2017b. Morph-fitting: Fine-tuning word vector spaces with simple language-specific rules. In *Proceedings of ACL*, pages 56–68.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of EACL*.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. WSABIE: Scaling up to large vocabulary image annotation. In *Proceedings of IJCAI*, pages 2764–2770.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. From paraphrase database to compositional paraphrase model and back. *Transactions of the ACL*, 3:345–358.

Jason D. Williams, Antoine Raux, and Matthew Henderson. 2016. The Dialog State Tracking Challenge series: A review. *Dialogue & Discourse*, 7(3):4–33.

Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. 2014. RC-NET: A general framework for incorporating knowledge into word representations. In *Proceedings of CIKM*, pages 1219–1228.

Steve Young. 2010. Cognitive User Interfaces. *IEEE Signal Processing Magazine*.

Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *Proceedings of ACL*, pages 545–550.

Jingwei Zhang, Jeremy Salwen, Michael Glass, and Alfio Gliozzo. 2014. Word semantic representations using bayesian probabilistic tensor factorization. In *Proceedings of EMNLP*, pages 1522–1531.

Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features

Matteo Pagliardini*
Iprova SA, Switzerland
mpagliardini@iprova.com

Prakhar Gupta*
EPFL, Switzerland
prakhar.gupta@epfl.ch

Martin Jaggi
EPFL, Switzerland
martin.jaggi@epfl.ch

Abstract

The recent tremendous success of unsupervised word embeddings in a multitude of applications raises the obvious question if similar methods could be derived to improve embeddings (i.e. semantic representations) of word sequences as well. We present a simple but efficient unsupervised objective to train distributed representations of sentences. Our method outperforms the state-of-the-art unsupervised models on most benchmark tasks, highlighting the robustness of the produced general-purpose sentence embeddings.

1 Introduction

Improving unsupervised learning is of key importance for advancing machine learning methods, as to unlock access to almost unlimited amounts of data to be used as training resources. The majority of recent success stories of deep learning does not fall into this category but instead relied on supervised training (in particular in the vision domain). A very notable exception comes from the text and natural language processing domain, in the form of semantic word embeddings trained unsupervised (Mikolov et al., 2013b,a; Pennington et al., 2014). Within only a few years from their invention, such word representations – which are based on a simple matrix factorization model as we formalize below – are now routinely trained on very large amounts of raw text data, and have become ubiquitous building blocks of a majority of current state-of-the-art NLP applications.

While very useful semantic representations are available for words, it remains challenging to produce and learn such semantic embeddings for longer pieces of text, such as sentences, paragraphs or entire documents. Even more so, it remains a key goal to learn such general-purpose representations in an unsupervised way.

Currently, two contrary research trends have emerged in text representation learning: On one hand, a strong trend in deep-learning for NLP leads towards increasingly powerful and complex models, such as recurrent neural networks (RNNs), LSTMs, attention models and even Neural Turing Machine architectures. While extremely strong in expressiveness, the increased model complexity makes such models much slower to train on larger datasets. On the other end of the spectrum, simpler "shallow" models such as matrix factorizations (or bilinear models) can benefit from training on much larger sets of data, which can be a key advantage, especially in the unsupervised setting.

Surprisingly, for constructing sentence embeddings, naively using averaged word vectors was shown to outperform LSTMs (see Wieting et al. (2016b) for plain averaging, and Arora et al. (2017) for weighted averaging). This example shows potential in exploiting the trade-off between model complexity and ability to process huge amounts of text using scalable algorithms, towards the simpler side. In view of this trade-off, our work here further advances unsupervised learning of sentence embeddings. Our proposed model can be seen as an extension of the C-BOW (Mikolov et al., 2013b,a) training objective to train sentence instead of word embeddings. We demonstrate that the empirical performance of our resulting general-purpose sentence embeddings very significantly exceeds the state of the art, while keeping the model simplicity as well as training and inference complexity exactly as low as in averaging methods (Wieting et al., 2016b; Arora et al., 2017), thereby also putting the work by (Arora et al., 2017) in perspective.

* indicates equal contribution

Proceedings of NAACL-HLT 2018, pages 528–540
New Orleans, Louisiana, June 1 - 6, 2018.

Contributions. The main contributions in this work can be summarized as follows:

- **Model.** We propose **Sent2Vec**[1], a simple unsupervised model allowing to compose sentence embeddings using word vectors along with n-gram embeddings, simultaneously training composition and the embedding vectors themselves.

- **Efficiency & Scalability.** The computational complexity of our embeddings is only $\mathcal{O}(1)$ vector operations per word processed, both during training and inference of the sentence embeddings. This strongly contrasts all neural network based approaches, and allows our model to learn from extremely large datasets, in a streaming fashion, which is a crucial advantage in the unsupervised setting. Fast inference is a key benefit in downstream tasks and industry applications.

- **Performance.** Our method shows significant performance improvements compared to the current state-of-the-art unsupervised and even semi-supervised models. The resulting general-purpose embeddings show strong robustness when transferred to a wide range of prediction benchmarks.

2 Model

Our model is inspired by simple matrix factor models (bilinear models) such as recently very successfully used in unsupervised learning of word embeddings (Mikolov et al., 2013b,a; Pennington et al., 2014; Bojanowski et al., 2017) as well as supervised of sentence classification (Joulin et al., 2017). More precisely, these models can all be formalized as an optimization problem of the form

$$\min_{\boldsymbol{U},\boldsymbol{V}} \sum_{S \in \mathcal{C}} f_S(\boldsymbol{U}\boldsymbol{V}\boldsymbol{\iota}_S) \qquad (1)$$

for two parameter matrices $\boldsymbol{U} \in \mathbb{R}^{k \times h}$ and $\boldsymbol{V} \in \mathbb{R}^{h \times |\mathcal{V}|}$, where $\mathcal{V}$ denotes the vocabulary. Here, the columns of the matrix $\boldsymbol{V}$ represent the learnt source word vectors whereas those of $\boldsymbol{U}$ represent the target word vectors. For a given sentence S, which can be of arbitrary length, the indicator vector $\boldsymbol{\iota}_S \in \{0,1\}^{|\mathcal{V}|}$ is a binary vector encoding S (bag of words encoding).

Fixed-length context windows S running over the corpus are used in word embedding methods as in C-BOW (Mikolov et al., 2013b,a) and GloVe (Pennington et al., 2014). Here we have $k = |\mathcal{V}|$ and each cost function $f_S : \mathbb{R}^k \to \mathbb{R}$ only depends on a single row of its input, describing the observed target word for the given fixed-length context S. In contrast, for sentence embeddings which are the focus of our paper here, S will be entire sentences or documents (therefore variable length). This property is shared with the supervised FastText classifier (Joulin et al., 2017), which however uses soft-max with $k \ll |\mathcal{V}|$ being the number of class labels.

2.1 Proposed Unsupervised Model

We propose a new unsupervised model, **Sent2Vec**, for learning universal sentence embeddings. Conceptually, the model can be interpreted as a natural extension of the word-contexts from C-BOW (Mikolov et al., 2013b,a) to a larger sentence context, with the sentence words being specifically optimized towards additive combination over the sentence, by means of the unsupervised objective function.

Formally, we learn a source (or context) embedding $\boldsymbol{v}_w$ and target embedding $\boldsymbol{u}_w$ for each word w in the vocabulary, with embedding dimension h and $k = |\mathcal{V}|$ as in (1). The sentence embedding is defined as the average of the source word embeddings of its constituent words, as in (2). We augment this model furthermore by also learning source embeddings for not only unigrams but also n-grams present in each sentence, and averaging the n-gram embeddings along with the words, i.e., the sentence embedding $\boldsymbol{v}_S$ for S is modeled as

$$\boldsymbol{v}_S := \frac{1}{|R(S)|}\boldsymbol{V}\boldsymbol{\iota}_{R(S)} = \frac{1}{|R(S)|}\sum_{w \in R(S)} \boldsymbol{v}_w \qquad (2)$$

where $R(S)$ is the list of n-grams (including unigrams) present in sentence S. In order to predict a missing word from the context, our objective models the softmax output approximated by negative sampling following (Mikolov et al., 2013b). For the large number of output classes $|\mathcal{V}|$ to be predicted, negative sampling is known to significantly improve training efficiency, see also (Goldberg and Levy, 2014). Given the binary logistic

[1] All our code and pre-trained models will be made publicly available on http://github.com/epfml/sent2vec

loss function $\ell : x \mapsto \log\left(1 + e^{-x}\right)$ coupled with negative sampling, our unsupervised training objective is formulated as follows:

$$\min_{\boldsymbol{U},\boldsymbol{V}} \sum_{S\in\mathcal{C}} \sum_{w_t\in S} \Bigg(\ell\big(\boldsymbol{u}_{w_t}^\top \boldsymbol{v}_{S\setminus\{w_t\}}\big) + \sum_{w'\in N_{w_t}} \ell\big(-\boldsymbol{u}_{w'}^\top \boldsymbol{v}_{S\setminus\{w_t\}}\big)\Bigg)$$

where S corresponds to the current sentence and N_{w_t} is the set of words sampled negatively for the word $w_t \in S$. The negatives are sampled[2] following a multinomial distribution where each word w is associated with a probability $q_n(w) := \sqrt{f_w} \,/\, \big(\sum_{w_i\in\mathcal{V}} \sqrt{f_{w_i}}\big)$, where f_w is the normalized frequency of w in the corpus.

To select the possible target unigrams (positives), we use subsampling as in (Joulin et al., 2017; Bojanowski et al., 2017), each word w being discarded with probability $1 - q_p(w)$ where $q_p(w) := \min\big\{1, \sqrt{t/f_w} + t/f_w\big\}$. Where t is the subsampling hyper-parameter. Subsampling prevents very frequent words of having too much influence in the learning as they would introduce strong biases in the prediction task. With positives subsampling and respecting the negative sampling distribution, the precise training objective function becomes

$$\min_{\boldsymbol{U},\boldsymbol{V}} \sum_{S\in\mathcal{C}} \sum_{w_t\in S} \Bigg(q_p(w_t)\ell\big(\boldsymbol{u}_{w_t}^\top \boldsymbol{v}_{S\setminus\{w_t\}}\big) + |N_{w_t}| \sum_{w'\in\mathcal{V}} q_n(w')\ell\big(-\boldsymbol{u}_{w'}^\top \boldsymbol{v}_{S\setminus\{w_t\}}\big)\Bigg) \tag{3}$$

2.2 Computational Efficiency

In contrast to more complex neural network based models, one of the core advantages of the proposed technique is the low computational cost for both inference and training. Given a sentence S and a trained model, computing the sentence representation $\boldsymbol{v}_S$ only requires $|S| \cdot h$ floating point operations (or $|R(S)| \cdot h$ to be precise for the n-gram case, see (2)), where h is the embedding dimension. The same holds for the cost of training with SGD on the objective (3), per sentence seen in the training corpus. Due to the simplicity of the model, parallel training is straight-forward using parallelized or distributed SGD.

Also, in order to store higher-order n-grams efficiently, we use the standard hashing trick, see e.g. (Weinberger et al., 2009), with the same hashing function as used in FastText (Joulin et al., 2017; Bojanowski et al., 2017).

2.3 Comparison to C-BOW

C-BOW (Mikolov et al., 2013b,a) aims to predict a chosen target word given its fixed-size context window, the context being defined by the average of the vectors associated with the words at a distance less than the window size hyper-parameter ws. If our system, when restricted to unigram features, can be seen as an extension of C-BOW where the context window includes the entire sentence, in practice there are few important differences as C-BOW uses important tricks to facilitate the learning of word embeddings. C-BOW first uses frequent word subsampling on the sentences, deciding to discard each token w with probability $q_p(w)$ or alike (small variations exist across implementations). Subsampling prevents the generation of n-grams features, and deprives the sentence of an important part of its syntactical features. It also shortens the distance between subsampled words, implicitly increasing the span of the context window. A second trick consists of using dynamic context windows: for each subsampled word w, the size of its associated context window is sampled uniformly between 1 and ws. Using dynamic context windows is equivalent to weighing by the distance from the focus word w divided by the window size (Levy et al., 2015). This makes the prediction task local, and go against our objective of creating sentence embeddings as we want to learn how to compose all n-gram features present in a sentence. In the results section, we report a significant improvement of our method over C-BOW.

2.4 Model Training

Three different datasets have been used to train our models: the Toronto book corpus[3], Wikipedia sentences and tweets. The Wikipedia and Toronto books sentences have been tokenized using the Stanford NLP library (Manning et al., 2014), while for tweets we used the NLTK tweets tokenizer (Bird et al., 2009). For training, we select a

[2] To efficiently sample negatives, a pre-processing table is constructed, containing the words corresponding to the square root of their corpora frequency. Then, the negatives N_{w_t} are sampled uniformly at random from the negatives table except the target w_t itself, following (Joulin et al., 2017; Bojanowski et al., 2017).

[3] http://www.cs.toronto.edu/~mbweb/

sentence randomly from the dataset and then proceed to select all the possible target unigrams using subsampling. We update the weights using SGD with a linearly decaying learning rate.

Also, to prevent overfitting, for each sentence we use dropout on its list of n-grams $R(S) \setminus \{U(S)\}$, where $U(S)$ is the set of all unigrams contained in sentence S. After empirically trying multiple dropout schemes, we find that dropping K n-grams ($n > 1$) for each sentence is giving superior results compared to dropping each token with some fixed probability. This dropout mechanism would negatively impact shorter sentences. The regularization can be pushed further by applying L1 regularization to the word vectors. Encouraging sparsity in the embedding vectors is particularly beneficial for high dimension h. The additional soft thresholding in every SGD step adds negligible computational cost. See also Appendix B. We train two models on each dataset, one with unigrams only and one with unigrams and bigrams. All training parameters for the models are provided in Table 5 in the appendix. Our C++ implementation builds upon the FastText library (Joulin et al., 2017; Bojanowski et al., 2017). We will make our code and pre-trained models available open-source.

3 Related Work

We discuss existing models which have been proposed to construct sentence embeddings. While there is a large body of works in this direction – several among these using e.g. labelled datasets of paraphrase pairs to obtain sentence embeddings in a supervised manner (Wieting et al., 2016a,b; Conneau et al., 2017) to learn sentence embeddings – we here focus on unsupervised, task-independent models. While some methods require ordered raw text i.e., a coherent corpus where the next sentence is a logical continuation of the previous sentence, others rely only on raw text i.e., an unordered collection of sentences. Finally, we also discuss alternative models built from structured data sources.

3.1 Unsupervised Models Independent of Sentence Ordering

The **ParagraphVector DBOW** model (Le and Mikolov, 2014) is a log-linear model which is trained to learn sentence as well as word embeddings and then use a softmax distribution to predict words contained in the sentence given the sentence vector representation. They also propose a different model **ParagraphVector DM** where they use n-grams of consecutive words along with the sentence vector representation to predict the next word.

(Lev et al., 2015) also presented an early approach to obtain compositional embeddings from word vectors. They use different compositional techniques including static averaging or Fisher vectors of a multivariate Gaussian to obtain sentence embeddings from word2vec models.

Hill et al. (2016a) propose a **Sequential (Denoising) Autoencoder, S(D)AE**. This model first introduces noise in the input data: Firstly each word is deleted with probability p_0, then for each non-overlapping bigram, words are swapped with probability p_x. The model then uses an LSTM-based architecture to retrieve the original sentence from the corrupted version. The model can then be used to encode new sentences into vector representations. In the case of $p_0 = p_x = 0$, the model simply becomes a Sequential Autoencoder. Hill et al. (2016a) also propose a variant **(S(D)AE + embs.)** in which the words are represented by fixed pre-trained word vector embeddings.

Arora et al. (2017) propose a model in which sentences are represented as a weighted average of fixed (pre-trained) word vectors, followed by post-processing step of subtracting the principal component. Using the generative model of (Arora et al., 2016), words are generated conditioned on a sentence "discourse" vector $\boldsymbol{c}_s$:

$$Pr[w \,|\, \boldsymbol{c}_s] = \alpha f_w + (1-\alpha)\frac{\exp(\tilde{\boldsymbol{c}}_s^\top \boldsymbol{v}_w)}{Z_{\tilde{\boldsymbol{c}}_s}},$$

where $Z_{\tilde{\boldsymbol{c}}_s} := \sum_{w \in \mathcal{V}} \exp(\tilde{\boldsymbol{c}}_s^\top \boldsymbol{v}_w)$ and $\tilde{\boldsymbol{c}}_s := \beta \boldsymbol{c}_0 + (1-\beta)\boldsymbol{c}_s$ and α, β are scalars. $\boldsymbol{c}_0$ is the common discourse vector, representing a shared component among all discourses, mainly related to syntax. It allows the model to better generate syntactical features. The αf_w term is here to enable the model to generate some frequent words even if their matching with the discourse vector $\tilde{\boldsymbol{c}}_s$ is low.

Therefore, this model tries to generate sentences as a mixture of three type of words: words matching the sentence discourse vector $\boldsymbol{c}_s$, syntactical words matching $\boldsymbol{c}_0$, and words with high f_w. (Arora et al., 2017) demonstrated that for this model, the MLE of $\tilde{\boldsymbol{c}}_s$ can be approximated by $\sum_{w \in S} \frac{a}{f_w + a} \boldsymbol{v}_w$, where a is a scalar. The sentence

discourse vector can hence be obtained by subtracting $\boldsymbol{c}_0$ estimated by the first principal component of $\tilde{\boldsymbol{c}}_s$'s on a set of sentences. In other words, the sentence embeddings are obtained by a weighted average of the word vectors stripping away the syntax by subtracting the common discourse vector and down-weighting frequent tokens. They generate sentence embeddings from diverse pre-trained word embeddings among which are unsupervised word embeddings such as GloVe (Pennington et al., 2014) as well as supervised word embeddings such as paragram-SL999 (PSL) (Wieting et al., 2015) trained on the Paraphrase Database (Ganitkevitch et al., 2013).

In a very different line of work, **C-PHRASE** (Pham et al., 2015) relies on additional information from the syntactic parse tree of each sentence, which is incorporated into the C-BOW training objective.

Huang and Anandkumar (2016) show that single layer CNNs can be modeled using a tensor decomposition approach. While building on an unsupervised objective, the employed dictionary learning step for obtaining phrase templates is task-specific (for each use-case), not resulting in general-purpose embeddings.

3.2 Unsupervised Models Depending on Sentence Ordering

The **SkipThought** model (Kiros et al., 2015) combines sentence level models with recurrent neural networks. Given a sentence S_i from an ordered corpus, the model is trained to predict S_{i-1} and S_{i+1}.

FastSent (Hill et al., 2016a) is a sentence-level log-linear bag-of-words model. Like SkipThought, it uses adjacent sentences as the prediction target and is trained in an unsupervised fashion. Using word sequences allows the model to improve over the earlier work of paragraph2vec (Le and Mikolov, 2014). (Hill et al., 2016a) augment FastSent further by training it to predict the constituent words of the sentence as well. This model is named **FastSent + AE** in our comparisons.

Compared to our approach, **Siamese C-BOW** (Kenter et al., 2016) shares the idea of learning to average word embeddings over a sentence. However, it relies on a Siamese neural network architecture to predict surrounding sentences, contrasting our simpler unsupervised objective.

Note that on the character sequence level instead of word sequences, FastText (Bojanowski et al., 2017) uses the same conceptual model to obtain better word embeddings. This is most similar to our proposed model, with two key differences: Firstly, we predict from source word sequences to target words, as opposed to character sequences to target words, and secondly, our model is averaging the source embeddings instead of summing them.

3.3 Models requiring structured data

DictRep (Hill et al., 2016b) is trained to map dictionary definitions of the words to the pre-trained word embeddings of these words. They use two different architectures, namely BOW and RNN (LSTM) with the choice of learning the input word embeddings or using them pre-trained. A similar architecture is used by the **CaptionRep** variant, but here the task is the mapping of given image captions to a pre-trained vector representation of these images.

4 Evaluation Tasks

We use a standard set of supervised as well as unsupervised benchmark tasks from the literature to evaluate our trained models, following (Hill et al., 2016a). The breadth of tasks allows to fairly measure generalization to a wide area of different domains, testing the general-purpose quality (universality) of all competing sentence embeddings. For downstream supervised evaluations, sentence embeddings are combined with logistic regression to predict target labels. In the unsupervised evaluation for sentence similarity, correlation of the cosine similarity between two embeddings is compared to human annotators.

Downstream Supervised Evaluation. Sentence embeddings are evaluated for various supervised classification tasks as follows. We evaluate paraphrase identification (MSRP) (Dolan et al., 2004), classification of movie review sentiment (MR) (Pang and Lee, 2005), product reviews (CR) (Hu and Liu, 2004), subjectivity classification (SUBJ) (Pang and Lee, 2004), opinion polarity (MPQA) (Wiebe et al., 2005) and question type classification (TREC) (Voorhees, 2002). To classify, we use the code provided by (Kiros et al., 2015) in the same manner as in (Hill et al., 2016a). For the MSRP dataset, containing pairs of sentences (S_1, S_2) with associated paraphrase label, we generate feature vectors by concatenating

their Sent2Vec representations $|\boldsymbol{v}_{S_1} - \boldsymbol{v}_{S_2}|$ with the component-wise product $\boldsymbol{v}_{S_1} \odot \boldsymbol{v}_{S_2}$. The predefined training split is used to tune the L2 penalty parameter using cross-validation and the accuracy and F1 scores are computed on the test set. For the remaining 5 datasets, Sent2Vec embeddings are inferred from input sentences and directly fed to a logistic regression classifier. Accuracy scores are obtained using 10-fold cross-validation for the MR, CR, SUBJ and MPQA datasets. For those datasets nested cross-validation is used to tune the L2 penalty. For the TREC dataset, as for the MRSP dataset, the L2 penalty is tuned on the predefined train split using 10-fold cross-validation, and the accuracy is computed on the test set.

Unsupervised Similarity Evaluation. We perform unsupervised evaluation of the learnt sentence embeddings using the sentence cosine similarity, on the STS 2014 (Agirre et al., 2014) and SICK 2014 (Marelli et al., 2014) datasets. These similarity scores are compared to the gold-standard human judgements using Pearson's r (Pearson, 1895) and Spearman's ρ (Spearman, 1904) correlation scores. The SICK dataset consists of about 10,000 sentence pairs along with relatedness scores of the pairs. The STS 2014 dataset contains 3,770 pairs, divided into six different categories on the basis of the origin of sentences/phrases, namely Twitter, headlines, news, forum, WordNet and images.

5 Results and Discussion

In Tables 1 and 2, we compare our results with those obtained by (Hill et al., 2016a) on different models. Table 3 in the last column shows the dramatic improvement in training time of our models (and other C-BOW-inspired models) in contrast to neural network based models. All our Sent2Vec models are trained on a machine with 2x Intel Xeon E5−2680v3, 12 cores @2.5GHz. Along with the models discussed in Section 3, this also includes the sentence embedding baselines obtained by simple averaging of word embeddings over the sentence, in both the C-BOW and skip-gram variants. TF-IDF BOW is a representation consisting of the counts of the 200,000 most common feature-words, weighed by their TF-IDF frequencies. To ensure coherence, we only include unsupervised models in the main paper. Performance of supervised and semi-supervised models on these evaluations can be observed in Tables 6 and 7 in the appendix.

Downstream Supervised Evaluation Results. On running supervised evaluations and observing the results in Table 1, we find that on an average our models are second only to SkipThought vectors. Also, both our models achieve state of the art results on the CR task. We also observe that on half of the supervised tasks, our unigrams + bigram model is the best model after SkipThought. Our models are weaker on the MSRP task (which consists of the identification of labelled paraphrases) compared to state-of-the-art methods. However, we observe that the models which perform very strongly on this task end up faring very poorly on the other tasks, indicating a lack of generalizability.

On rest of the tasks, our models perform extremely well. The SkipThought model is able to outperform our models on most of the tasks as it is trained to predict the previous and next sentences and a lot of tasks are able to make use of this contextual information missing in our Sent2Vec models. For example, the TREC task is a poor measure of how one predicts the content of the sentence (the question) but a good measure of how the next sentence in the sequence (the answer) is predicted.

Unsupervised Similarity Evaluation Results. In Table 2, we see that our Sent2Vec models are state-of-the-art on the majority of tasks when comparing to all the unsupervised models trained on the Toronto corpus, and clearly achieve the best averaged performance. Our Sent2Vec models also on average outperform or are at par with the C-PHRASE model, despite significantly lagging behind on the STS 2014 WordNet and News subtasks. This observation can be attributed to the fact that a big chunk of the data that the C-PHRASE model is trained on comes from English Wikipedia, helping it to perform well on datasets involving definition and news items. Also, C-PHRASE uses data three times the size of the Toronto book corpus. Interestingly, our model outperforms C-PHRASE when trained on Wikipedia, as shown in Table 3, despite the fact that we use no parse tree information.

Official STS 2017 benchmark. In the official results of the most recent edition of the STS 2017 benchmark (Cer et al., 2017), our model also significantly outperforms C-PHRASE, and in fact delivers the best unsupervised baseline method.

[4]For the Siamese C-BOW model trained on the Toronto

Data	Model	MSRP (Acc / F1)	MR	CR	SUBJ	MPQA	TREC	Average
Unordered Sentences: (Toronto Books; 70 million sentences, 0.9 Billion Words)	SAE	**74.3** / 81.7	62.6	68.0	86.1	76.8	80.2	74.7
	SAE + embs.	70.6 / 77.9	73.2	75.3	89.8	86.2	80.4	79.3
	SDAE	**76.4 / 83.4**	67.6	74.0	89.3	81.3	77.7	78.3
	SDAE + embs.	**73.7** / 80.7	74.6	78.0	90.8	**86.9**	78.4	80.4
	ParagraphVec DBOW	72.9 / 81.1	60.2	66.9	76.3	70.7	59.4	67.7
	ParagraphVec DM	73.6 / **81.9**	61.5	68.6	76.4	78.1	55.8	69.0
	Skipgram	69.3 / 77.2	73.6	77.3	89.2	85.0	82.2	78.5
	C-BOW	67.6 / 76.1	73.6	77.3	89.1	85.0	82.2	79.1
	Unigram TFIDF	73.6 / 81.7	73.7	79.2	90.3	82.4	**85.0**	80.7
	Sent2Vec uni.	72.2 / 80.3	75.1	**80.2**	90.6	**86.3**	83.8	**81.4**
	Sent2Vec uni. + bi.	72.5 / 80.8	**75.8**	**80.3**	**91.2**	85.9	**86.4**	**82.0**
Ordered Sentences: Toronto Books	SkipThought	73.0 / **82.0**	**76.5**	**80.1**	**93.6**	**87.1**	**92.2**	**83.8**
	FastSent	72.2 / 80.3	70.8	78.4	88.7	80.6	76.8	77.9
	FastSent+AE	71.2 / 79.1	71.8	76.7	88.8	81.5	80.4	78.4
2.8 Billion words	C-PHRASE	72.2 / 79.6	**75.7**	78.8	**91.1**	86.2	78.8	80.5

Table 1: Comparison of the performance of different models on different **supervised evaluation** tasks. An underline indicates the best performance for the dataset. Top 3 performances in each data category are shown in bold. The average is calculated as the average of accuracy for each category (For MSRP, we take the accuracy).)

	STS 2014						SICK 2014	
Model	News	Forum	WordNet	Twitter	Images	Headlines	Test + Train	Average
SAE	.17/.16	.12/.12	.30/.23	.28/.22	.49/.46	.13/.11	.32/.31	.26/.23
SAE + embs.	.52/.54	.22/.23	.60/.55	.60/.60	.64/.64	.41/.41	.47/.49	.50/.49
SDAE	.07/.04	.11/.13	.33/.24	.44/.42	.44/.38	.36/.36	.46/.46	.31/.29
SDAE + embs.	.51/.54	.29/.29	.56/.50	.57/.58	.59/.59	.43/.44	.46/.46	.49/.49
ParagraphVec DBOW	.31/.34	.32/.32	.53/.50	.43/.46	.46/.44	.39/.41	.42/.46	.41/.42
ParagraphVec DM	.42/.46	.33/.34	.51/.48	.54/.57	.32/.30	.46/.47	.44/.40	.43/.43
Skipgram	.56/.59	.42/.42	.73/**.70**	**.71**/.74	.65/.67	.55/.58	.60/.69	.60/.63
C-BOW	.57/.61	**.43/.44**	.72/.69	**.71/.75**	.71/.73	.55/.59	.60/.69	.60/.65
Unigram TF-IDF	.48/.48	.40/.38	.60/.59	.63/.65	.72/.74	.49/.49	.52/.58	.55/.56
Sent2Vec uni.	**.62/.67**	**.49/.49**	**.75/.72**	.70/**.75**	**.78/.82**	**.61/.63**	**.61/.70**	**.65/.68**
Sent2Vec uni. + bi.	**.62/.67**	**.51/.51**	.71/.68	.70/**.75**	**.75/.79**	.59/.62	**.62/.70**	**.65/.67**
SkipThought	.44/.45	.14/.15	.39/.34	.42/.43	.55/.60	.43/.44	.57/.60	.42/.43
FastSent	.58/.59	.41/.36	**.74/.70**	.63/.66	.74/.78	.57/.59	**.61/.72**	.61/.63
FastSent+AE	.56/.59	.41/.40	.69/.64	.70/.74	.63/.65	.58/.60	.60/.65	.60/.61
Siamese C-BOW[4]	.58/.59	.42/.41	.66/.61	**.71**/.73	.65/.65	**.63/.64**	–	–
C-PHRASE	**.69/.71**	**.43**/.41	**.76/.73**	.60/.65	**.75/.79**	**.60/.65**	.60/**.72**	**.63/.67**

Table 2: **Unsupervised Evaluation Tasks**: Comparison of the performance of different models on Spearman/Pearson correlation measures. An underline indicates the best performance for the dataset. Top 3 performances in each data category are shown in bold. The average is calculated as the average of entries for each correlation measure.

Macro Average. To summarize our contributions on both supervised and unsupervised tasks, in Table 3 we present the results in terms of the macro average over the averages of both supervised and unsupervised tasks along with the training times of the models[5]. For unsupervised tasks, averages are taken over both Spearman and Pearson scores. The comparison includes the best performing unsupervised and semi-supervised methods described in Section 3. For models trained on the Toronto books dataset, we report a 3.8 % points improvement over the state of the art. Considering all supervised, semi-supervised methods and all datasets compared in (Hill et al., 2016a), we report a 2.2 % points improvement.

We also see a noticeable improvement in accuracy as we use larger datasets like Twitter and Wikipedia. We furthermore see that the Sent2Vec models are faster to train when compared to methods like SkipThought and DictRep, owing to the SGD optimizer allowing a high degree of parallelizability.

We can clearly see Sent2Vec outperforming other unsupervised and even semi-supervised methods. This can be attributed to the superior generalizability of our model across supervised and unsupervised tasks.

Comparison with Arora et al. (2017). We also compare our work with Arora et al. (2017) who also use additive compositionality to obtain sentence embeddings. However, in contrast to our

corpus, supervised evaluation as well as similarity evaluation results on the SICK 2014 dataset are unavailable.

[5]time taken to train C-PHRASE models is unavailable

Type	Training corpus	Method	Supervised average	Unsupervised average	Macro average	Training time (in hours)
unsupervised	twitter (19.7B words)	**Sent2Vec uni. + bi.**	83.5	68.3	75.9	6.5*
unsupervised	twitter (19.7B words)	**Sent2Vec uni.**	82.2	69.0	75.6	3*
unsupervised	Wikipedia (1.7B words)	**Sent2Vec uni. + bi.**	83.3	66.2	74.8	2*
unsupervised	Wikipedia (1.7B words)	**Sent2Vec uni.**	82.4	66.3	74.3	3.5*
unsupervised	Toronto books (0.9B words)	**Sent2Vec books uni.**	81.4	66.7	74.0	1*
unsupervised	Toronto books (0.9B words)	**Sent2Vec books uni. + bi.**	82.0	65.9	74.0	1.2*
semi-supervised	structured dictionary dataset	DictRep BOW + emb	80.5	66.9	73.7	24**
unsupervised	2.8B words + parse info.	C-PHRASE	80.5	64.9	72.7	–
unsupervised	Toronto books (0.9B words)	C-BOW	79.1	62.8	70.2	2
unsupervised	Toronto books (0.9B words)	FastSent	77.9	62.0	70.0	2
unsupervised	Toronto books (0.9B words)	SkipThought	83.8	42.5	63.1	336**

Table 3: Best unsupervised and semi-supervised methods ranked by macro average along with their training times. ** indicates trained on GPU. * indicates trained on a single node using 30 threads. Training times for non-Sent2Vec models are due to Hill et al. (2016a). For CPU based competing methods, we were able to reproduce all published timings (+-10%) using our same hardware as for training Sent2Vec.

Dataset	Unsupervised GloVe (840B words) + WR	Semi-supervised PSL + WR	Sent2Vec Unigrams (19.7B words) Tweets Model	Sent2Vec Unigrams + Bigrams (19.7B words) Tweets Model
STS 2014	0.685	0.735	0.710	0.701
SICK 2014	0.722	0.729	0.710	0.715
Supervised average	0.815	0.807	0.822	0.835

Table 4: Comparison of the performance of the unsupervised and semi-supervised sentence embeddings by (Arora et al., 2017) with our models. Unsupervised comparisons are in terms of Pearson's correlation, while comparisons on supervised tasks are stating the average described in Table 1.

model, they use fixed, pre-trained word embeddings to build a weighted average of these embeddings using unigram probabilities. While we couldn't find pre-trained state of the art word embeddings trained on the Toronto books corpus, we evaluated their method using GloVe embeddings obtained from the larger Common Crawl Corpus[6], which is 42 times larger than our twitter corpus, greatly favoring their method over ours.

In Table 4, we report an experimental comparison to their model on unsupervised tasks. In the table, the suffix W indicates that their down-weighting scheme has been used, while the suffix R indicates the removal of the first principal component. They report values of $a \in [10^{-4}, 10^{-3}]$ as giving the best results and used $a = 10^{-3}$ for all their experiments. We observe that our results are competitive with the embeddings of Arora et al. (2017) for purely unsupervised methods. It is important to note that the scores obtained from supervised task-specific PSL embeddings trained for the purpose of semantic similarity outperform our method on both SICK and average STS 2014, which is expected as our model is trained purely unsupervised.

In order to facilitate a more detailed comparison, we also evaluated the unsupervised Glove + WR embeddings on downstream supervised tasks and compared them to our twitter models. To use Arora et al. (2017)'s method in a supervised setup, we precomputed and stored the common discourse vector $\boldsymbol{c}_0$ using 2 million random Wikipedia sentences. On an average, our models outperform their unsupervised models by a significant margin, this despite the fact that they used GloVe embeddings trained on larger corpora than ours (42 times larger). Our models also outperform their semi-supervised PSL + WR model. This indicates our model learns a more precise weighing scheme than the static one proposed by Arora et al. (2017).

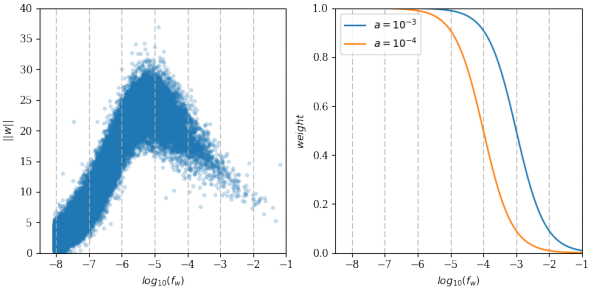


Figure 1: *Left figure:* the profile of the word vector L_2-norms as a function of $\log(f_w)$ for each vocabulary word w, as learnt by our unigram model trained on Toronto books. *Right figure:* down-weighting scheme proposed by Arora et al. (2017): $weight(w) = \frac{a}{a+f_w}$.

The effect of datasets and n-grams. Despite being trained on three very different datasets, all of our models generalize well to sometimes very

[6] http://www.cs.toronto.edu/~mbweb/

specific domains. Models trained on Toronto Corpus are the state-of-the-art on the STS 2014 images dataset even beating the supervised CaptionRep model trained on images. We also see that addition of bigrams to our models doesn't help much when it comes to unsupervised evaluations but gives a significant boost-up in accuracy on supervised tasks. We attribute this phenomenon to the ability of bigrams models to capture some non-compositional features missed by unigrams models. Having a single representation for *"not good"* or *"very bad"* can boost the supervised model's ability to infer relevant features for the corresponding classifier. For semantic similarity tasks however, the relative uniqueness of bigrams results in pushing sentence representations further apart, which can explain the average drop of scores for bigrams models on those tasks.

On learning the importance and the direction of the word vectors. Our model – by learning how to generate and compose word vectors – has to learn both the direction of the word embeddings as well as their norm. Considering the norms of the used word vectors as by our averaging over the sentence, we observe an interesting distribution of the "importance" of each word. In Figure 1 we show the profile of the L_2-norm as a function of $\log(f_w)$ for each $w \in \mathcal{V}$, and compare it to the static down-weighting mechanism of Arora et al. (2017). We can observe that our model is learning to down-weight frequent tokens by itself. It is also down-weighting rare tokens and the $norm$ profile seems to roughly follow Luhn's hypothesis (Luhn, 1958), a well known information retrieval paradigm, stating that mid-rank terms are the most significant to discriminate content.

6 Conclusion

In this paper, we introduce a novel, computationally efficient, unsupervised, C-BOW-inspired method to train and infer sentence embeddings. On supervised evaluations, our method, on an average, achieves better performance than all other unsupervised competitors with the exception of SkipThought. However, SkipThought vectors show a very poor performance on sentence similarity tasks while our model is state-of-the-art for these evaluations on average. Also, our model is generalizable, extremely fast to train, simple to understand and easily interpretable, showing the relevance of simple and well-grounded representation models in contrast to the models using deep architectures. Future work could focus on augmenting the model to exploit data with ordered sentences. Furthermore, we would like to investigate the model's ability to use pre-trained embeddings for downstream transfer learning tasks.

Acknowledgments

We are indebted to Piotr Bojanowski and Armand Joulin for helpful discussions. This project was supported by a Google Faculty Research Award.

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. Semeval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*. Association for Computational Linguistics Dublin, Ireland, pages 81–91.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2016. A Latent Variable Model Approach to PMI-based Word Embeddings. In *Transactions of the Association for Computational Linguistics*. pages 385–399.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *International Conference on Learning Representations (ICLR)*.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* 5:135–146.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Cross-lingual Focused Evaluation. In *SemEval-2017 - Proceedings of the 11th International Workshop on Semantic Evaluations*. Association for Computational Linguistics, Vancouver, Canada, pages 1–14.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364* .

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on*

Computational Linguistics. Association for Computational Linguistics, page 350.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In *HLT-NAACL*. pages 758–764.

Yoav Goldberg and Omer Levy. 2014. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv* .

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016a. Learning Distributed Representations of Sentences from Unlabelled Data. In *Proceedings of NAACL-HLT*.

Felix Hill, KyungHyun Cho, Anna Korhonen, and Yoshua Bengio. 2016b. Learning to understand phrases by embedding the dictionary. *TACL* 4:17–30.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 168–177.

Furong Huang and Animashree Anandkumar. 2016. Unsupervised Learning of Word-Sequence Representations from Scratch via Convolutional Tensor Decomposition. *arXiv* .

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, Short Papers*. Valencia, Spain, pages 427–431.

Tom Kenter, Alexey Borisov, and Maarten de Rijke. 2016. Siamese CBOW: Optimizing Word Embeddings for Sentence Representations. In *ACL - Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Berlin, Germany, pages 941–951.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-Thought Vectors. In *NIPS 2015 - Advances in Neural Information Processing Systems 28*. pages 3294–3302.

Quoc V Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *ICML 2014 - Proceedings of the 31st International Conference on Machine Learning*. volume 14, pages 1188–1196.

Guy Lev, Benjamin Klein, and Lior Wolf. 2015. In defense of word embedding for generic text representation. In *International Conference on Applications of Natural Language to Information Systems*. Springer, pages 35–50.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics* 3:211–225.

Hans Peter Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of research and development* 2(2):159–165.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*. pages 55–60.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A sick cure for the evaluation of compositional distributional semantic models. In *LREC*. pages 216–223.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS - Advances in Neural Information Processing Systems 26*. pages 3111–3119.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, page 271.

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

Karl Pearson. 1895. Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London* 58:240–242.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. volume 14, pages 1532–1543.

NT Pham, G Kruszewski, A Lazaridou, and M Baroni. 2015. Jointly optimizing word representations for lexical and sentential tasks with the c-phrase model. *ACL/IJCNLP* .

R Tyrrell Rockafellar. 1976. Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization* 14(5):877–898.

Charles Spearman. 1904. The proof and measurement of association between two things. *The American journal of psychology* 15(1):72–101.

Ellen M Voorhees. 2002. Overview of the trec 2001 question answering track. In *NIST special publication*. pages 42–51.

Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. 2009. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, pages 1113–1120.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation* 39(2):165–210.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016a. Charagram: Embedding Words and Sentences via Character n-grams. In *EMNLP - Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 1504–1515.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016b. Towards universal paraphrastic sentence embeddings. In *International Conference on Learning Representations (ICLR)*.

John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Dan Roth. 2015. From paraphrase database to compositional paraphrase model and back. In *TACL - Transactions of the Association for Computational Linguistics*.

A Parameters for training models

Model	Embedding Dimensions	Minimum word count	Minimum Target word Count	Initial Lear ning Rate	Epochs	Subsampling hyper-parameter	Bigrams Dropped per sentence	Number of negatives sampled
Book corpus Sent2Vec unigrams	700	5	8	0.2	13	1×10^{-5}	-	10
Book corpus Sent2Vec unigrams + bigrams	700	5	5	0.2	12	5×10^{-6}	7	10
Wiki Sent2Vec unigrams	600	8	20	0.2	9	1×10^{-5}	-	10
Wiki Sent2Vec unigrams + bigrams	700	8	20	0.2	9	5×10^{-6}	4	10
Twitter Sent2Vec unigrams	700	20	20	0.2	3	1×10^{-6}	-	10
Twitter Sent2Vec unigrams + bigrams	700	20	20	0.2	3	1×10^{-6}	3	10

Table 5: Training parameters for the Sent2Vec models

B L1 regularization of models

Optionally, our model can be additionally improved by adding an L1 regularizer term in the objective function, leading to slightly better generalization performance. Additionally, encouraging sparsity in the embedding vectors is beneficial for memory reasons, allowing higher embedding dimensions h.

We propose to apply L1 regularization individually to each word (and n-gram) vector (both source and target vectors). Formally, the training objective function (3) then becomes

$$\min_{\boldsymbol{U},\boldsymbol{V}} \sum_{S \in \mathcal{C}} \sum_{w_t \in S} q_p(w_t) \Bigg(\Big(\ell\big(\boldsymbol{u}_{w_t}^\top \boldsymbol{v}_{S\setminus\{w_t\}}\big) + \tau(\|\boldsymbol{u}_{w_t}\|_1 + \|\boldsymbol{v}_{S\setminus\{w_t\}}\|_1) \Big) + |N_{w_t}| \sum_{w' \in \mathcal{V}} q_n(w') \Big(\ell\big(-\boldsymbol{u}_{w'}^\top \boldsymbol{v}_{S\setminus\{w_t\}}\big) + \tau(\|\boldsymbol{u}_{w'}\|_1) \Big) \Bigg) \tag{4}$$

where τ is the regularization parameter.

Now, in order to minimize a function of the form $f(\mathbf{z}) + g(\mathbf{z})$ where $g(\mathbf{z})$ is not differentiable over the domain, we can use the basic proximal-gradient scheme. In this iterative method, after doing a gradient descent step on $f(\mathbf{z})$ with learning rate α, we update $\mathbf{z}$ as

$$\mathbf{z}_{n+1} = prox_{\alpha,g}(\mathbf{z}_{n+\frac{1}{2}}) \tag{5}$$

where $prox_{\alpha,g}(\mathbf{x}) = \arg\min_{\mathbf{y}} \{g(\mathbf{y}) + \frac{1}{2\alpha}\|\mathbf{y}-\mathbf{x}\|_2^2\}$ is called the proximal function (Rockafellar, 1976) of g with α being the proximal parameter and $\mathbf{z}_{n+\frac{1}{2}}$ is the value of $\mathbf{z}$ after a gradient (or SGD) step on $\mathbf{z}_n$.

In our case, $g(\mathbf{z}) = \|\mathbf{z}\|_1$ and the corresponding proximal operator is given by

$$prox_{\alpha,g}(\mathbf{x}) = sign(\mathbf{x}) \odot \max(|\mathbf{x}_n| - \alpha, 0) \tag{6}$$

where $\odot$ corresponds to element-wise product.

Similar to the proximal-gradient scheme, in our case we can optionally use the thresholding operator on the updated word and n-gram vectors after an SGD step. The soft thresholding parameter used for this update is $\frac{\tau \cdot lr'}{|R(S\setminus\{w_t\})|}$ and $\tau \cdot lr'$ for the source and target vectors respectively where lr' is the current learning rate, τ is the $L1$ regularization parameter and S is the sentence on which SGD is being run.

We observe that $L1$ regularization using the proximal step gives our models a small boost in performance. Also, applying the thresholding operator takes only $|R(S \setminus \{w_t\})| \cdot h$ floating point operations for the updating the word vectors corresponding to the sentence and $(|N|+1) \cdot h$ for updating the target

as well as the negative word vectors, where $|N|$ is the number of negatives sampled and h is the embedding dimension. Thus, performing $L1$ regularization using soft-thresholding operator comes with a small computational overhead.

We set τ to be 0.0005 for both the Wikipedia and the Toronto Book Corpus unigrams + bigrams models.

C Performance comparison with Sent2Vec models trained on different corpora

Data	Model	MSRP (Acc / F1)	MR	CR	SUBJ	MPQA	TREC	Average
Unordered Sentences: (Toronto Books)	**Sent2Vec uni.**	72.2 / 80.3	75.1	**80.2**	90.6	86.3	83.8	81.4
	Sent2Vec uni. + bi.	72.5 / 80.8	75.8	**80.3**	91.2	85.9	86.4	82.0
	Sent2Vec uni. + bi. $L1$-reg	71.6 / 80.1	76.1	**80.9**	91.1	86.1	86.8	82.1
Unordered sentences: Wikipedia (69 million sentences; 1.7 B words)	**Sent2Vec uni.**	71.8 / 80.2	**77.3**	**80.3**	92.0	**87.4**	85.4	82.4
	Sent2Vec uni. + bi.	72.4 / 80.8	**77.9**	**80.9**	92.6	86.9	89.2	83.3
	Sent2Vec uni. + bi. $L1$-reg	73.6 / 81.5	**78.1**	**81.5**	92.8	**87.2**	87.4	83.4
Unordered sentences: Twitter (1.2 billion sentences; 19.7 B words)	**Sent2Vec uni.**	71.5 / 80.0	**77.1**	**81.3**	90.8	**87.3**	85.4	82.2
	Sent2Vec uni. + bi.	72.4 / 80.6	**78.0**	**82.1**	91.8	86.7	89.8	83.5
Other structured Data Sources	CaptionRep BOW	73.6 / 81.9	61.9	69.3	77.4	70.8	72.2	70.9
	CaptionRep RNN	72.6 / 81.1	55.0	64.9	64.9	71.0	62.4	65.1
	DictRep BOW	73.7 / 81.6	71.3	75.6	86.6	82.5	73.8	77.3
	DictRep BOW+embs	68.4 / 76.8	76.7	78.7	90.7	87.2	81.0	80.5
	DictRep RNN	73.2 / 81.6	67.8	72.7	81.4	82.5	75.8	75.6
	DictRep RNN+embs.	66.8 / 76.0	72.5	73.5	85.6	85.7	72.0	76.0

Table 6: Comparison of the performance of different Sent2Vec models with different semi-supervised/supervised models on different **downstream supervised evaluation** tasks. An underline indicates the best performance for the dataset and Sent2Vec model performances are bold if they perform as well or better than all other non-Sent2Vec models, including those presented in Table 1.

	STS 2014						SICK 2014	Average
Model	News	Forum	WordNet	Twitter	Images	Headlines	Test + Train	
Sent2Vec book corpus uni.	.62/.67	**.49/.49**	.75/.72.	.70/**.75**	**.78/.82**	.61/.63	**.61**/.70	**.65**/.68
Sent2Vec book corpus uni. + bi.	.62/.67	**.51/.51**	.71/.68	.70/**.75**	.75/.79	.59/.62	**.62**/.70	**.65**/.67
Sent2Vec book corpus uni. + bi. $L1$-reg	.62/.68	**.51/.52**	.72/.70	.69/**.75**	.76/.81	.60/.63	**.62**/.71	**.66**/.68
Sent2Vec wiki uni.	.66/.71	.47/**.47**	.70/.68	.68/.72	.76/.79	**.63/.67**	**.64**/.71	**.65**/.68
Sent2Vec wiki uni. + bi.	.68/**.74**	**.50/.50**	.66/.64	.67/.72	.75/.79	.62/**.67**	**.63**/.71	**.65**/.68
Sent2Vec wiki uni. + bi. $L1$-reg	**.69/.75**	**.52/.52**	.72/.69	.67/.72	.76/.80	.61/**.66**	**.63/.72**	**.66**/.69
Sent2Vec twitter uni.	.67/**.74**	**.52/.53**	.75/.72	**.72/.78**	.77/.81	**.64/.68**	**.62**/.71	**.67/.71**
Sent2Vec twitter uni. + bi.	.68/**.74**	**.54/.54**	.72/.69	.70/**.77**	.76/.79	.62/**.67**	**.63/.72**	**.66/.70**
CaptionRep BOW	.26/.26	.29/.22	.50/.35	.37/.31	.78/.81	.39/.36	.45/.44	.54/.62
CaptionRep RNN	.05/.05	.13/.09	.40/.33	.36/.30	.76/.82	.30/.28	.36/.35	.51/.59
DictRep BOW	.62/.67	.42/.40	.81/.81	.62/.66	.66/.68	.53/.58	.61/.63	.58/.66
DictRep BOW + embs.	.65/.72	.49/.47	.85/.86	.67/.72	.71/.74	.57/.61	.61/.70	.62/.70
DictRep RNN	.40/.46	.26/.23	.78/.78	.42/.42	.56/.56	.38/.40	.47/.49	.49/.55
DictRep RNN + embs.	.51/.60	.29/.27	.80/.81	.44/.47	.65/.70	.42/.46	.52/.56	.49/.59

Table 7: **Unsupervised Evaluation**: Comparison of the performance of different Sent2Vec models with semi-supervised/supervised models on Spearman/Pearson correlation measures. An underline indicates the best performance for the dataset and Sent2Vec model performances are bold if they perform as well or better than all other non-Sent2Vec models, including those presented in Table 2.

D Dataset Description

	STS 2014						SICK 2014	Wikipedia	Twitter	Book Corpus
Sentence Length	News	Forum	WordNet	Twitter	Images	Headlines	Test + Train	Dataset	Dataset	Dataset
Average	17.23	10.12	8.85	11.64	10.17	7.82	9.67	25.25	16.31	13.32
Standard Deviation	8.66	3.30	3.10	5.28	2.77	2.21	3.75	12.56	7.22	8.94

Table 8: Average sentence lengths for the datasets used in the comparison.

Learning Domain Representation for Multi-Domain Sentiment Classification

Qi Liu[1], **Yue Zhang**[1] **and Jiangming Liu**[2]
Singapore University of Technology and Design, Singapore[1]
University of Edinburgh, UK[2]
{qi_liu, yue_zhang}@sutd.edu.sg
jiangming.liu@ed.ac.uk

Abstract

Training data for sentiment analysis are abundant in multiple domains, yet scarce for other domains. It is useful to leveraging data available for all existing domains to enhance performance on different domains. We investigate this problem by learning domain-specific representations of input sentences using neural network. In particular, a descriptor vector is learned for representing each domain, which is used to map adversarially trained domain-general Bi-LSTM input representations into domain-specific representations. Based on this model, we further expand the input representation with exemplary domain knowledge, collected by attending over a memory network of domain training data. Results show that our model outperforms existing methods on multi-domain sentiment analysis significantly, giving the best accuracies on two different benchmarks.

1 Introduction

Sentiment analysis has received constant research attention due to its importance to business (Pang et al., 2002; Hu and Liu, 2004; Choi and Cardie, 2008; Socher et al., 2012; Vo and Zhang, 2015; Tang et al., 2014). For multiple domains, such as movies, restaurants and digital products, manually annotated datasets have been made available. A useful research question is how to leverage resources available across *all* domains to improve sentiment classification on *a certain* domain.

One naive domain-agnostic baseline is to combine all training data, ignoring domain differences. However, domain knowledge is one valuable source of information available. To utilize this, there has been recent work on *domain-aware* models via multi-task learning (Liu et al., 2016; Nam and Han, 2016), building an output layer for each domain while sharing a representation network. Given an input sentence and a specific test domain, the output layer of the test domain is chosen for calculating the output.

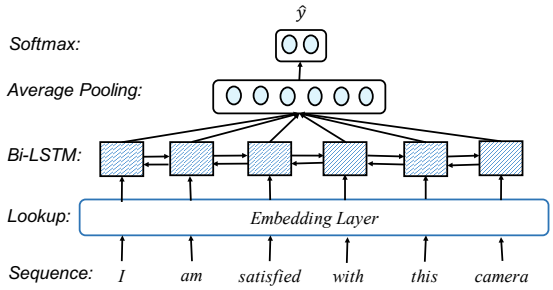
These methods have been shown to improve over the naive domain-agnostic baseline. However, a limitation is that outputs for different domains are constructed using the same domain-agnostic input representation, which leads to weak utilization of domain knowledge. For different domains, sentiment words can differ. For example, the word "beast" can be a positive indicator of camera quality, but irrelevant to restaurants or movies. Also, "easy" is frequently used in the electronics domain to express positive sentiment (e.g. the camera is easy to use), while expressing negative sentiment in the movie domain (e.g. the ending of this movie is easy to guess).

We address this issue by investigating a model that learns domain-specific input representations for multi-domain sentiment analysis. In particular, given an input sentence, our model first uses a bi-directional LSTM to learn a general sentence-level representation. For better utilizing data from all domains, we use adversarial training (Ganin and Lempitsky, 2015; Goodfellow et al., 2014) on the Bi-LSTM representation.

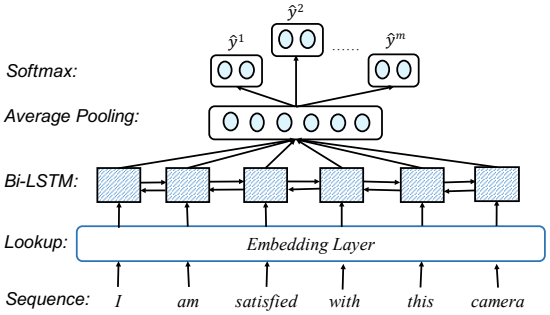
The general sentence representation is then mapped into a domain-specific representation by attention over the input sentence using explicitly learned *domain descriptors*, so that the most salient parts of the input are selected for the specific domain for sentiment classification. Some examples are shown in Figure 2, where our model pays attention to word "engaging" for movie reviews, but not for laptops, restaurants or cameras. Similarly, the word "beast" receives attention for laptops and cameras, but not for restaurants or movies.

In addition to the domain descriptors, we further introduce a memory network for explicitly representing *domain knowledge*. Here domain knowl-

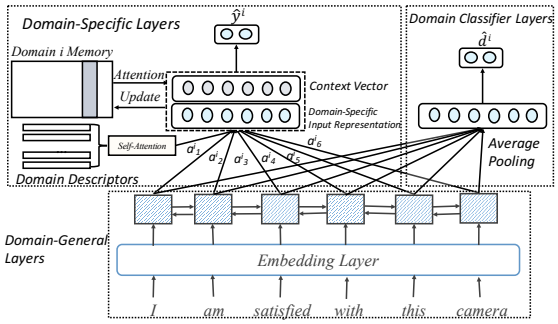
Proceedings of NAACL-HLT 2018, pages 541–550
New Orleans, Louisiana, June 1 - 6, 2018.



(a) **Mix**: shared parameters for all domains.



(b) **Multi**: shared input representations and domain-specific prediction layers.



(c) Our model: domain knowledge is better utilized by domain descriptors, memories and adversarial training.

Figure 1: Models.

edge refers to example training data in a specific domain, which can offer useful background context. For example, given a sentence 'Keep cool if you think it's a wonderful life will be a heart-warming tale about life like finding nemo', algorithms can mistakenly classify it as positive based on 'wonderful' and 'heartwarming', ignoring the fact that 'it's a wonderful life' is a movie. In this case, necessary domain knowledge revealed in other sentences, such as 'The last few minutes of the movie: it's a wonderful life don't cancel out all the misery the movie contained' is helpful. Given a domain-specific input representation, we make attention over the domain knowledge memory network to obtain a background context vector, which is used in conjunction with the input representation for sentiment classification.

Results on two real-world datasets show that our model outperforms the aforementioned multi-task learning methods for domain-aware training, and also generalizes to unseen domains. Our code is released[1].

2 Problem Definition

Formally, we assume the existence of m sentiment datasets $\{D_i\}_{i=1}^m$, each being drawn from a domain i. D_i contains $|D_i|$ data points (s_j^i, d_i, y_j^i), where s_j^i is a sequence of words $w_1, w_2 ... w_{|s_j^i|}$, each being drawn from a vocabulary V, y_j^i indicates the sentiment label (e.g. $y_j^i \in \{-1, +1\}$ for binary sentiment classification) and d_i is a domain indicator (since we use 1 to m to number each domain, $d_i = i$). The task is to learn a function f which maps each input (s_j^i, d_i) to its corresponding sentiment label y_j^i. The challenge of the task lies in how to improve the generalization performance of mapping function f both in-domain and cross-domain by exploring the correlations between different domains.

3 Baselines

3.1 Domain-Agnostic Model

One naive baseline solution ignores the domain characteristics when learning f. It simply combines the datasets $\{D_i\}_{i=1}^m$ into one and learns a single mapping function f. We refer to this baseline as **Mix**, which is depicted in Figure 1 (a).

Given an input s_j^i, its word sequence $w_1, w_2 ... w_{|s_j^i|}$ is fed into a word embedding layer to obtain embedding vectors $x_1, x_2 ... x_{|s_j^i|}$. The word embedding layer is parameterized by an embedding matrix $E_w \in R^{K \times |V|}$, where K is the embedding dimension.

Bidirectional LSTM: To acquire a semantic representation of input s_j^i, a bidirectional extension (Graves and Schmidhuber, 2005) of Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) is applied to capture sentence-level semantics both left-to-right and right-to-left. As a result, two sequences of hidden states are obtained, denoted as $\overrightarrow{h}_1, \overrightarrow{h}_2 ... \overrightarrow{h}_{|s_j^i|}$ and $\overleftarrow{h}_1, \overleftarrow{h}_2 ... \overleftarrow{h}_{|s_j^i|}$, respectively. We concatenate $\overrightarrow{h}_t$

[1] https://github.com/leuchine/multi-domain-sentiment

and $\overleftarrow{h}_t$ at each time step to obtain the hidden states $h_1, h_2 ... h_{|s^i_j|}$, which are of sizes $2K$.

Output Layer: Average pooling (Boureau et al., 2010) is applied on the hidden states $h_1, h_2 ... h_{|s^i_j|}$ to obtain an input representation I^i_j for s^i_j,

$$I^i_j = \frac{\sum_{t=1}^{|s^i_j|} h_t}{|s^i_j|} \quad (1)$$

Finally, softmax is applied over I^i_j to obtain a probability distribution of all sentiment labels. During training, cross entropy is used as loss function, denoted as $L(f(s^i_j), y^i_j)$ for data points (s^i_j, d_i, y^i_j), and AdaGrad (Duchi et al., 2011) is applied to update parameters.

3.2 Multi-Domain Training

We build a second baseline for domain-aware sentiment analysis. A state-of-the-art architecture (Liu et al., 2016; Nam and Han, 2016) is used as depicted in Figure 1 (b), where m mapping functions f_i are learned for each domain. Given the input representation I^i_j obtained in Equation 1, multi-task learning is conducted, where each domain has a domain-specific set of parameters for softmax to predict sentiment labels with shared input representation layers. The input domain indicator d_i instructs which set of softmax parameters to use here and each domain has its own cross entropy loss $L_i(f_i(s^i_j, d_i), y^i_j)$ for data points (s^i_j, d_i, y^i_j). We denote this baseline as **Multi**.

4 Method

4.1 Domain-Aware Input Representation

The above baseline **Multi** achieves state-of-the-art performance for multi-domain sentiment analysis (Liu et al., 2016), yet the domain indicator d_i is used solely to select softmax parameters. As a result, domain knowledge is hidden and under-utilized. Similar to **Mix** and **Multi**, we use a Bi-LSTM to learn representations shared across domains. However, we introduce domain-specific layers to better capture domain characteristics as shown in Figure 1 (c).

Different domains have their own sentiment lexicons and domain differences largely lie in which words are relatively more important for deciding the sentiment signals. We use the neural attention mechanism (Bahdanau et al., 2014) to select words, obtaining domain-specific input representations.

In our model, *domain descriptors* are introduced to explicitly capture domain characteristics, which are parametrized by a matrix $N \in R^{2K \times m}$. Each domain descriptor corresponds to one column of N and has a length of $2K$, the same as the bidirectional LSTM hidden states h_t. This matrix is automatically learned during training.

Given an input (s^i_j, d_i), we apply an embedding layer and Bi-LSTM to generate its domain-general representation $h_1, h_2, ..., h_{|s^i_j|}$ and use the corresponding domain descriptor N_i to weigh $h_1, h_2, ..., h_{|s^i_j|}$ for obtaining a domain-specific representation. To this end, there are two most commonly used attention mechanisms: additive attention (Bahdanau et al., 2014) and dot product attention (Ashish Vaswani, 2017). We choose additive attention here, which utilizes a feed-forward network with a single hidden layer, since it achieves better accuracies in our development. The input representation I^i_j becomes a weighted sum of hidden states:

$$I^i_j = \sum_{t=1}^{|s^i_j|} a^i_{jt} h_t \quad s.t. \sum_{t=1}^{|s^i_j|} a^i_{jt} = 1 \quad (2)$$

The weight a^i_{jt} reflects the similarity between the domain i's descriptor N_i and the hidden state h_t. a^i_{jt} is evaluated as:

$$l^i_{jt} = v^T tanh(PN_i + Qh_t)$$
$$a^i_{jt} = \frac{exp(l^i_{jt})}{\sum_{p=1}^{|s^i_j|} exp(l^i_{jp})} \quad (3)$$

Here $P \in R^{4K \times 2K}$, $Q \in R^{4K \times 2K}$ and $v \in R^{4K}$ are parameters of additive attention. P and Q linearly project N_i and h_t to a hidden layer, respectively. The projected space is set as $4K$ empirically, since we find it beneficial to project the vectors into a larger layer. v serves as the output layer. Softmax is applied to normalize l^i_{jt}. We name this method **DSR** for learning domain-specific representations.

4.2 Self-Attention over Domain Descriptors

DSR uses a single domain descriptor to attend over input words. However, relations between domains are not considered (e.g. sentiment lexicons for domain 'camera' are more similar to the lexicons of domain 'laptop' than those of domain

'restaurant'). To model the interaction between domains, a self-attention layer is applied using dot product attention empirically, as shown in Figure 1 (c):

$$N_i^{new} = N\ softmax(N^T N_i) \quad (4)$$

We compute dot products between N_i and every domain descriptors. The dot products are normalized using the softmax function, and N_i^{new} is a weighted sum of all domain descriptors. N_i^{new} is used to attend over hidden states, employing Equation 2 and 3. During back propagation training, domain descriptors of similar domains could be updated simultaneously. We name this method **DSR-sa**, which denotes domain-specific representation with self-attention.

4.3 Explicit Domain Knowledge

To further capture domain characteristics, we devise a memory network (Weston et al., 2014; Sukhbaatar et al., 2015; Kumar et al., 2016) framework to explicitly represent domain knowledge. Our memory networks hold example training data of a specific domain for retrieving context data during predictions.

Formally, we use a memory $M^i \in R^{2K \times |D_i|}$ ($|D_i|$ is the total number of training instances of domain i) to hold domain-specific representations I_j^i of training instances for the domain i.

Memory Network: We directly set I_j^i as the jth column of the memory M^i. Formally,

$$M_j^i = I_j^i \quad (5)$$

Obtaining A Context Vector Using Background Knowledge: Given an input I_j^i, we generate a context vector C_j^i to support predictions by memory reading:

$$C_j^i = M^i\ softmax((M^i)^T I_j^i) \quad (6)$$

Dot product attention is applied here, which is faster and more space-efficient than additive attention, since it can be implemented using highly optimized matrix multiplication. Dot products are performed between I_j^i and each column of M^i and the scores are normalized using the softmax function. The final context vector is a weighted sum of M^i's columns.

Output: We concatenate the context vector and the domain-specific input representation, feeding the result to softmax layers. Similar to the baseline **Multi**, each domain has its own loss $L_i(f_i(s_j^i, d_i), y_j^i)$. We name this method as **DSR-ctx** for context vector enhancements.

Reducing Memory Size: In the naive implementation, the memory size $|M^i|$ is equal to the total number of saved sequences, which can be very large in practice. We explore two ways to reduce memory size.

(1) Organizing memory by the vocabulary. We set $|M^i| = |V|$, where each memory column of M^i corresponds to a word in the vocabulary. During memory writing, I_j^i updates all the columns that correspond to the words w in its input sequence s_j^i by exponential moving average:

$$M_w^i = decay * M_w^i + (1 - decay) I_j^i$$

In this way, two input representations update the same column of the memory network if and only if they share at least one common word.

(2) Fixing the memory size by clustering. $|M^i|$ is set to a fixed size and I_j^i only updates the memory column that is most similar to I_j^i, i.e. I_j^i only update the column $arg\,max\,(M^i)^T I_j^i$. In this way, semantically similar inputs are clustered and update the same column.

4.4 Adversarial Training

We use embeddings and Bi-LSTM, parametrized by θ_{dg}, to generate domain-general representations. However, the distributions of domain-general representations for all domains can be different (Goodfellow et al., 2014), which contaminates the representations (Liu et al., 2017) and imposes negative effects for in-domain predictions. For cross-domain testing, the discrepancies cause domain shift, which harms prediction accuracies on target domains (Ganin and Lempitsky, 2015). Thus, models that can generate domain-invariant representations for all domains are favorable for utilizing multi-domain datasets.

We incorporate adversarial training to enhance the domain-general representations. As shown in Figure 1 (c), domain classifier layers are introduced, parametrized by θ_{dc}, which predicts how likely the input sequence s_j^i comes from each domain i. We denote its cross entropy loss as $L_{at}(f_{at}(s_j^i), d_i)$ for data points (s_j^i, d_i, y_j^i) from domain i (note that we use d_i as its label instead of input here).

Now consisting of domain-general layers, domain-specific layers and domain classifier lay-

ers, the model is trained by a minimax game. For dataset D_i drawn from domain i, we minimize its loss $L_i(f_i(s^i_j, d_i), y^i_j)$ for sentiment predictions, while maximizing the domain classifier loss $L_{at}(f_{at}(s^i_j), d_i)$, controlled by λ:

$$\min_{\theta_{dg},\theta_{ds}} \sum_{D_i} L_i(f_i(s^i_j, d_i), y^i_j) - \lambda L_{at}(f_{at}(s^i_j), d_i), \tag{7}$$

where θ_{ds} is the set of domain-specific parameters including domain descriptors, attention weights and softmax parameters. We fix θ_{dc} and update θ_{dg} and θ_{ds} here. Its adversarial part maximizes the loss by updating θ_{dc}, while fixing θ_{dg} and θ_{ds}.

$$\max_{\theta_{dc}} \sum_{D_i} L_i(f_i(s^i_j, d_i), y^i_j) - \lambda L_{at}(f_{at}(s^i_j), d_i) \tag{8}$$

Equations 7 and 8 are performed iteratively to generate domain-invariant representations. We name this method **DSR-at**.

5 Experiments

We evaluate the effectiveness of the model both in-domain and cross-domain. The former refers to the setting where the domain of the test data falls into one of the m training data domains, and the latter refers to the setting where the test data comes from one unknown domain.

5.1 Experimental Settings

We conduct experiments on two benchmark datasets. The datasets are balanced, so we use accuracy as the evaluation metric in the experiments.

The dataset 1 contains four domains. The statistics are shown in Table 1 , which also shows the accuracies using baseline method **Mix** trained and tested on each domain. *Camera*[2] consists of reviews with respect to digital products such as cameras and MP3 players (Hu and Liu, 2004). *Laptop* and *Restaurant* are laptop and restaurant reviews, respectively, obtained from SemEval 2015 Task 12[3]. Movie[4] are movie reviews provided by Pang and Lee (2004).

The dataset 2 is Blitzer's multi-domain sentiment dataset (Blitzer et al., 2007), which contains product reviews taken from Amazon.com, including 25 product types (domains) such as books, beauty and music. More statistics can be found at its official website[5].

Domain	Instance	Vocab Size	Accuracy
Camera (CR)	3770	5340	0.802
Laptop (LT)	1907	2837	0.871
Restaurant (RT)	1572	2930	0.783
Movie (M)	10662	18765	0.773

Table 1: Dataset 1 statistics.

Given each dataset, we randomly select 80%, 10% and 10% of the instances as training, development and testing sets, respectively.

5.2 Baselines and Hyperparameters

In addition to the **Mix** baseline, the **Multi** baseline (Liu et al., 2016) and our domain-aware models, **DSR**, **DSR-sa**, **DSR-ctx**, **DSR-at**, we also experiment with the following baselines:

MTRL (Zhang and Yeung, 2012) is a state-of-the-art multi-task learning method with discrete features. The method models covariances between task classifiers, and in turn the covariances regularize task-specific parameters. The feature extraction for **MTRL** follows (Blitzer et al., 2007). We use this baseline to demonstrate the effectiveness of dense features generated by neural models.

MDA (Chen et al., 2012) is a cross-domain baseline, which utilizes marginalized de-noising auto-encoders to learn a shared hidden representation by reconstructing pivot features from corrupted inputs.

FEMA (Yang and Eisenstein, 2015) is a cross-domain baseline, which utilizes techniques from neural language models to directly learn feature embeddings and is more robust to domain shift.

NDA (Kim et al., 2016) is a cross-domain baseline, which uses $m + 1$ LSTMs, where one LSTM captures global information across all m domains and the remaining m LSTM capture domain-specific information.

We set the size of word embeddings K to 300, which are initialized using the word2vec model[6] on news. To obtain the best performance, the parameters are set using grid search based on development results. The dropout ratio is chosen from $[0.3, , 1]$. Learning rate is chosen from

[2] http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html

[3] Since the original dataset targets aspect-level sentiment analysis, we remove the sentences with opposite polarities on different aspects. The remaining sentences are labeled with the unambiguous polarity.

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

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

[6] https://code.google.com/archive/p/word2vec/

Dataset		Method						
Train	Test	MTRL	Mix	Multi	DSR	DSR-sa	DSR-ctx	DSR-at
LT+RT	*LT*	0.817	0.896	0.90	0.908	0.911	0.914	**0.92***
LT+RT	*RT*	0.781	0.820	0.85	0.860	0.859	0.863	**0.883***
LT+M	*LT*	0.825	0.882	0.90	0.887	0.90	0.904	**0.913***
LT+M	*M*	0.743	0.778	0.772	0.788	0.79	0.803	**0.811***
LT+CR	*LT*	0.869	0.904	0.906	0.921	0.915	0.92	**0.925**
LT+CR	*CR*	0.774	0.800	0.802	0.822	0.826	0.832	**0.844***
RT+M	*RT*	0.792	0.830	0.833	0.853	0.86	0.883	**0.9***
RT+M	*M*	0.729	0.765	0.785	0.795	0.801	0.816	**0.83***
RT+CR	*RT*	0.783	0.828	0.822	0.847	0.851	0.878	**0.887***
RT+CR	*CR*	0.756	0.804	0.814	0.812	0.817	0.831	**0.84***
M+CR	*M*	0.745	0.775	0.788	0.798	0.802	0.830	**0.839***
M+CR	*CR*	0.758	0.799	0.811	**0.819**	0.812	0.817	0.812
Average		0.778	0.818	0.832	0.842	0.845	0.857	**0.867***

Table 2: Results using two training domains on dataset 1. * denotes $p < 0.01$ VS. the second best using McNemar's test.

$[0.0001, 0.001, , 1]$. The vocabulary size is chosen from $[6000, 8000, , 16000]$. The batch size is chosen from $[10, , 100]$. λ is chosen from $[0.0001, 0.001, , 1]$. As a result, the mini-batch size, the size of the vocabulary V, dropout rate, learning rate for AdaGra and λ for adversarial training are set to 50, 10000, 0.4, 0.5 and 0.1, respectively. Also, gradient clipping (Pascanu et al., 2013) is adopted to prevent gradient exploding and vanishing during training process. Since all datasets only have thousands of instances, we set memory network sizes as training instance sizes in the experiments.

5.3 Working with Known Domains

In this section, we perform in-domain validations. We first combine two datasets for training and test on each domain's hold-out testing dataset. The results on dataset 1 are shown in Table 2 (the results on Blitzer's dataset exhibit similar results and are omitted due to space constraints).

The accuracies of **MTRL** are significantly lower than the neural models, which demonstrates the effectiveness of dense features over discrete features. The baseline **Mix** improves the average accuracy from 0.778 to 0.818, and most multi-domain training accuracies are better compared to single-domain training in Table 1. **Mix** simply combines the two datasets for trainings and ignores domain characteristics, yet improves over single dataset training. This demonstrates that more data reduces over-fitting and leads to better generalization capabilities. **Multi** further improves the average accuracy by 1.4%, which confirms the effectiveness of utilizing domain information.

Among our models, **DSR** further improves the accuracy over **Multi** by 1%, which confirms the effectiveness of domain-specific input representations in multi-domain sentiment analysis. **DSR-sa** slightly outperforms **DSR** by 0.03%. Adopting an additional self-attention layer, **DSR-sa** trains similar domain descriptors together, thus better modeling domain relations, which will be further studied in Section 5.5.2. **DSR-ctx** outperforms **DSR-sa** by 1.2%, which demonstrates the effectiveness of memory networks in utilizing domain-specific example knowledge. **DSR-at** gives significantly the best results, confirming that domain-invariant representations achieved by adversarial training indeed benefit in-domain training. The results are significant using McNeymar's test.

The results combining all the 4 domains and the 25 domains of the two datasets are shown in the 'In domain' sections of Table 3 and Table 4, respectively. Here the models are trained using all domains' training data, and tested on each domain's hold-out test data. Similar patterns are observed as in Table 2 and **DSR-at** achieves significantly the best accuracies (0.867 and 0.907 for the two datasets, respectively).

5.4 Working with Unknown New Domains

We validate the algorithms cross-domain. For dataset 1, models are trained on three domains, yet validated and tested on the other domain. For dataset 2, models are trained on 24 domains, yet validated and tested on the 25th.

Since **DSR-at** has m outputs (one for each training domain), we adopt an ensemble approach to obtain a single output for unknown test domains. In particular, since the domain classifier outputs probabilities on how likely the test data come from each training domain, we use these probabilities as weights to average the m outputs.

For **NDA**, **Multi**, **DSR** and **DSR-sa** and **DSR-ctx**, we use average pooling to combine the m outputs. Since **MDA** and **FEMA** are devised to train on a single source domain, we combine the training data of m domains for training.

The results are shown in the 'Cross domain' section of Table 3 and Table 4, respectively. One observation is that cross-domain accuracies are worse than in-domain accuracies, showing challenges in unknown-domain testing.

Contrast between our models and **FEMA/NDA** shows the advantage of leveraging resources from all domains, versus a single source domain for cross-domain modelling. Among the baselines,

	In domain							Cross domain									
Dataset	MTRL	Mix	Multi	DSR	DSR-sa	DSR-ctx	DSR-at	MTRL	Mix	MDA	Multi	FEMA	NDA	DSR	DSR-sa	DSR-ctx	DSR-at
LT	0.813	0.831	0.900	0.897	0.902	0.898	**0.915***	0.763	0.792	0.801	0.808	0.811	0.816	0.822	0.823	0.854	**0.878***
RT	0.776	0.801	0.825	0.841	0.845	0.855	**0.870***	0.772	0.786	0.789	0.779	0.774	0.776	0.78	0.784	0.814	**0.847***
M	0.800	0.803	0.783	0.807	0.812	0.820	**0.828***	0.616	0.636	0.642	0.668	0.679	0.684	0.692	0.695	0.725	**0.729**
CR	0.775	0.786	0.819	0.825	0.828	0.836	**0.854***	0.714	0.721	0.736	0.735	0.741	0.745	0.751	0.753	0.789	**0.809***
Average	0.791	0.805	0.832	0.843	0.847	0.852	**0.867***	0.716	0.734	0.742	0.748	0.751	0.755	0.761	0.764	0.796	**0.815***

Table 3: In-domain learning and cross-domain results on dataset 1. * denotes $p < 0.01$ VS. the second best.

	In domain							Cross domain									
Dataset	MTRL	Mix	Multi	DSR	DSR-sa	DSR-ctx	DSR-at	MTRL	Mix	MDA	Multi	FEMA	NDA	DSR	DSR-sa	DSR-ctx	DSR-at
Apparel	0.883	0.912	0.921	0.927	0.928	0.92	**0.938***	0.828	0.843	0.863	0.854	0.865	0.873	0.882	0.899	0.896	**0.909***
Electronics	0.853	0.881	**0.899**	0.884	0.879	0.883	0.891	0.804	0.826	0.836	0.849	0.845	0.834	0.857	0.859	0.861	**0.875***
Office	0.863	0.88	0.89	0.903	0.914	0.925	**0.933***	0.824	0.825	0.818	0.824	0.843	0.839	0.854	0.876	0.883	**0.894***
Automotive	0.842	0.864	0.873	0.886	0.891	0.902	**0.917***	0.791	0.786	0.791	0.797	0.816	0.826	0.835	0.847	0.857	**0.867***
Gourmet	0.814	0.838	0.84	0.852	0.856	0.858	**0.863***	0.777	0.775	0.764	0.784	0.796	0.803	0.814	0.826	**0.832**	0.828
Outdoor	0.853	0.889	0.899	0.903	0.907	0.915	**0.927***	0.785	0.796	0.805	0.815	0.836	0.829	0.856	0.861	0.867	**0.887***
Baby	0.816	0.853	0.86	0.875	0.877	0.892	**0.91***	0.803	0.816	0.814	0.821	0.834	0.84	0.845	0.878	0.873	**0.895***
Grocery	0.862	0.886	0.898	0.907	0.911	0.917	**0.933***	0.806	0.817	0.826	0.846	0.846	0.862	0.88	0.873	0.865	**0.886***
Software	0.851	0.876	0.88	0.893	0.898	0.904	**0.92***	0.795	0.811	0.816	0.836	0.845	0.836	0.85	0.862	0.884	**0.897***
Beauty	0.816	0.843	0.8567	0.862	0.867	0.864	**0.889***	0.756	0.768	0.775	0.785	0.795	0.804	0.812	0.812	0.838	**0.851***
Health	0.871	0.901	0.904	0.896	0.897	0.896	**0.907**	0.785	0.807	0.819	0.832	0.845	0.848	0.843	0.834	0.857	**0.871***
Sports	0.851	0.883	0.899	0.889	0.882	0.895	**0.9**	0.759	0.768	0.775	0.784	0.816	0.819	0.821	0.836	0.848	**0.864***
Book	0.743	0.803	0.79	0.804	0.809	0.815	**0.822***	0.694	0.705	0.716	0.723	0.745	0.743	0.751	0.758	0.779	**0.798***
Jewelry	0.816	0.891	0.881	0.893	0.891	0.894	**0.909***	0.762	0.769	0.774	0.785	0.795	0.808	0.815	0.835	0.857	**0.874***
Camera	0.912	0.937	0.968	0.966	0.959	0.968	**0.989***	0.869	0.878	0.886	0.896	0.894	0.908	0.917	0.925	0.942	**0.963***
Kitchen	0.815	0.858	0.863	0.875	0.887	0.894	**0.913***	0.759	0.768	0.775	0.776	0.794	0.818	0.826	0.856	0.865	**0.884** *
Toy	0.823	0.863	0.875	0.881	0.884	0.88	**0.892***	0.814	0.824	0.815	0.803	0.813	0.832	0.826	0.843	0.845	**0.857***
Phone	0.879	0.936	0.94	0.943	**0.949***	0.941	0.933	0.805	0.813	0.808	0.818	0.821	0.833	0.836	0.856	0.874	**0.894***
Magazine	0.835	0.874	0.872	0.883	0.895	0.917	**0.937***	0.805	0.819	0.817	0.816	0.83	0.841	0.845	0.857	0.871	**0.896***
Video	0.851	0.873	0.882	0.891	0.896	0.912	**0.925***	0.754	0.774	0.794	0.795	0.815	0.822	0.834	0.845	0.855	**0.875***
Games	0.867	0.886	0.89	0.883	0.886	0.887	**0.9***	0.681	0.684	0.708	0.718	0.723	0.734	0.746	0.765	**0.781**	0.778
Music	0.752	0.782	0.8	0.798	0.8	0.798	**0.81***	0.775	0.769	0.779	0.784	0.795	0.824	0.815	0.823	0.842	**0.858***
Dvd	0.795	0.826	0.834	0.847	0.854	0.867	**0.889***	0.801	0.794	0.804	0.794	0.814	0.827	0.835	0.845	0.851	**0.875***
Instrument	0.873	0.943	**0.957***	0.896	0.906	0.898	0.9	0.814	0.805	0.813	0.815	0.825	0.836	0.833	0.835	0.845	**0.865***
Tools	0.887	0.915	0.931	0.928	0.93	0.932	**0.94***	0.805	0.814	0.828	0.835	0.846	0.857	0.864	0.866	0.873	**0.897***
Average	0.841	0.875	0.884	0.887	0.89	0.895	**0.907***	0.786	0.794	0.801	0.807	0.82	0.827	0.835	0.847	0.858	**0.873***

Table 4: In-domain learning and cross-domain results on dataset 2. * denotes $p < 0.01$ VS. the second best.

NDA also considered domain-specific representations. On the other hand, it duplicates the full set of model parameters for each domain, yet underperforms **DSR** and **DSR-sa**, which records only one domain descriptor vector for each domain. The contrast shows the advantages of learning domain descriptors explicitly in terms of both efficiency and accuracy.

Similar to the known domain results, **DST-sa** and **DSR-ctx** further improve upon **DSR** and **DSR-sa**, showing the effectiveness of domain memory and adversarial learning. On both datasets, **DSR-at** achieves significantly the best performances, which shows the advantages of domain-invariant representations for unknown-domain testing.

5.5 Case Study

5.5.1 Input Attention

To obtain a better understanding of input attention with domain descriptors, we examine the attention weights of inputs and three examples are displayed in Figure 2, where the x axis denotes the four domains from the first dataset and the y axis shows the words.

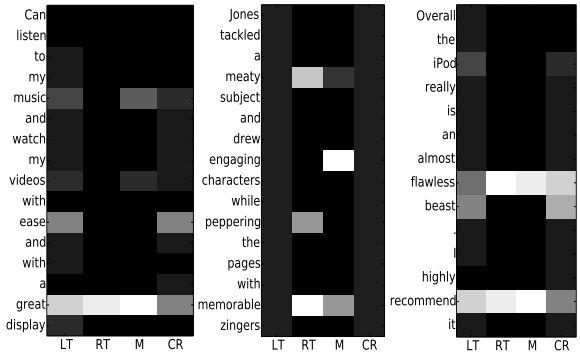
In Figure 2 (a), the domain-specific word 'ease' is only selected for the domains *LT* and *CR*, while the domain-independent word 'great' is salient in all domains. Similarly, in Figure 2 (b), 'meaty' and 'engaging' are only salient in *RT* and *M*, respectively. In Figure 2 (c), the domain-specific word 'beast' is chosen in **LT** and **CR**.

These confirm the effectiveness of input attention and **DSR-ctx** has the capability to pick out sentiment lexicons in conformity with domain characteristics.

5.5.2 Domain Descriptors

With the self-attention layer, one interesting question is whether learned domain descriptors can reflect domain similarities/dissimilarities.

We take out the twenty-five domain descriptors for Blitzer's dataset and calculate the cosine similarities between each pair. Also, we calculate the cosine similarities of twenty-five domains based on unigram and bigram representations for ground truth. Pearson correlation coefficient is used to measure the correlations between two sets of cosine values. The final score is 0.796, which shows that domain descriptor similarities can serve as indicators for domain similarities.



(a) Example1 (b) Example2 (c) Example3

Figure 2: Attention values (0: black, 1: white).

This place blew me away. By far my new favorite restaurant on the upper-east side.
(a1) This is one of my favorite spot, very relaxing. The food is great all the times. Celebrated my engagement and my wedding here. It was very well organized.
(a2) This is one of my favorite restaurants and it is not to be missed.
(a3) I didn't complain. I liked the atmosphere so much.
I started accessing and transferring files to find it to be extremely slow.
(b1) Only thing I don't like about it is slow in changing apps, boot up, and sometime it has problem connect through bluetooth.
(b2) I must say, this one is quite slow to open an application.
(b3) The subscription files are still a little slower to transfer, but it 's only by about 10% or so.
Keep cool if you think it's a wonderful life will be a heartwarming tale about life like finding nemo.
(c1) I heard so much about It's a wonderful life's happy ending and I just wasn't prepared for so much misery.
(c2) The last few minutes of the movie: its a wonderful life dont cancel out all the misery the movie contained.
(c3) It's a wonderful life was so incredibly over-sentimental and highly manipulative.

Table 5: Memory Network Attention.

5.5.3 Memory Network Attention

We further study the attention of memory networks by randomly picking instances in the test sets and listing the context instances with the greatest attention weights obtained from Equation 6. The results of three test instances and their context instances are shown in Table .

One observation is that semantically similar instances are selected to provide extra knowledge for predictions (e.g. a1, a2, b3, c1, c2, c3). Another observation is that the sentiment polarities between test instances and selected context instances are usually the same. We conclude that the memory networks are capable of selecting instructive instances for facilitating predictions.

6 Related Work

Domain Adaptation (Blitzer et al., 2007; Titov, 2011; Yu and Jiang, 2015) adapts classifiers trained on a source domain to an unseen target domain. One stream of work focuses on learning a general representation for different domains based on the co-occurrences of domain-specific and domain-independent features (Blitzer et al., 2007; Pan et al., 2011; Yu and Jiang, 2015; Yang et al., 2017). Another stream of work tries to identify domain-specific words to improve cross-domain classification (Bollegala et al., 2011; Li et al., 2012; Zhang et al., 2014; Qiu and Zhang, 2015). Different from previous work, we utilize *multiple* source domains for cross-domain validation, which makes our method more general and domain-aware.

Multi-domain Learning jointly learn multiple domains to improve generalization. One strand of work (Dredze and Crammer, 2008; Saha et al., 2011; Zhang and Yeung, 2012) uses covariance matrix to model domain relatedness, jointly learns domain-specific parameters and domain-independent parameters of linear classifiers. Another strand of work (Liu et al., 2016; Nam and Han, 2016) adopts neural network with shared input layers and multiple output layers for prediction. Our work belongs to the latter, yet we introduce domain descriptor matrix and memory networks to better capture domain characteristics and achieve better performance.

Memory Networks reason with inference components combined with a long-term memory component. Weston et al. (2014) devise a memory network to explicitly store the entire input sequences for question answering. An end-to-end memory network is further proposed by Sukhbaatar et al. (2015) by storing embeddings of input sequences, which requires much less supervision compared to Weston et al. (2014). Kumar et al. (2016) introduces a general dynamic memory network, which iteratively attends over episodic memories to generate answers. Xiong et al. (2016) extends Kumar et al. (2016) by introducing a new architecture to cater image inputs and better capture input dependencies. In similar spirits, our memory network stores the domain-specific training instances for obtaining context knowledge.

7 Conclusion

We investigated domain representations in multi-task learning for multi-domain sentiment analysis, showing that leveraging domain descriptors, examples and adversarial training to learn domain representations give significant improve-

ments compared with strong multi-task learning baselines.

Acknowledgments

We thank the anonymous reviewers for their insightful comments. Yue Zhang is the corresponding author.

References

Noam Shazeer Niki Parmar Ashish Vaswani. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

John Blitzer, Mark Dredze, Fernando Pereira, et al. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45nd ACL*. volume 7, pages 440–447.

Danushka Bollegala, David Weir, and John Carroll. 2011. Using multiple sources to construct a sentiment sensitive thesaurus for cross-domain sentiment classification. In *Proceedings of the 49th ACL*. Association for Computational Linguistics, pages 132–141.

Y-Lan Boureau, Jean Ponce, and Yann LeCun. 2010. A theoretical analysis of feature pooling in visual recognition. In *ICML*. pages 111–118.

Minmin Chen, Zhixiang Xu, Kilian Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. *arXiv preprint arXiv:1206.4683* .

Yejin Choi and Claire Cardie. 2008. Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 793–801.

Mark Dredze and Koby Crammer. 2008. Online methods for multi-domain learning and adaptation. In *EMNLP*. pages 689–697.

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

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*. pages 1180–1189.

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

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks* pages 602–610.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. MIT Press, volume 9, pages 1735–1780.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD*. ACM, pages 168–177.

Young-Bum Kim, WA Redmond, Karl Stratos, and Ruhi Sarikaya. 2016. Frustratingly easy neural domain adaptation. In *Proceedings of COLING 2016*.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *ICML*. pages 1378–1387.

Fangtao Li, Sinno Jialin Pan, Ou Jin, Qiang Yang, and Xiaoyan Zhu. 2012. Cross-domain co-extraction of sentiment and topic lexicons. In *ACL*. Association for Computational Linguistics, pages 410–419.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101* .

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. *arXiv preprint arXiv:1704.05742* .

Hyeonseob Nam and Bohyung Han. 2016. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*. pages 4293–4302.

Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. 2011. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks* 22(2):199–210.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *ACL*. Association for Computational Linguistics, page 271.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *ACL*. Association for Computational Linguistics, pages 79–86.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. *ICML* 28:1310–1318.

Likun Qiu and Yue Zhang. 2015. Word segmentation for chinese novels. In *AAAI*. pages 2440–2446.

Avishek Saha, Piyush Rai, and Hal Daumé III Suresh Venkatasubramanian. 2011. Online learning of multiple tasks and their relationships. *update* .

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*. Association for Computational Linguistics, pages 1201–1211.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*. pages 2440–2448.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 1555–1565.

Ivan Titov. 2011. Domain adaptation by constraining inter-domain variability of latent feature representation. In *Proceedings of the 49th ACL*. Association for Computational Linguistics, pages 62–71.

Duy-Tin Vo and Yue Zhang. 2015. Target-dependent twitter sentiment classification with rich automatic features. In *IJCAI*. pages 1347–1353.

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

Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In *ICML*. pages 2397–2406.

Yi Yang and Jacob Eisenstein. 2015. Unsupervised multi-domain adaptation with feature embeddings. In *HLT-NAACL*. pages 672–682.

Zhilin Yang, Ruslan Salakhutdinov, and William W Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. *arXiv preprint arXiv:1703.06345* .

Jianfei Yu and Jing Jiang. 2015. Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification. *Conference on Empirical Methods in Natural Language Processing* pages 236–246.

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2014. Type-supervised domain adaptation for joint segmentation and pos-tagging. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. pages 588–597.

Yu Zhang and Dit-Yan Yeung. 2012. A convex formulation for learning task relationships in multi-task learning. *arXiv preprint arXiv:1203.3536* .

Learning Sentence Representations over Tree Structures for Target-dependent Classification

Junwen Duan, Xiao Ding, Ting Liu*
Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, Harbin, China
{jwduan,xding,tliu}@ir.hit.edu.cn

Abstract

Target-dependent classification tasks, such as aspect-level sentiment analysis, perform fine-grained classifications towards specific targets. Semantic compositions over tree structures are promising for such tasks, as they can potentially capture long-distance interactions between targets and their contexts. However, previous work that operates on tree structures resorts to syntactic parsers or Treebank annotations, which are either subject to noise in informal texts or highly expensive to obtain. To address above issues, we propose a reinforcement learning based approach, which automatically induces target-specific sentence representations over tree structures. The underlying model is a RNN encoder-decoder that explores possible binary tree structures and a reward mechanism that encourages structures that improve performances on downstream tasks. We evaluate our approach on two benchmark tasks: firm-specific cumulative abnormal return prediction (based on formal news texts) and aspect-level sentiment analysis (based on informal social media texts). Experimental results show that our model gives superior performances compared to previous work that operates on parsed trees. Moreover, our approach gives some intuitions on how target-specific sentence representations can be achieved from its word constituents.

1 Introduction

We investigate target-dependent classification problem in this paper, with a special focus on the sentence level. Target-dependent classification aims to identify the fine-grained polarities of sentences towards specific targets, which is challenging but also important for deep text understanding. The definitions of polarity vary across different tasks, which can be *positive* or *negative* in aspect-level sentiment analysis, *favor* or *against* in stance detection, and *rise* or *drop* in financial news analysis towards the stock price movement of a particular firm.

Task	Example
Aspect-level Sentiment Analysis	The **food** is good but the **service** is dreadful.
Stance Detection	I don't care about **global climate change**.
Firm-specific Financial News Analysis	**Nike** sues **Wal-Mart** for Patent Infringement.

Table 1: Samples of target-dependent classification tasks. The targets of interest are in bold.

Table 1 gives examples of three target-dependent classification tasks. We can find that there can be multiple target mentions in the same text scope, which makes it challenging for generic sentence representation approaches. For the first example, a restaurant manager or a potential customer may be interested in both *food* and *service*; however, the sentiment polarities towards the two targets are different. Hence, it would be beneficial for such tasks to tailor the sentence representations with respect to particular targets.

Tree structures are promising for such tasks, as they can potentially capture long-distance dependencies between target words and their contexts (Li et al., 2015). Therefore, it is not surprising to find work that exploits the syntactically parsed trees for learning target-specific sentence representations. Dong et al. (2014) and Chang et al. (2016) adapted the word orders in a parsed tree, depending on their distances to the target entities. Nguyen et al. (2015) extended Dong et al. (2014) by combining the constituency tree and the dependency tree of a sentence. An important assumption of such work is that different tree structures lead to different semantic representations even for the same sentence. However, they all resort to ex-

* Corresponding author

Proceedings of NAACL-HLT 2018, pages 551–560
New Orleans, Louisiana, June 1 - 6, 2018.

ternal syntactic resources, such as parse trees or Treebank annotations (Marcus et al., 1993), which limits their broader applications. On the one hand, annotated data are highly expensive to produce; and informal texts, such those on the social media, remain a challenge for syntactic parsers (Kong et al., 2014). On the other hand, the tree structures in their pipeline-style architecture are fixed during training, which cascade errors to later representation learning stage.

A desirable solution would be to automatically and dynamically induce the tree structures for target-specific sentence representations. However, the challenge is that the absence of external supervisions makes it difficult to evaluate the quality of the tree structures and train the parameters. Inspired by Yogatama et al. (2016), we propose a reinforcement learning based approach that integrates target information and generates target-specific tree structures that benefit downstream classification tasks.

The underlying framework consists of two key components, a RNN encoder-decoder that explores possible binary tree structures according to a given target, and a tree-structured neural network that composes the input words into sentence representation based on the structure. The REINFORCE algorithm with the self-critic baseline (Rennie et al., 2016) is applied to update the parameters of the two components.

We evaluate our approach on two benchmark tasks: a firm-specific cumulative abnormal return prediction task (based on formal news texts) and an aspect-level sentiment analysis task (based on informal social media texts). Experimental results show that our approach achieves superior performances compared to baseline methods that operate on parsed trees. Moreover, our model sheds lights on understanding how sentences are composed from its word constituents towards specific targets.

2 Problem Definition

We formalize the problem of learning sentence representations for target-dependent classification tasks as constructing and semantically composing the target-specific binary syntactic trees of sentences. The input of the model is a tuple $(\mathbf{x}, x_{target}, c_{target})$, in which $\mathbf{x}$ is a sentence of n words $\{x_1, x_2, \cdots, x_n\}$; x_{target} is the target of interest mentioned in the sentence and c_{target} is the polarity regarding the target. For sentence $\mathbf{x}$, we can construct a valid binary syntactic tree by n SHIFT and $n-1$ REDUCE transitions $\mathbf{a} = \{a_0, a_1, \cdots, a_{2n-1}\}$, in which $a_t \in \{\text{SHIFT}, \text{REDUCE}\}$ specifies the transition taken at step t. The SHIFT transition adds a leaf node to the tree while the REDUCE transition combines two leaf nodes to form a parent node.

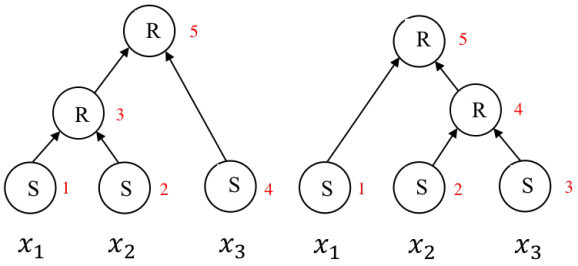


Figure 1: For input sequence $\{x_1, x_2, x_3\}$, the shift-reduce orders can be {S,S,R,S,R} and {S,S,S,R,R}, where **S** stands for SHIFT and **R** stands for REDUCE.

Figure 1 illustrates two examples on how can we construct a binary tree by only using SHIFT and REDUCE transitions and how can we obtain different binary trees by varying the SHIFT-REDUCE transition orders.

We design a transition generator $\mathbf{G}$ (Section 3.1) for generating transition orders $\mathbf{a}$, $\mathbf{G}(\mathbf{x}, x_{target}) \rightarrow \mathbf{a}$ and a composition function $\mathbf{C}$ (Section 3.2) that composes sentence $\mathbf{x}$ following the transition orders $\mathbf{a}$ into sentence representation $\mathbf{s}$, $\mathbf{C}(\mathbf{a}, \mathbf{x}) \rightarrow \mathbf{s}$.

Our ultimate goal is to use the sentence representation $\mathbf{s}$ for target-dependent classification. The objective is thus to minimize the negative log-likelihood Eq 1 with L2 norm, in which θ denotes all the parameters of our model.

$$J(\theta) = -\log P(c_{target}|\mathbf{s}; \theta) + \lambda||\theta||_2 \quad (1)$$

3 Model

The architecture of our proposed approach is illustrated in Figure 2, which is made up of two main components, a transition generator $\mathbf{G}$ and a composition function $\mathbf{C}$. The transition generator is a RNN encoder-decoder that generates discrete target-specific SHIFT-REDUCE transition orders, given a sentence and the target of interest. The composition function is a tree-structured neural network that semantically composes the word constituents following the transition orders. The main challenges for such a framework are two-fold. On

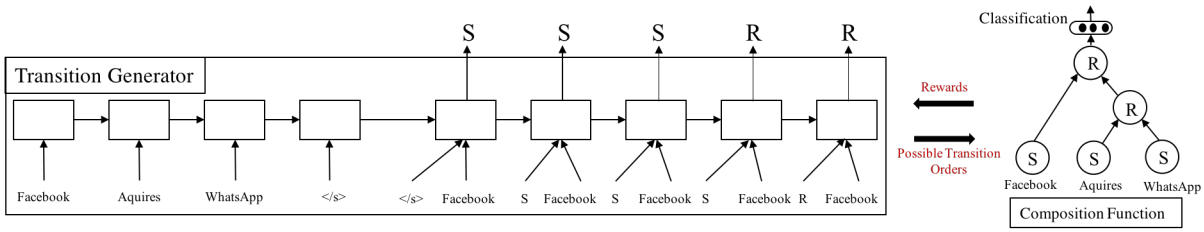


Figure 2: The framework of our proposed method. The left side is a variant of standard encoder-decoder that generates discrete SHIFT-REDUCE transition orders. It considers the target information at decoding. The right side is a composition function that semantically composes word representations into sentence representation following the transition orders. REINFORCE with the self-critical baseline is applied to reward the generated structures and update the parameters.

the one hand, the transition generator is fully unsupervised as we do not resort to external syntactic resources. On the other hand, the transitions generated at each step are discrete, making it difficult to train and propagate errors to update the model parameters. We give details of the two components and how we address the challenges in this section.

3.1 Transition Generator

The basic idea of the transition generator is to generate different transition orders given different targets. We propose using the RNN encoder-decoder framework (Cho et al., 2014), which has shown capacity in shift-reduce parsing (Vinyals et al., 2015; Liu and Zhang, 2017b). A standard RNN encoder-decoder contains two recurrent neural networks, one for encoding a sequence of variable-length into a vector representation and the other for decoding the representation back into another variable-length sequence.

Encoder We employ a standard Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) as our encoder. Given the input sentence $\{x_1, x_2, \cdots, x_n\}$, we first obtain their word vectors $\{\overrightarrow{e}(x_1), \overrightarrow{e}(x_2), \cdots, \overrightarrow{e}(x_n)\}$ by looking them up from a pre-trained embedding matrix $\overrightarrow{e}$. We reverse the input sentence and feed their word embeddings sequentially to the LSTM. The hidden states of each token $\{h_1, h_2, \cdots, h_n\}$ are kept for the decoding stage. The hidden state and cell state of the last LSTM unit are used as the initial states for decoder.

Decoder Following Bahdanau et al. (2014), we use an attention-based decoder. The decoder aligns with all the encoder hidden states at each step of decoding to obtain a context vector c_t, such that each input words show different weights at decoding. We denote the hidden states of our decoder as $\{d_1, d_2, \cdots, d_{2n-1}\}$. The attention score over each of the encoder hidden state h_i is computed by:

$$u_t^i = d_{t-1} \odot h_i \tag{2}$$

$$a_t^i = \frac{\exp(u_t^i)}{\sum_{i'} \exp(u_t^{i'})} \tag{3}$$

$$c_t = \sum_{i=1}^{n} a_t^i \cdot h_i, \tag{4}$$

in which $\odot$ denotes element-wise dot product; a_t^i is the normalized attention score and the context vector c_t is a weighted sum of all the encoder hidden states.

To enable the target of interest to influence the decoding process, we enrich the input of the decoder by concatenating the target entity. The hidden state of the decoder at time t is obtained by:

$$d_t = \text{LSTM}(c_t \oplus \overrightarrow{e}(a_{t-1}) \oplus \overrightarrow{e}(x_{target}), d_{t-1}), \tag{5}$$

in which $\oplus$ denotes concatenation operation; x_{target} is the embedding of the target entity; $\overrightarrow{e}(a_{t-1})$ is the embedding of the last decoded transition and c_t is the context vector.

Decoding In a supervised RNN decoder setting, the goal of each step is to estimate the conditional probability

$$P(a_t|a_{1:t-1}, c_t, d_t) = g(a_{t-1}, c_t, d_t), \tag{6}$$

in which $a_{1:t-1}$ are previously decoded transitions, c_t is the context vector, d_t is current decoder hidden state and g is non-linear network. $P(a_t|a_{1:t-1}, c_t, d_t\}$ is a distribution over the transition space $\{\text{SHIFT}, \text{REDUCE}\}$. By comparing

the decoded outputs with the ground-truth labels, the prediction errors can back-propagate to update parameters of the encoder-decoder network.

However, it is no more applicable in our settings, as we do not have any explicit supervisions from external syntactic resources. To make training the transition generator possible, we resort to a reinforcement learning framework, obtaining the transitions by sampling from a policy network. We represent the current state $\mathbb{S}_t$ by concatenating $\overrightarrow{e}(a_{t-1}), \overrightarrow{e}(x_{target}), c_t, d_t$.

$$\mathbb{S}_t = [\overrightarrow{e}(a_{t-1}) \oplus \overrightarrow{e}(x_{target}) \oplus c_t \oplus d_t] \quad (7)$$

The policy network $\pi(a_t|\mathbb{S}_t)$ is defined by Eq 8,

$$\pi(a_t|\mathbb{S}_t) \propto \exp(g(\mathbb{S}_t)), \quad (8)$$

in which g is a one-layer non-linear feed-forward neural network. We decode the transition a_t by sampling from the distributions given by the policy network.

3.2 Composition Function

When a valid binary tree of a sentence is generated, we use the composition function to obtain the representation following the transition orders. We maintain two data structures at composition; a buffer that stores words yet to be processed and a stack that stores the partially completed subtrees. Initially, the stack is empty, and the buffer stores all the words in the sentence. The operations specified by SHIFT and REDUCE are as follows.

- For a SHIFT transition, the buffer pops the topmost word out and pushes it to the top of the stack.
- For a REDUCE transition, the topmost two elements of the stack are popped out and composed. Their compositions are then pushed back to the stack.

To produce a valid binary tree, we follow Yogatama et al. (2016) to disallow SHIFT transition when the buffer is empty and forbid REDUCE transition when the stack has no more than two elements.

We use a tree-LSTM (Tai et al., 2015) to semantically compose the top two elements of the stack. Initially, the hidden state h_t and the cell state s_t of leaf nodes are given by another LSTM. The tree-LSTM works as follows,

$$\begin{bmatrix} i_t \\ f_t^l \\ f_t^r \\ o_t \\ g_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \cdot W \cdot \begin{bmatrix} h_t^l \\ h_t^r \end{bmatrix} \quad (9)$$

$$s_t = f_t^l \odot s_t^l + f_t^r \odot s_t^r + i_t \odot g_t$$
$$h_t = o_t \odot \tanh(\hat{s}_t),$$

in which $\odot$ denotes element-wise dot product; i_t and o_t are the input and output gate, respectively; f_t^l and f_t^r are the left and right forget gates; $h_t^l, h_t^r, s_t^l, s_t^r$ are the hidden and cell states of the left and right nodes in the subtree. The hidden state of the topmost node is used as the representation for the input sentence.

3.3 Training with REINFORCE

The goal for training is to optimize the parameters of the transition generator θ_G and the composition function θ_C. It is easy to optimize θ_C, the output of which is directly connected to the classifier, the classification loss can back-propagate to update its parameters.

However, the transitions sampled from the policy network $\pi(\mathbf{a}|\mathbb{S})$ are discrete, which makes θ_G no more differentiable to our objective. A possible solution is to maximize the expected reward $\mathbb{E}_{p(\mathbf{a};\theta_G)}R(\mathbf{a})$. As we are in a reinforcement learning setting, we can immediately receive a reward $\mathrm{R}(\mathbf{a})$ for transitions $\mathbf{a} = \{a_1, a_2, \cdots, a_t\}$ at the end of the classification. The reward is defined as the logarithm of classification probability for the right label c_{target}, $R(\mathbf{a}) = \log P(c_{target}|C(\mathbf{a}, \mathbf{x}))$.

However, it is computationally intractable to compute $\mathbb{E}_{p(\mathbf{a};\theta_G)}R(\mathbf{a})$, as the number of possible transition orders $\mathbf{a}$ is exponentially large. To address this, we use the REINFORCE algorithm to approximate the gradients by running M examples.

$$\bigtriangledown_{\theta_G} J(\theta_G) \approx -\frac{1}{M}\sum_{m=1}^{M}[\bigtriangledown_{\theta_G} \log p(\mathbf{a}) R^m(\mathbf{a})] \quad (10)$$

The $\bigtriangledown_{\theta_G} \log p(\mathbf{a})$ can be used to update θ_G.

REINFORCE algorithm is non-biased but may have high variance. To reduce the variance, a widely used trick is to subtract a baseline from the reward. It has been theoretically proven that

any baselines that do not depend on the actions are applicable. In this paper, we follow Rennie et al. (2016) to apply a self-critical baseline to the rewards. Rather than estimating a baseline reward, the self-critical method uses the outputs given by the test-time inference algorithm as the baselines. This can thus alleviate the over-fitting problem on test dataset.

At inference, we use a greedy decoding strategy by selecting the most probable transitions given by the policy network (Eq 8).

$$\hat{a}_i = \arg\max_{a_i} \pi(a_i|\mathbb{S}_t) \tag{11}$$

The self-critical baseline reward is $R(\hat{\mathbf{a}}) = \log P(c_{target}|C(\hat{\mathbf{a}}, \mathbf{x}))$, the formula to update θ_G become Eq 12.

$$\begin{aligned} &\nabla_{\theta_G} J(\theta_G) \approx \\ &-\frac{1}{M}\sum_{m=1}^{M}[\nabla_{\theta_G} \log p(\mathbf{a})(R^m(\mathbf{a}) - R^m(\hat{\mathbf{a}}))] \end{aligned} \tag{12}$$

4 Experiments and Results

The proposed approach is evaluated on two aspect-level tasks: (1) firm-oriented cumulative abnormal return prediction on formal financial news texts and (2) aspect-level sentiment analysis on informal social media texts.

4.1 Firm-specific cumulative abnormal return prediction

Firm-specific Cumulative Abnormal Return (CAR) prediction task (Chang et al., 2016) studies the impact of new information towards a specific firm. Multiple firms may be involved in the same new event, however, the event can present different impacts to these firms. Conceptually, **Abnormal Return** is the difference between the actual return of a stock and its expected return. The expected return can be approximated by daily indexes, such as S&P 500 index. For example, if a stock is expected to rise by 5%, but on the event day, it rises by 2%, although it gives a positive return, the abnormal return is -3%. **Cumulative Abnormal Return** is the accumulated abnormal return in an event window, which is usually triggered by new events. We use a three-day window (-1, 0, 1), denoted as CAR_3, with event day centering at day 0. We predict whether an event has positive or negative impact to the cumulative abnormal return of a given firm.

	Training	Development	Test
$+CAR_3$	7167	354	728
$-CAR_3$	7102	387	731
Total	14269	741	1459
Firms	1216	302	424

Table 2: Number of CAR_3 in the datasets

4.1.1 Data

We use the same news dataset as Chang et al. (2016), which are abstracts extracted from the Reuters news dataset released by Ding et al. (2014; 2015; 2016). Compared to the full texts of news documents, abstracts are supposed to be more informative and less noisy. Ding et al. (2014) show that modeling abstracts alone can achieve comparable or even better performances compared to full texts in stock market prediction. To better interpret our approach, we only extract event days with a single news document, which covers over 70% cases in the dataset. This final dataset yields a total of 16469 instances, including 1291 firms, of which 10% are reserved for validation, and 20% are used for testing. The numbers of positive and negative CAR_3 examples and number of firms in the subsets are listed in Table 2.

4.1.2 Baseline

To evaluate the performance of our approach on formal news texts, we compare with state-of-the-art target-independent and target-dependent baselines. Among the baselines, *Sentiment-based* and *Bi-LSTM* are target-independent, which learn generic representations for sentences, while *Bi-LSTM + Attention* and *TGT-CTX-TLSTM* are target-dependent.

Sentiment-Based Sentiments among breaking news, earning reports and online message boards, are found to be correlated with market volatility (Schumaker and Chen, 2009; Das and Chen, 2007). We adopt lexicon-based sentiment analysis as our baseline, using the sentiment lexicons released by Loughran and McDonald (2011). We follow the prior literature (Mayew and Venkatachalam, 2012) and use the count of positive words, negative words, the differences between positives and negatives, and their length-normalized values as our feature vectors.

Bi-LSTM We stack a forward and a backward LSTM to capture the contextual representations for the sentence. The last hidden states of both

Parameters	Value
word dimension	200
LSTM hidden dimension	200
dropout probablity	0.5
batch size	64
initial learning rate	0.0005

Table 3: Hyper-parameters for firm-oriented cumulative abnormal return task

directions are concatenated and then used for classification.

Bi-LSTM + Attention We extend vanilla Bi-LSTM by adding an attention mechanism over the hidden states. We concatenated the hidden states $\hat{h}_t = \{h_t^l, h_t^r\}$ of each input token x_t, the target representation $\overrightarrow{e}_{target}$ is adopted to weigh each of the hidden states.

$$u_t = v^\top \tanh(W_1 \overrightarrow{e}(x_{target}) + W_2 h_t + b) \quad (13)$$

$$a_t = softmax(u_t) \quad (14)$$

$$d_t = \sum a_t h_t \quad (15)$$

TGT-CTX-TLSTM The method of Chang et al. (2016), which we follow and is used as our main baseline. It is a hybrid model which integrates both sequential information and syntactic parse tree information. As the first step, the abstract is parsed with an external syntactic parser to obtain the dependency relations between the words. The parse tree are then adapted and binarized depending on their distances to targets in the dependency graph. A tree-structured Long Short-Term Memory Network (Tai et al., 2015) is then applied to learn a vector representation of the binarized tree structure.

4.1.3 Parameters & Metrics

The hyper-parameters used in this paper are listed in Table 3. We pretrain word vectors with the Word2Vec (Mikolov et al., 2013) tool on the news dataset released by Ding et al. (2014), which are fine-tuned during training. The embeddings of target firms are obtained by averaging their words of constituents.

We use macro-F1 to evaluate the performance on both positive and negative classes.

4.1.4 Test Results

The macro-F1 scores of our method and baselines are presented in Table 4. Sentiment-based method

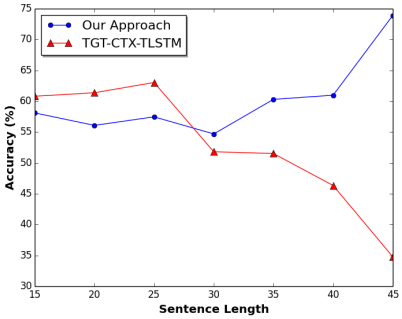


Figure 3: Accuracy with respect to sentence length.

gives the highest F1 score on the positive class. However, its performance is not consistent on the negative class, which suggests that it tends to misclassify the sentence as positive. Bi-LSTM + Attention outperforms the vanilla one without attention and is much robust in both positive and negative analysis. Our approach achieves an overall Macro-F1 of 58.2%, with an F1 score of 57.2% and 59.2% on positive and negative classes, respectively. Compared to the state-of-the-art model that exploits automatically parsed structures, we obtain an over 2% absolute gains without using explicit supervisions in learning the structures.

Method	**Class**	**F1-score**
Sentiment-based	+CAR$_3$	**0.597**
	-CAR$_3$	0.476
	Macro	0.536
Bi-LSTM	+CAR$_3$	0.557
	-CAR$_3$	0.490
	Macro	0.523
Bi-LSTM + Attention	+CAR$_3$	0.575
	-CAR$_3$	0.523
	Macro	0.549
TD-CTX-TLSTM	+CAR$_3$	0.552
	-CAR$_3$	0.570
	Macro	0.561
Our Approach	+CAR$_3$	0.572
	-CAR$_3$	**0.592**
	Macro	**0.582**

Table 4: Results for cumulative abnormal return prediction task

4.1.5 Accuracy Versus Sentence Length

Longer sentences are much more challenging for syntactic parsers. To gain insights on the performances of our approach on long sentences, we further inspect the accuracies with regards to different sentence lengths. As shown in Figure 3, we compare with structure-dependent baseline TGT-CTX-TLSTM. We divide the sentences into seven bins, each of which contains sentences with length $[5 *$

$i, 5 * (i + 1)]$. TGT-CTX-TLSTM gives higher accuracies over sentences with shorter lengths, while the accuracies decline sharply over sentences with lengths of over 30. Our approach is more consistent on both long and short sentences. As the sentence length grows, the accuracy our model gradually increases, showing its robustness and effectiveness across sentences of variable lengths.

4.2 Aspect-level Sentiment Analysis

To verify our proposed approach on informal social media texts, we apply it to aspect-level sentiment analysis on tweets. Aspect-level sentiment analysis aims to identify sentiment polarities towards specific targets mentioned in a sentence. Target-specific sentence representations can be naturally applied to this task.

Dataset	#Target	#Positive	#Negative	#Neutral
Training	6248	1568	1560	3127
Testing	692	173	173	346

Table 5: Statistics of aspect-level sentiment analysis datasets

4.2.1 Dataset

We apply our model to a benchmark aspect-level sentiment analysis dataset used in previous work (Dong et al., 2014). The statistics of the dataset are shown in Table 5. The target entities and corresponding ground-truth labels are annotated. The labels belong to one of {positive, neutral, negative}, thus the task is a three-way classification.

4.2.2 Baselines

We compare our approach with feature-based and neural-based models.

Jiang et al. (2011) They extract rich target-dependent and target-independent lexical and syntactic features for classification.

Dong et al. (2014) They adapt the parse tree of a sentence concerning the target with predefined rules and use recursive neural network (Socher et al., 2013) to learn a target-specific sentence representation.

4.2.3 Parameters & Metrics

The parameter settings are listed in Table 6. We use 100-dimension GloVe vectors which are pre-trained on a large Twitter Corpus (Pennington et al., 2014) and fine-tuned during training.

Parameters	Value
word dimension	100
LSTM hidden dimension	100
dropout probablity	0.5
batch size	32
initial learning rate	0.0005

Table 6: Hyper-parameters for aspect-level sentiment analysis

The commonly-used metrics classification accuracy and macro-F1 are adopted to evaluate the performances.

Model	Acc	F1
Jiang et al.(2011)	63.4	63.3
Dong et al.(2014)	66.3	65.9
Our Method	**68.2**	**66.3**

Table 7: Final results on aspect-level sentiment analysis task

4.2.4 Final Results

The final results on aspect-level sentiment analysis task are shown in Table 7. Dong et al. (2014) are used as our main baseline, as they build target-specific sentence representation over adapted tree structures. Neural-based models outperform Jiang et al. (2011), which did a lot of feature engineerings, showing the effectiveness of automatically induced features. Our approach gives superior performances compared to Dong et al. (2014), which operates on parsed trees. We achieve 68.2% classification accuracy and 66.3 macro-F1. We do not rely on a preprocessing syntactic parser as the first step to obtain the tree structures. On the one hand, social media texts are informal and extremely noisy, which remains a challenge for syntactic parsers. The pipeline-style architecture of Dong et al. (2014) cascades parse errors to later stages, which will hurt the performances on downstream tasks. On the other hand, the adapted tree structures in Dong et al. (2014), while in our approach, the tree structures are also tuned dynamically during training, so as to find the optimal structures that would benefit downstream classification tasks.

4.3 Case Study

To gain further insights on the induced structures, we inspect the shift-reduce trees our approach generated in this section. We present two examples that our model gives high confidences in Figure 4. For the sentence *"Nike NKE.N has sued Wal-Mart WMT.N, saying the world 's largest retailer*

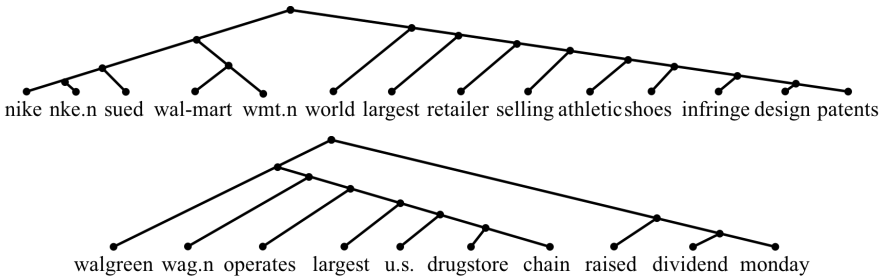


Figure 4: Two tree structures generated by our model. We removed stop words and punctuations. The upper tree structure is for the sentence *"Nike NKE.N has sued Wal-Mart WMT.N, saying the world 's largest retailer is selling athletic shoes that infringe on its design patents"* and the bottom one is for the sentence *"Walgreen WAG.N , which operates the largest U.S. drugstore chain , raised its dividend on Monday."*

is selling athletic shoes that infringe on its design patents", the core part *"Nike sued Wal-mart"* and the rest of the sentence are in two separate subtrees, which reduces potentially information loss about the key event when composing them into sentence representation. Similarly, for the sentence *"Walgreen WAG.N , which operates the largest U.S. drugstore chain , raised its dividend on Monday."*, the model learns to make the target *"Walgreen"* and key event *"raised its dividend on Monday"* close to each other in the tree, although there are sequentially many words in between. These are good examples given by our model, we also find a lot of highly left- or right-biased tree structures. Intuitively, the completely left- and right-biased tree structures are equivalent to forward and backward sequential structures, respectively.

5 Related Work

Our model is related to the following research areas, each having tremendous literatures.

5.1 Target-specific Sentence Representation

It is beneficial for numerous tasks, such as aspect-level sentiment analysis and stance detection, to have the sentence representations being tailored to specific targets. Early approaches rely on feature engineering by extracting target-dependent features (Jiang et al., 2011), while recent work mainly focuses on semantic compositions over the vector space with deep neural models. Depending on how they model the target and context, we further classify related work into three categories.

The first category relies on syntactic parse trees. Dong et al. (2014) are among the first to exploit tree structures, in which they adapt the parse trees based on the dependency relations between the words and the target, and then use a recursive neural network to learn the sentence representations. Similarly, Chang et al. (2016) explore a hybrid model that considers both sequential and structural information of a sentence. Nguyen et al. (2015) extend Dong et al. (2014) by combining the constituency tree and the dependency tree of a sentence. The performances of their methods highly rely on external parsers, which is subject to noise in informal social media texts.

The second category models the interactions between the target and its left context and right context. Vo and Zhang (2015) split a sentence into three parts and use pooling function to automatic inducing features for a given target. Similar to Vo and Zhang (2015), Zhang et al. (2016) exploit the gates instead of pooling functions to control the information flow of contexts. Tang et al. (2015) model by concatenating the word embeddings and target entity embeddings and use two LSTMs to encode left- and right contexts. Liu et al. (2017a) propose to use the attention mechanism to assign different weights to the left and right context depending on the target.

The third category controls the information flow from the target to the sentence representation. Augenstein et al. (2016) use conditional encoding to encode the target and use it as the initial states for the sentence representation.

Our method belongs to the first category that exploits tree structures. The main difference is we do not use external supervision from dependency parser or treebank annotations.

5.2 Neural-based Syntactic Constituency Parsing

Our work is related to syntactic constituency parsing as we build the tree structure in a transition

manner. Syntactic constituency parsing is a fundamental task in natural language processing, which uses phrase structure to organize words into nested constituents. Early approaches rely on probabilistic context-free grammars or transition-based models with rich features (Collins, 1997; Klein and Manning, 2003). Recently, recursive neural network (Socher et al., 2013) and neural-based transition model (Liu and Zhang, 2009) are also applied, which achieve competitive or even better performances compared to traditional state-of-the-art approaches that rely on hand-crafted features. Vinyals et al. (2015), from which we get inspirations, use the RNN Encoder-Decoder to encode the sentence and generate its corresponding full parse tree. Bowman et al. (2016) propose a Stack SPINN framework that integrates parsing and interpreting the sentence in a hybrid model. Yogatama et al. (2016) extend their model by using reinforcement learning to build the tree structures that can improve performances of end tasks.

We differ from the aforementioned approaches in two aspects. First, we do not use any explicit supervisions to guide the decoder. The parameters of our framework are optimized by the objective of end tasks. Another difference is that we learn target-specific instead of general-purpose sentence representations.

6 Conclusion

In this paper, we propose a framework that automatically induces target-specific sentence representations over tree structures without recourse to external syntactic resources. Experimental results on formal and informal texts showed that our approach is both robust and effective compared to previous work that operates on parsed trees. Moreover, the approach gives intuitions on how sentence structures are composed from their word constituents concerning a specific target.

Acknowledgements

We would like to thank the anonymous reviewers for their insightful comments and suggestions to help improve this paper. This work was partly supported by the National Key Basic Research Program of China via grant 2014CB340503, the National Natural Science Foundation of China (NSFC) via grant 61472107 and 61702137.

References

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. 2016. Stance Detection with Bidirectional Conditional Encoding pages 876–885. http://arxiv.org/abs/1606.05464.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. 2016. A Fast Unified Model for Parsing and Sentence Understanding https://doi.org/10.18653/v1/P16-1139.

Ching-Yun Chang, Yue Zhang, Zhiyang Teng Teng, Zahn Bozanic, and Bin Ke. 2016. Measuring the information content of financial news. In *26th Coling*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 16–23.

Sanjiv R Das and Mike Y Chen. 2007. Yahoo! for amazon: Sentiment extraction from small talk on the web. *Management Science* 53(9):1375–1388.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2014. Using structured events to predict stock price movement: An empirical investigation. In *EMNLP*. pages 1415–1425.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2015. Deep learning for event-driven stock prediction. In *IJCAI*. pages 2327–2333.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2016. Knowledge-driven event embedding for stock prediction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. pages 2133–2142.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *ACL (2)*. pages 49–54.

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

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pages 151–160.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*. Association for Computational Linguistics, pages 423–430.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A Smith. 2014. A dependency parser for tweets. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 1001–1012.

Jiwei Li, Minh-Thang Luong, Dan Jurafsky, and Eudard Hovy. 2015. When are tree structures necessary for deep learning of representations? *arXiv preprint arXiv:1503.00185* .

Jiangming Liu and Yue Zhang. 2009. Shift-Reduce Constituent Parsing with Neural Lookahead Features 1.

Jiangming Liu and Yue Zhang. 2017a. Attention Modeling for Targeted Sentiment. *Short Papers* 2:572–577. https://www.aclweb.org/anthology/E/E17/E17-2091.pdf.

Jiangming Liu and Yue Zhang. 2017b. Encoder-decoder shift-reduce syntactic parsing. *arXiv preprint arXiv:1706.07905* .

Tim Loughran and Bill McDonald. 2011. When is a liability not a liability? textual analysis, dictionaries, and 10-ks. *The Journal of Finance* 66(1):35–65.

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

William J Mayew and Mohan Venkatachalam. 2012. The power of voice: Managerial affective states and future firm performance. *The Journal of Finance* 67(1):1–43.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Thien Hai Nguyen. 2015. PhraseRNN : Phrase Recursive Neural Network for Aspect-based Sentiment Analysis (September):2509–2514.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2016. Self-critical sequence training for image captioning. *arXiv preprint arXiv:1612.00563* .

Robert P Schumaker and Hsinchun Chen. 2009. Textual analysis of stock market prediction using breaking financial news: The azfin text system. *TOIS* 27(2):12.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*. Citeseer, volume 1631, page 1642.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075* .

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2015. Effective lstms for target-dependent sentiment classification. *arXiv preprint arXiv:1512.01100* .

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*. pages 2773–2781.

Duy-tin Vo and Yue Zhang. 2015. Target-Dependent Twitter Sentiment Classification with Rich Automatic Features (Ijcai):1347–1353.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2016. Learning to Compose Words into Sentences with Reinforcement Learning pages 1–10. https://doi.org/10.1051/0004-6361/201527329.

M Zhang, Y Zhang, and DT Vo. 2016. Gated Neural Networks for Targeted Sentiment Analysis. *30th Conference on Artificial Intelligence (AAAI 2016)* pages 3087–3093. http://zhangmeishan.github.io/targeted-sentiment.pdf.

Relevant Emotion Ranking from Text Constrained with Emotion Relationships

Deyu Zhou[†] **Yang Yang**[†] **Yulan He**[§]
[†] School of Computer Science and Engineering, Key Laboratory of Computer Network and Information Integration, Ministry of Education, Southeast University, China
[§] School of Engineering and Applied Science, Aston University, UK
{d.zhou,yyang}@seu.edu.cn, y.he@cantab.net

Abstract

Text might contain or invoke multiple emotions with varying intensities. As such, emotion detection, to predict multiple emotions associated with a given text, can be cast into a multi-label classification problem. We would like to go one step further so that a ranked list of relevant emotions are generated where top ranked emotions are more intensely associated with text compared to lower ranked emotions, whereas the rankings of irrelevant emotions are not important. A novel framework of relevant emotion ranking is proposed to tackle the problem. In the framework, the objective loss function is designed elaborately so that both emotion prediction and rankings of only relevant emotions can be achieved. Moreover, we observe that some emotions co-occur more often while other emotions rarely coexist. Such information is incorporated into the framework as constraints to improve the accuracy of emotion detection. Experimental results on two real-world corpora show that the proposed framework can effectively deal with emotion detection and performs remarkably better than the state-of-the-art emotion detection approaches and multi-label learning methods.

1 Introduction

With the growing prosperity of Web 2.0, people tend to share their feelings, attitudes and opinions through the social platforms such as online news sites, blogs. Detecting emotions from text can enhance the understanding of users' emotional states, which is useful in many downstream applications, such as human-computer interaction and personalized recommendation. Therefore, it is crucial to analyze and predict emotions from text accurately (Picard and Picard, 1997).

Research on emotion detection can be roughly categorized into two types: lexicon-based and learning-based approaches. Lexicon-based approaches usually rely on emotion lexicons (Lei et al., 2014; Rao et al., 2012). They cannot deal with text when words can't be found in emotion lexicons. Learning-based approaches can be furthered classified into unsupervised and supervised learning methods. Unsupervised approaches do not require annotated data for training. For example, by adding an emotion layer into traditional topic models, emotion-topic models were constructed to detect users' emotions (Bao et al., 2012, 2009). Supervised learning approaches consider each emotion category as a class label and emotion detection is cast as a classification problem. If only choosing the strongest emotion as the emotion label for a given text, emotion detection is essentially a single-label classification problem (Lin et al., 2008; Quan et al., 2015). To predict multiple emotions simultaneously, emotion detection can be solved in the multi-label classification framework (Bhowmick, 2009). Moreover, to predict both multiple emotions and their intensities, some approaches have been proposed using emotion distribution learning (Zhou et al., 2016). Some lexicon-based approaches such as (Wang and Pal, 2015) can also output multiple emotions with intensities using non-negative matrix factorization.

In this paper, we are interested in exploring emotion ranking from either readers' perspective or writers' perspective in two different real-world corpora. In both cases, a given text is associated with multiple emotions. For example, Figure 1 illustrates an online news article crawled from Sina News *Society* Channel together with readers' emotion votes. It can be observed that when reading the news article, readers expressed different emotions with the majority showed "*Sadness*" and "*Anger*". We notice that some emotions such as "*Touching*", "*Curiosity*" and "*Amusement*" on-

Proceedings of NAACL-HLT 2018, pages 561–571
New Orleans, Louisiana, June 1 - 6, 2018.

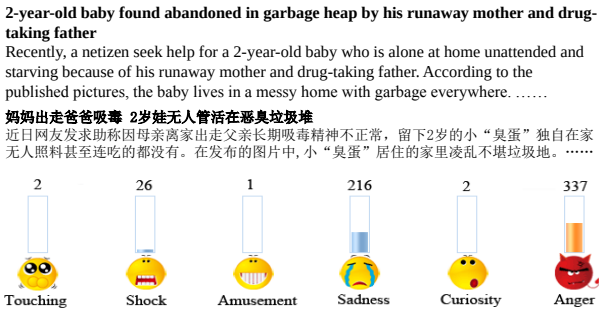


Figure 1: An example of an online news article from Sina *Society* Channel with voted emotions.

ly received 1 to 3 votes. In comparison to the total number of votes received, these votes could be considered as outliers or irrelevances. Also, the extremely low emotion votes might be due to readers' clicking errors. Taking into account such emotions during the learning process could introduce bias. Therefore, we aim to differentiate relevant emotions from irrelevant ones and only learn the rankings of relevant emotions while neglecting the irrelevant ones.

Our work makes the following contributions:

- We propose a novel framework based on relevant emotion ranking to identify multiple emotions and produce the rankings of relevant emotions from text. In the framework, the objective emotion loss function is designed elaborately so that both emotion prediction and rankings for only relevant emotions are achieved without being affected by irrelevant ones. To the best of our knowledge, it is the first attempt to perform emotion detection and relevant emotion ranking at the same time.

- As some emotions co-occur more often while others rarely co-exist, the prior knowledge of emotion relationships is incorporated into the framework as a constraint. Such emotion relationship can provide important cues for emotion detection.

- Experimental results on two real-world corpora show that the proposed framework can effectively deal with the emotion detection problem and performs better than the state-of-the-art emotion detection methods and multi-label learning methods.

2 Related work

Emotion detection is one of the subfields of sentiment analysis where emotions are more fine-grained and expressive. In general, emotion detection approaches can be categorized into two types: lexicon-based and learning-based approaches.

Lexicon-based approaches usually rely on emotion lexicons consisting of words and their corresponding emotion labels. For example, Aman and Szpakowicz (2007) classified emotional and non-emotional sentences with a predefined emotion lexicon. Emotional dictionaries could also be constructed from training corpora of news articles and be used to predict the readers' emotion of a new articles (Lei et al., 2014; Rao et al., 2012). Agrawal and An (2012) proposed a context-based approach to detect emotions from text at sentence level. An emotion vector for each potential affect bearing word based on the semantic relation between emotion concepts and words was generated. The emotion vector was then tuned based on the syntactic dependencies within a sentence structure. Other lexicon-based approach such as (Wang and Pal, 2015) can also output multiple emotions with intensities using non-negative matrix factorization with constraints derived based on an emotion lexicon.

Learning-based approaches can be further categoried as unsupervised and supervised learning methods. Unsupervised learning approaches do not require labelled data for training. For example, the emotion-topic models (Bao et al., 2012, 2009) were proposed by adding an extra emotion layer into traditional topic models such as Latent Dirichlet Allocation (Blei et al., 2003), thus capturing the generation of both emotion and text at the same time.

Supervised learning approaches typically cast emotion detection as a classification problem by considering each emotion category as a class label. If only choosing the strongest emotion as the label for a given text, emotion detection is essentially a single-label classification problem. Lin, Yang and Chen (2008) studied the classification of news articles into different categories based on readers' emotions with various combinations of feature sets. Strapparava and Mihalcea (2008) proposed several knowledge-based and corpus-based methods for emotion classification. Quan et al. (2015) proposed a logistic regression model with emotion dependency for emotion detection. Latent vari-

ables were introduced to model the latent structure of input text. To predict multiple emotions simultaneously, emotion detection can be solved using multi-label classification. Bhowmick (2009) presented a method for classifying news sentences into multiple emotion categories using an ensemble based multi-label classification technique. Zhou et al. (2016) proposed a novel approach based on emotion distribution learning to predict multiple emotions with different intensities in a single sentence.

3 Methodology

Assuming a set of T emotions $E = \{e_1, e_2, ...e_T\}$ and a set of n instances $X = \{x_1, x_2, x_3, ..., x_n\}$, each instance $x_i \in \mathbb{R}^d$ is associated with a ranked list of its relevant emotions $R_i \subseteq E$ and also a list of irrelevant emotions $\overline{R_i} = E - R_i$. Relevant emotion ranking aims to learn a score function $\mathbf{g}(x_i) = [g_1(x_i), ..., g_T(x_i)]$ assigning a score $g_t(x_i)$ to each emotion $e_t, (t \in \{1, ..., T\})$. As mentioned before, it is unnecessary to consider the rankings of irrelevant emotions since they might introduce errors into the model during the learning process. In order to differentiate relevant emotions from irrelevant ones, we need to define a threshold $g_\Theta(x)$ which could be simply set to 0 or learned from data (Fürnkranz et al., 2008). Those emotions with scores lower than the threshold will be considered as irrelevant and hence discarded. The identification of relevant emotions and their ranking can be obtained simultaneously according to their scores assigned by the ranking function $\mathbf{g}$. Here, the predicted relevant emotions of instance x_i are denoted as $\hat{R}_i = \{e_t \in E | g_t(x_i) > g_\Theta(x_i)\}$.

3.1 Emotion Loss Function

The goal of relevant emotion ranking is to learn the parameter of the ranking function $\mathbf{g}$. Without loss of generality, we assume that $\mathbf{g}$ are linear models, i.e., $g_t(x_i) = w_t^\intercal \cdot x_i$, $t \in \{1, 2, 3, ..., T\} \cup \{\Theta\}$, where Θ denotes the threshold. Relevant emotion ranking can be regarded as a special case of multi-label learning. Several evaluation criteria typically used in multi-label learning can also be used to measure the ranking function's ability of distinguishing relevant emotions from irrelevant ones, such as hamming loss, one error, coverage, ranking loss, and average precision as suggested in (Zhang and Zhou, 2014). However, these multi-label criteria cannot meet our requirement exactly as none of them considers the ranking among emotions which are considered relevant. Therefore, by incorporating PRO loss (Xu et al., 2013), the loss function for the instance x_i is defined as follows:

$$L(x_i, R_i, \prec, \mathbf{g}) = \sum_{e_t \in R_i \cup \{\Theta\}} \sum_{e_s \in \prec(e_t)} \frac{1}{norm_{t,s}} l_{t,s} \tag{1}$$

where e_t refers to the emotion belonging to relevant emotion set R_i or the threshold Θ of instance x_i while e_s refers to the emotion which is less relevant than e_t denoted as $\prec$. Thus, (e_t, e_s) represents four types of emotion pairs: i.e., *(relevant, relevant)*, *(relevant, irrelevant)*, *(relevant, threshold)*, and *(threshold, irrelevant)*. The normalization term $norm_{t,s}$ is used to balance those four types of emotion pairs to avoid dominated terms by their respective set sizes. The set sizes of the four different types of emotion pairs mentioned above are $|R_i| \times (|R_i| - 1)/2$, $|R_i| \times |\overline{R_i}|$, $|R_i|$, and $|\overline{R_i}|$, respectively. Here, $l_{t,s}$ refers to a modified 0-1 error. Specifically,

$$l_{t,s} = \begin{cases} 1, & g_t(x_i) < g_s(x_i) \\ \frac{1}{2}, & g_t(x_i) = g_s(x_i) \\ 0, & \text{otherwise} \end{cases}$$

Note that $l_{t,s}$ is non-convex and difficult to optimize. Thus, a large margin surrogate convex loss (Vapnik and Vapnik, 1998) implemented in hinge form is used instead as follows:

$$\hat{L}(x_i, R_i, \prec, \mathbf{g}) = \sum_{e_t \in R_i \cup \{\Theta\}} \sum_{e_s \in \prec(e_t)} \frac{1}{norm_{t,s}} (1 + g_s(x_i) - g_t(x_i))_+ \tag{2}$$

where $(u)_+ = \max\{0, u\}$.

However, Eq. 2 ignores the relationships between different emotions. As mentioned in Introduction section, some emotions often co-occur such as "*joy*" and "*love*" while some rarely coexist such as "*joy*" and "*anger*". Such relationship information among emotions can provide important clues for emotion ranking. Therefore, we incorporate this information into the emotion loss function as constraints. The objective function

$\hat{L}(x_i, R_i, \prec, \mathbf{g})$ can be redefined as:

$$\hat{L_\omega}(x_i, R_i, \prec, \mathbf{g}) = \sum_{e_t \in R_i \cup \{\Theta\}} \sum_{e_s \in \prec(e_t)} \frac{1}{norm_{t,s}} \times (1 + g_s(x_i) - g_t(x_i) + \omega_{ts}(w_t - w_s))_+ \quad (3)$$

where the weight ω_{ts} models the relationship between the *t*-th emotion and the *s*-th emotion in the emotion set and can be calculated in multiple ways. Since the Pearson correlation coefficient (Nicewander, 1988) is the most familiar measure of relationship between two variables, we use it to measure the relationship of two emotions using their original emotion scores across each corpus.

From the above, it can be observed that the goal of relevant emotion ranking can be achieved through predicting an accurate relevant emotion set as well as the ranking of relevant emotions.

3.2 Relevant Emotion Ranking

After defining an appropriate loss function, we need to define a way to minimize the empirical error measured by the appropriate loss function and at the same time to control the complexity of the resulting model. It can be done by introducing a maximum margin strategy and regularization to deal with emotion ranking data, where a set of linear classifiers are optimized to minimize the emotion loss function mentioned before while having a large margin. We could potentially use an approach based on a label ranking method (Elisseeff and Weston, 2001). It is worth mentioning that the margin of the *(relevant, relevant)* label pair needs to be dealt with carefully, which is not considered in (Elisseeff and Weston, 2001).

The learning procedure of relevant emotion ranking (RER) is illustrated in Figure 2. The big rectangular dash line boxes denoted by x_1 to x_n represent n instances in the training set. In each small box, $e_i, i \in \{1, ...T\} \cup \{\Theta\}$ represents an emotion of the instance where the shaded small boxes represent the relevant emotions while the dashed small boxes represent irrelevant ones and the last one e_Θ is the threshold. Each emotion's corresponding weight vector is w_i. We use $m_{t,s}$ to represents the margin between label e_t and e_s. There are four types of emotion pairs' margins in total, i.e., *(relevant, relevant)*, *(relevant, irrelevant)*, *(relevant, threshold)*, and *(threshold, irrelevant)*. Different types of emotion pairs' margins are denoted using different text/line colors. For each training instance x_i, $margin(x_i)$ represents the margin of instance x_i which can be obtained by taking the minimum margin of all its possible label pairs $m_{t,s}$. Similarly, the margin of the learning system $margin(learningsystem)$ can be obtained by taking the minimum margin of all the training instances. By maximizing the margin of the learning system, the weight vector of each emotion can be derived from which the predicted emotion set and the ranking of relevant emotions can be obtained.

The learning system is composed of $T + 1$ linear classifiers $[w_1; ...; w_T; w_\Theta]$ with one classifier for each emotion label and the threshold, where $w_t, t \in \{1, ...T\} \cup \{\Theta\}$ is the weight vector for the t-th classifier of emotion e_t. For a training instance x_i and its corresponding emotion label set E_i, the learning system's margin on instance x_i is defined as follows by considering its ranking ability on x_i's four types of emotion pairs, i.e., *(relevant, relevant)*, *(relevant, irrelevant)*, *(relevant, threshold)*, and *(threshold, irrelevant)*:

$$\min_{e_t \in R_i \cup \{\Theta\}, e_s \in \prec(e_t)} \frac{\langle w_t - w_s, x_i \rangle}{||w_t - w_s||} \quad (4)$$

Here, $\langle u, v \rangle$ returns the inner product $u^\top v$. For each emotion pair (e_t, e_s), its discrimination boundary corresponds to the hyperplane $\langle w_t - w_s, x_i \rangle = 0$. Therefore, Eq. 4 returns the minimum value as the margin on instance x_i. The margin on the whole training set G can be calculated as follows:

$$\min_{x_i \in G} \min_{e_t \in R_i \cup \{\Theta\}, e_s \in \prec(e_t)} \frac{\langle w_t - w_s, x_i \rangle}{||w_t - w_s||} \quad (5)$$

If the learning algorithm is capable of properly ranking the four types of label pairs for each training instance, Eq. 5 will return a positive margin. In this ideal case, the final goal is to maximize the margin in Eq. 5:

$$\max_{w_j} \min_{x_i \in G} \min_{e_t \in R_i \cup \{\Theta\}, e_s \in \prec(e_t)} \frac{1}{||w_t - w_s||} \quad s.t. \langle w_t - w_s, x_i \rangle \geq 1, 1 \leq i \leq n, 1 \leq j \leq T + 1 \quad (6)$$

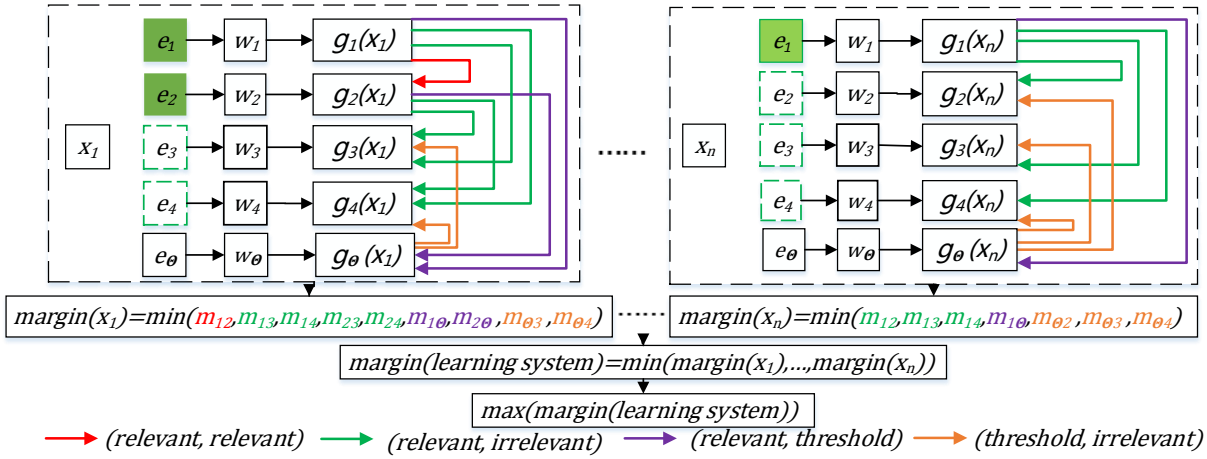


Figure 2: The overall framework of our proposed Relevant Emotion Ranking (RER) method.

Suppose we have sufficient training examples such that for each label pair (e_t, e_s), there exists $x_i \in G$satisfying $e_t \in R_i \cup \{\Theta\}, e_s \in \prec (e_t)$. Thus, the objective in Eq.6 becomes equivalent to $\max_{w_j} \min_{1 \le s < t \le T+1} \frac{1}{||w_t - w_s||}$ and can be rewritten as $\min_{w_j} \max_{1 \le s < t \le T+1} ||w_t - w_s||$.

Moreover, to overcome the complexity brought in by the max operator, the objective of the optimization problem can be re-written by approximating the max operator with the sum operator. Thus, the objective of Eq. 6 can be transformed as:

$$\begin{aligned} &\min_{w_j} \sum_{t=1}^{T+1} ||w_t||^2 \\ &s.t.\ \langle w_t - w_s, x_i \rangle \ge 1, 1 \le i \le n, \\ &1 \le j \le T+1, e_t \in R_i \cup \{\Theta\}, e_s \in \prec (e_t) \end{aligned} \tag{7}$$

To accommodate real-world scenarios where constraints in Eq. 7 can not be fully satisfied, slack variables can be incorporated into the objective function:

$$\begin{aligned} &\min_{w_j, \xi_{its}} \sum_{t=1}^{T+1} ||w_t||^2 + \lambda \sum_{i=1}^{n} \sum_{e_t \in R_i \cup \{\Theta\}} \sum_{e_s \in \prec (e_t)} \frac{1}{norm_{t,s}} \xi_{its} \\ &s.t. \langle w_t - w_s, x_i \rangle \ge 1 - \xi_{its}, 1 \le j \le T+1, \xi_{its} \ge 0 \end{aligned} \tag{8}$$

Since ξ_{its} does not need to be optimized since it can be easily determined by w_t, w_s. The final objective function can be reformulated as:

$$\min_{w_t, \hat{L}} \sum_{t=1}^{T+1} ||w_t||^2 + \lambda \sum_{i=1}^{n} \hat{L}(x_i, R_i, \prec, \mathbf{g}) \tag{9}$$

As can be seen, Eq.9 consists of two parts balanced by the trade-off parameter λ. Specifically, the first part corresponds to the maximum margin of the learning system and it can also represent the complexity of the learning system, while the second part corresponds to the emotion loss function of the learning system implemented in hinge form.

3.3 Parameter Estimation

Let $\mathbf{w} = [w_1; ...; w_T; w_\Theta]$, Eq. 9 is cast into a general form in SVM-type:

$$\begin{aligned} &\min_{\mathbf{w}, \boldsymbol{\xi}} \frac{1}{2} ||\mathbf{w}||^2 + \lambda C^\top \boldsymbol{\xi} \\ &s.t.\ A\mathbf{w} \ge \mathbf{1}_p - \boldsymbol{\xi}, \boldsymbol{\xi} \ge \mathbf{0}_p \end{aligned} \tag{10}$$

where p is the total number of label pairs, calculated by $\sum_{i=1}^{n} \sum_{e_t \in R_i \cup \{\Theta\}} \sum_{e_s \in \prec (e_t)} norm_{t,s}$ and $\mathbf{1}_p (\mathbf{0}_p)$ is the $p \times 1$ all one (zero) vector. The entries in vector C correspond to the weights of hinge losses, i.e., the normalization term to balance the four kinds of label pairs. The matrix A corresponds to the constraints for instances which reflects the emotion relationships and the margin of the label pairs.

$\boldsymbol{\xi}$ does not need to be optimized since it can be easily determined by $\mathbf{w}$. Hence the objective function can be reformulated into the following form without $\boldsymbol{\xi}$:

$$\min_{\mathbf{w}} F(\mathbf{w}, G) = \frac{1}{2} ||\mathbf{w}||^2 + \lambda C^\top (1_p - A\mathbf{w})_+ \tag{11}$$

Through minimizing the objective function $F(\mathbf{w}, G)$, we can finally obtain parameter $\mathbf{w}$ and the ranking function $\mathbf{g}$. Eq. 11 involves a large scale optimization. To address Eq. 11, we consider an efficient Alternating Direction Method of Multipliers (ADMM) solution (Bertsekas and Tsitsiklis, 1989). The basic idea of ADMM is to take the decomposition-coordinate procedure such that the solution of subproblems can be coordinated to find the solution to the original problem. We decompose G into M disjoint subsets, i.e., $\{G_1, G_2, ..., G_M\}$ and then Eq. 11 is converted into the following form:

$$\begin{aligned} &\min_{\mathbf{w}^0, \mathbf{w}^1, \mathbf{w}^m} \sum_{m=1}^{M} F(\mathbf{w}^m, G^m), \\ &s.t.\ \ \mathbf{w}^m = \mathbf{w}^0, \forall m = 1, ..., M \end{aligned} \tag{12}$$

The surrogate augmented Lagrangian Function (LF) was introduced into Eq. 12 and it was cast into the following form:

$$\begin{aligned} LF(\{\mathbf{w}^0, \mathbf{w}^1, ..., \mathbf{w}^m\}, \{\alpha^m\}_{m=1}^M, \beta) = \sum_{m=1}^{M} F(\mathbf{w}^m, G^m) \\ + \sum_{m=1}^{M} (\alpha^m)^\top (\mathbf{w}^m - \mathbf{w}^0) + \frac{\beta}{2} \sum_{m=1}^{M} ||\mathbf{w}^m - \mathbf{w}^0||^2 \end{aligned} \tag{13}$$

where α, β are the Lagrange multiplies. The updating process of Eq. 13 is shown in Algorithm 1.

Algorithm 1 Parameter updating process.

1: Decompose data set G into M disjoint subsets i.e., $\{G_1, G_2, ..., G_M\}$. Set iteration $i = 0$.
2: Initialize $\{\mathbf{w}_0^0, \mathbf{w}_0^1, ..., \mathbf{w}_0^M, \alpha_0^1, ..., \alpha_0^M\}$ as zeros.
3: **while** not converged **do**
4: Set $i = i + 1$
5: Update $\mathbf{w}_i^0, \{\mathbf{w}_i^m, \alpha_i^m\}_{m=1}^M$ as:
$\{\mathbf{w}_i^m\}_{m=1}^M = \underset{\mathbf{w}^1..\mathbf{w}^m}{\operatorname{argmin}} LF(\mathbf{w}_{i-1}^0, \{\mathbf{w}_{i-1}^m, \alpha_{i-1}^m\}_{m=1}^M, \beta)$
$\mathbf{w}_i^0 = \underset{\mathbf{w}^0}{\operatorname{argmin}} LF(\mathbf{w}^0, \{\mathbf{w}_{i-1}^m, \alpha_{i-1}^m\}_{m=1}^M, \beta$
$\alpha_i^m = \alpha_{i-1}^m + \beta(\mathbf{w}_i^m - \mathbf{w}_i^0)^\top, \forall m = 1, 2, ..., M$
6: **end while**
Output: Final $\mathbf{w}^0$

4 Experiments

4.1 Setup

We evaluate the proposed approach on two real-world corpora, one is document level and the other is sentence level:

Sina Social News (News) was collected from the Sina news *Society* channel where readers can choose one of the six emotions such as *Amusement*, *Touching*, *Anger*, *Sadness*, *Curiosity*, and *Shock* after reading a news article. As Sina is one of the largest online news sites in China, it is sensible to carry out experiments to explore the readers' emotion (social emotion). News articles with less than 20 votes were discarded since few votes can not be considered as proper representation of social emotion. In total, 5,586 news articles published from January 2014 to July 2016 were kept, together with the readers' emotion votes.

Ren-CECps corpus (Blogs) (Quan and Ren, 2010) contains 34,719 sentences selected from blogs in Chinese. Each sentence is annotated with eight basic emotions from writer's perspective, including *anger*, *anxiety*, *expect*, *hate*, *joy*, *love*, *sorrow* and *surprise*, together with their emotion scores indicating the level of emotion intensity which range from 0 to 1. Higher scores represents higher emotion intensity.

The statistics of the two corpora are shown in Table 1.

Sina Social News		**Ren-CECps Corpus**	
Category	#Votes	Category	#Scores
Touching	694,006	Joy	1,349.6
Shock	572,651	Hate	6,103.9
Amusement	869,464	Love	2,911.1
Sadness	837,431	Sorrow	2,042.5
Curiosity	212,559	Surprise	3,873.9
Anger	1,109,315	Anger	7,832.1
		Anxiety	5,006.4
		Expect	610.4
All	4,295,426	All	29,729.9

Table 1: Statistics for the two corpora used in our experiments.

The two corpora were preprocessed by using word segmentation and filtering. The python jieba segmenter is used for the segmentation and a removal of stop words is performed based on a stop word thesaurus. Words appeared only once or appeared in less than two documents were re-

moved to alleviate data sparsity. We used the single layer long short-term memory (LSTM) networks (Hochreiter and Schmidhuber, 1997) to extract the features of each text. LSTM is one kind of recurrent neural networks, which can capture sequence information from text and can represent meanings of inputs in the reduced dimensional space. It treats text as a sequence of word embeddings and outputs a state vector over each word, which contains the information of the previous words. The final state vector can be used as the representation of the text. In our experiments, we set the dimension of each text representation to 100. During LSTM model training, we optimized the hyper parameters using a development dataset which is built using external data. We train LSTM using a learning rate of 0.001, a dropout rate of 0.3 and categorical cross-entropy as the loss function. The mini batch (Cotter et al., 2011) size is set to 32. After that, the learned text representations are fed into the proposed system for relevant emotion ranking as has been previously presented in the Methodology section.

4.2 Comparison with Baselines

There are only few baselines which address multiple emotions learning from text. We first compare the proposed framework with two baselines which have previously achieved the state-of-the-art performances on multi-emotion detection.

- **Emotion Distribution Learning (EDL)** (Zhou et al., 2016): It learns a mapping function from texts to their emotion distributions describing multiple emotions and their respective intensities based on label distribution learning. Moreover, the relationships of emotions are captured based on the Plutchik's wheel of emotions which are subsequently incorporated into the learning algorithm in order to improve the accuracy of emotion detection.

- **EmoDetect** (Wang and Pal, 2015): It outputs the emotion distribution based on a dimensionality reduction method using non-negative matrix factorization which combines several constraints such as emotions bindings, topic correlations and emotion lexicons in a constraint optimization framework.

For each method, 10-fold cross validation is conducted using the same feature construction

Name	Definition
PRO Loss	$\frac{1}{n}\sum_{i=1}^{n}\sum_{e_t\in R_i\cup\{\Theta\}}\sum_{e_s\in\prec(e_t)}\frac{1}{norm_{t,s}}l_{t,s}$ $l_{t,s}$ is a modified 0-1 error;$norm_{t,s}$is the set size of label pair(t,s)
Hamming Loss	$\frac{1}{nT}\sum_{i=1}^{n}\lvert\hat{R}_i\triangle R_i\rvert$
Ranking Loss	$\frac{1}{n}\sum_{i=1}^{n}(\sum_{(e_t,e_s)\in R_i\times\overline{R_i}}\delta[g_t(x_i)<g_s(x_i)])/(\lvert R_i\rvert\times\lvert\overline{R_i}\rvert)$ where δ is the indicator function.
One Error	$\frac{1}{n}\sum_{i=1}^{n}\delta[\operatorname{argmax}_{e_t} g_t(x_i)\notin R_i]$
Average Precision	$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\lvert R_i\rvert}\times$ $(\sum_{t:e_t\in R_i}\lvert\{e_s\in R_i\lvert g_s(x_i)>g_t(x_i)\}\rvert)/(\lvert\{e_s\lvert g_s(x_i)>g_t(x_i)\}\rvert)$
Coverage	$\frac{1}{n}\sum_{i=1}^{n}\max_{t:e_t\in R_i}\lvert\{e_s\lvert g_s(x_i)>g_t(x_i)\}\rvert$
Subset Accuracy	$\frac{1}{n}\sum_{i=1}^{n}\delta[\hat{R}_i=R_i]$
$F1_{exam}$	$\frac{1}{n}\sum_{i=1}^{n}2\lvert R_i\cap\hat{R}_i\rvert/(\lvert R_i\rvert+\lvert\hat{R}_i\rvert)$
MicroF1	$F1(\sum_{t=1}^{T}TP_t,\sum_{t=1}^{T}FP_t,\sum_{t=1}^{T}TN_t,\sum_{t=1}^{T}FN_t)$
MacroF1	$\frac{1}{T}\sum_{t=1}^{T}F1(TP_t,FP_t,TN_t,FN_t)$

Table 2: Evaluation criteria for the Multi-Label Learning (MLL) methods. TP_t, FP_t, TN_t, FN_t represent the number of true positive, false positive, true negative, and false negative test examples with respect to emotion t respectively. $F1(TP_t, FP_t, TN_t, FN_t)$ represent specific binary classification metric F1 (Manning et al., 2008).

method to get the final performance. Supposing n test instances and T emotion categories, several evaluation criteria as presented in Table 2 typically used in multi-label learning can be used to measure the efficiency of the proposed framework and the baseline approaches. PRO Loss concerning the prediction on all labels as well as the rankings of only relevant labels. Hamming loss evaluates how many times an emotion label is misclassified. Ranking loss evaluates the fraction of reversely ordered emotion pairs. One-error evaluates the fraction of sentences whose top-ranked emotion is not in the relevant emotion set. Average precision evaluates the average fraction of the relevant emotions ranked higher than a particular emotion. Coverage evaluates how many steps are needed to move down the ranked emotion list so as to cover all the relevant emotions of the example. Subset Accuracy evaluates the fraction of correctly classified examples, i.e. the predicted label set is identical to the ground-truth label set. $F1_{exam}$ evaluates the averaged F1 (Manning et al., 2008) over instances. MicroF1 pools each document decisions across categories, and then computes an effectiveness measure on the pooled contingency table. MacroF1 take the average of F1 for all categories. Note that the threshold Θ is removed before evaluation. It should be pointed out that metrics from PRO Loss to $F1_{exam}$ work by evaluating the learning systems performance on each test ex-

ample separately, and then returning the mean value across the test set. MicroF1 and MacroF1 work by evaluating the learning systems performance on each emotion category separately, and then returning the macro/micro-averaged value across all emotion categories.

The evaluation results using 10 different evaluation criteria are shown in Table 3. It can be observed that our proposed method Relevant Emotion Ranking(RER) outperforms baseline approaches on all 10 evaluation metrics on both datasets.

Datasets	Evaluation Criterion	Methods			
		RER	RERc	EDL	EmoDetect
News	PRO loss(↓)	0.1992	**0.1913**	0.2596	0.2465
	Hamming Loss(↓)	0.2318	**0.2277**	0.2671	0.2696
	Ranking Loss(↓)	0.1477	**0.1405**	0.1689	0.1769
	One-error(↓)	0.1579	**0.1562**	0.2115	0.1903
	Average Precision(↑)	0.8775	**0.8789**	0.8028	0.7865
	Coverage(↓)	2.1398	**2.1316**	2.1595	2.2348
	Subset Accuracy(↓)	0.1899	**0.1822**	0.2026	0.2243
	$F1_{exam}$(↑)	0.7062	**0.7143**	0.6503	0.6469
	MicroF1(↑)	0.7086	**0.7171**	0.6346	0.6375
	MacroF1(↑)	0.6244	**0.6291**	0.5641	0.5767
Blogs	PRO loss(↓)	0.2354	**0.2321**	0.2739	0.2912
	Hamming Loss(↓)	0.2054	**0.2014**	0.2102	0.2202
	Ranking Loss(↓)	0.2137	**0.2102**	0.2589	0.2781
	One-error(↓)	0.4556	**0.4550**	0.5227	0.5352
	Average Precision(↑)	0.6749	**0.6803**	0.6411	0.5663
	Coverage(↓)	2.1269	**2.1268**	2.1699	2.8956
	Subset Accuracy(↓)	**0.1663**	**0.1663**	0.2116	0.2321
	$F1_{exam}$(↑)	0.5080	**0.5114**	0.4606	0.4650
	MicroF1(↑)	0.5093	**0.5116**	0.4620	0.4552
	MacroF1(↑)	0.4102	**0.4161**	0.3923	0.3622

Table 3: Comparison with emotion detection baselines. "↓" indicates "the smaller the better", while "↑" indicates "the larger the better". The best performance on each evaluation measure is highlighted by boldface.

We have also extended RER by incorporating emotion relationships as constraints into the learning framework, denoted as RERc in Table 3. The correlation of every pair of emotions is calculated based on their respective votes from news articles or scores from blogs. It can be observed from Table 3 that in almost all cases, incorporating the constraints gives better performance. It should be pointed out that the results of the baseline approach EDL are slightly different from those reported in (Zhou et al., 2016) since we employ LSTM for feature construction instead of recursive autoencoders.

Since relevant emotion ranking can be seen as an extension of multi-label learning, the proposed framework is also compared with 8 widely used multi-label learning methods using the threshold Θ which is initialized as 0.15 after normalization, such as ML-KNN (Zhang and Zhou, 2007), LIFT (Zhang, 2011), CLR (Fürnkranz et al., 2008), Rank-SVM (Zhang and Zhou, 2014) , MLLOC (Huang and Zhou, 2012), BP-MLL (Zhang and Zhou, 2006), ECC (Read et al., 2009) and ML-RBF (Zhang, 2009). ML-KNN is based on the traditional k-nearest neighbor (KNN) algorithm. LIFT constructs features specific to each emotion by conducting clustering analysis on its positive or negative instances. CLR transforms MLL into a label ranking problem by pairwise comparison which considers each label pairs and rank all the labels without recognizing that only the rankings of relevant ones are crucial. Rank-SVM focuses on distinguishing relevant from irrelevant while neglecting the rankings of relevant ones. MLLOC tries to exploit emotion correlations in the expression data locally. The global discrimination fitting and local correlation sensitivity are incorporated into a unified framework. BP-MLL is derived from the back propagation algorithm through employing a novel error function capturing the characteristics of multi-label learning. ECC applies classifier chains in an ensemble framework. ML-RBF gets the multi-label neural networks adopted from the traditional Radial Basis Function (RBF) neural networks.

Anger	Anxiety	Expect	Hate
生气(angry)	害怕(fear)	祝福(blessing)	讨厌(hate)
愤怒(rage)	失去(lose)	幸福(happy)	虚伪(hypocrisy)
抱怨(complain)	孤独(lonely)	美好(fine)	炒作(hype)
批评(criticize)	压力(pressure)	梦想(dream)	无耻(shameless)
利益(interest)	现实(reality)	自由(freedom)	手段(means)
歧视(discriminate)	陌生(strange)	渴望(long for)	愚蠢(silly)
制止(stop)	心灵(heart)	希望(hope)	浪费(waste)
指责(accuse)	痛苦(pain)	学习(learn)	背后(behind)
懊恼(annoy)	想象(imagine)	信念(faith)	肮脏(dirty)
无耻(shameless)	伤害(hurt)	家里(home)	欺骗(lie)
Joy	**Love**	**Sorrow**	**Surprised**
快乐(happy)	美丽(beautiful)	孤独(lonely)	好奇(curious)
高兴(joyful)	爱情(love)	眼泪(tears)	惊讶(surprise)
朋友(friend)	朋友(friend)	爱情(love)	震惊(shock)
感动(touching)	幸福(happiness)	寂寞(solitude)	惊奇(wonder)
心情(mood)	孩子(child)	痛苦(pain)	惊人(amazing)
温暖(warm)	生命(life)	感情(feeling)	意外(accident)
享受(enjoy)	阳光(sunshine)	伤害(hurt)	惊吓(fright)
兴奋(excited)	温暖(warmth)	失去(lose)	惊呼(scream)
收获(harvest)	思念(miss)	思念(miss)	不经意(accidently)
微笑(smile)	可爱(lovely)	生活(life)	诧异(amazed)

Figure 3: The top 10 words for each emotion label from Blogs dataset.

For the MLL methods, the value of k is set to 8 in ML-KNN, ratio is 0.02 and μ is 2 in ML-RBF. Linear kernel is used in LIFT. Rank-SVM uses the RBF kernel with the width σ equals to 1. The CR4.5 is used as the base classifier for CLR and ECC. The evaluation results of the proposed

Datasets	Evaluation Criterion	Methods								
		RERc	ML-KNN	LIFT	CLR	Rank-SVM	MLLOC	BP-MLL	ECC	ML-RBF
News	PRO loss(↓)	**0.1913**	0.2551	0.2426	0.2487	0.2670	0.3429	0.2603	0.2823	0.2658
	Hamming Loss(↓)	**0.2277**	0.2876	0.3118	0.3023	0.3127	0.3241	0.3040	0.3079	0.3599
	Ranking Loss(↓)	**0.1405**	0.1898	0.1987	0.2142	0.2271	0.3234	0.1897	0.2563	0.1949
	One-error(↓)	**0.1562**	0.2366	0.1881	0.2242	0.2258	0.2025	0.2043	0.2151	0.2240
	Average Precision(↑)	**0.8789**	0.8095	0.7945	0.7916	0.8001	0.7545	0.8044	0.6245	0.8106
	Coverage(↓)	**2.1316**	2.3602	2.4641	2.3453	2.6093	3.1272	2.4032	2.4122	2.4390
	Subset Accuracy(↓)	**0.1822**	0.1916	0.1857	0.2386	0.1839	0.2107	0.2765	0.2222	0.2609
	$F1_{exam}$(↑)	**0.7143**	0.6215	0.6262	0.6032	0.6244	0.5193	0.5879	0.5108	0.6147
	MicroF1(↑)	**0.7171**	0.6280	0.6131	0.6177	0.6268	0.5389	0.6231	0.5699	0.6160
	MacroF1(↑)	**0.6291**	0.5587	0.5593	0.5658	0.5613	0.4913	0.5563	0.4573	0.5543
Blogs	PRO loss(↓)	**0.2321**	0.3036	0.2912	0.3041	0.2869	0.3523	0.3429	0.2867	0.2922
	Hamming Loss(↓)	**0.2014**	0.2409	0.2242	0.2162	0.2585	0.2156	0.2241	0.2301	0.2204
	Ranking Loss(↓)	**0.2102**	0.2928	0.2881	0.2947	0.3024	0.4532	0.3234	0.3345	0.2364
	One-error(↓)	**0.4550**	0.5543	0.5152	0.5229	0.5606	0.6143	0.4625	0.6635	0.4679
	Average Precision(↑)	**0.6803**	0.5897	0.5963	0.6370	0.5832	0.4532	0.5545	0.5256	0.6412
	Coverage(↓)	**2.1268**	2.4448	2.4356	2.2671	2.5962	3.5634	3.1272	2.7756	2.5067
	Subset Accuracy(↓)	**0.1663**	0.1978	0.2116	0.1938	0.2321	0.2251	0.2107	0.2236	0.1803
	$F1_{exam}$(↑)	**0.5114**	0.4616	0.4620	0.4509	0.4832	0.4931	0.5093	0.4986	0.4997
	MicroF1(↑)	**0.5116**	0.4720	0.4552	0.4859	0.4962	0.4902	0.4889	0.5003	0.5051
	MacroF1(↑)	**0.4161**	0.3632	0.3656	0.4056	0.3965	0.3853	0.3813	0.3957	0.4086

Table 4: Comparison with Multi-Label Learning (MLL) Methods."↓" indicates "the smaller the better", while "↑" indicates "the larger the better". The best performance on each evaluation measure is highlighted by boldface.

approach in comparison to all MLL baselines are presented in Table 4. RERc performs the best on all evaluation measures. It verifies the advantage of RERc due to its consideration of varying intensities of the emotion labels and the ignorance of irrelevant ones during the learning of the relevant emotion ranking model. We also observe that, in most cases, the performance on the News dataset is better than that in the Blogs dataset. This may due to different types of text observed in these two platforms. News articles are more formal while bogs typically contain informal language and are more colloquial.

4.3 Result Analysis

To fully understand the emotion detection results, we generate the top 10 most frequent words in the test set for each emotion label from Blogs dataset presented in Figure 3. Intuitively, we can find that most top words are quite expressive of their associated emotions. For example, the word "happy" delivers the emotion of *Joy* and the word "tears" tells *Sorrow*, etc. Moreover, we also observe that there are some common words across multiple emotion categories. For instance, "friend" appears in both *Joy* and *Love*. The results demonstrate that the proposed framework can learn emotions from text precisely.

5 Conclusions

In this paper, we have proposed a novel framework to detect multiple emotions from text based on relevant emotion ranking. Moreover, the relationships between emotions are incorporated into the learning framework as constraints. Experimental results on both News and Blogs datasets show that the proposed framework is able to detect multiple emotions as well as generating rankings of relevant emotions. It performs remarkably better than the state-of-the-art baselines on multi-emotion detection and also outperforms several different methods used for multi-label learning. In the future, we will explore the possibility of extending the current framework by detecting emotions at more fine-grained level, for example, emotions associated with specific events reported in text.

Acknowledgments

The work was supported by the National Key R&D Program of China (No. 2017YFB1002801), the National Natural Science Foundation of China (61772132), the Natural Science Foundation of Jiangsu Province of China (BK20161430) and Innovate UK (103652).

References

Ameeta Agrawal and Aijun An. 2012. Unsupervised emotion detection from text using semantic and syntactic relations. In *Ieee/wic/acm International Conferences on Web Intelligence and Intelligent Agent Technology*. pages 346–353.

Saima Aman and Stan Szpakowicz. 2007. Identifying expressions of emotion in text. In *Text, Speech and Dialogue, International Conference, Tsd 2007, Pilsen, Czech Republic, September 3-7, 2007, Proceedings*. pages 196–205.

Shenghua Bao, Shengliang Xu, Li Zhang, Rong Yan, Zhong Su, Dingyi Han, and Yong Yu. 2009. Joint emotion-topic modeling for social affective text mining. In *2009 Ninth IEEE International Conference on Data Mining*. IEEE, pages 699–704.

Shenghua Bao, Shengliang Xu, Li Zhang, Rong Yan, Zhong Su, Dingyi Han, and Yong Yu. 2012. Mining social emotions from affective text. *IEEE transactions on knowledge and data engineering* 24(9):1658–1670.

Dimitri P Bertsekas and John N Tsitsiklis. 1989. *Parallel and distributed computation: numerical methods. Reprint of the 1989 edition published by Prentice-Hall.*. Prentice Hall.

Plaban Kumar Bhowmick. 2009. Reader perspective emotion analysis in text through ensemble based multi-label classification framework. *Computer and Information Science* 2(4):64.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3(Jan):993–1022.

Andrew Cotter, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. 2011. Better mini-batch algorithms via accelerated gradient methods. *Advances in Neural Information Processing Systems* pages 1647–1655.

Andr Elisseeff and Jason Weston. 2001. A kernel method for multi-labelled classification. In *International Conference on Neural Information Processing Systems: Natural and Synthetic*. pages 681–687.

Johannes Fürnkranz, Eyke Hüllermeier, Eneldo Loza Mencía, and Klaus Brinker. 2008. Multilabel classification via calibrated label ranking. *Machine learning* 73(2):133–153.

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

Sheng Jun Huang and Zhi Hua Zhou. 2012. Multi-label learning by exploiting label correlations locally. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*. pages 949–955.

Jingsheng Lei, Yanghui Rao, Qing Li, Xiaojun Quan, and Liu Wenyin. 2014. Towards building a social emotion detection system for online news. *Future Generation Computer Systems* 37:438–448.

Kevin Hsin-Yih Lin, Changhua Yang, and Hsin-Hsi Chen. 2008. Emotion classification of online news articles from the reader's perspective. In *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology-Volume 01*. IEEE Computer Society, pages 220–226.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. 2008. An introduction to information retrieval. *Journal of the American Society for Information Science and Technology* 43(3):824–825.

W. Alan Nicewander. 1988. Thirteen ways to look at the correlation coefficient. *American Statistician* 42(1):59–66.

Rosalind W Picard and Roalind Picard. 1997. *Affective computing*, volume 252. MIT press Cambridge.

Changqin Quan and Fuji Ren. 2010. Sentence emotion analysis and recognition based on emotion words using ren-cecps. *International Journal of Advanced Intelligence Paradigms* 2(1):105–117.

Xiaojun Quan, Qifan Wang, Ying Zhang, Luo Si, and Liu Wenyin. 2015. Latent discriminative models for social emotion detection with emotional dependency. *ACM Transactions on Information Systems (TOIS)* 34(1):2.

Yanghui Rao, Xiaojun Quan, Liu Wenyin, Qing Li, and Mingliang Chen. 2012. Building word-emotion mapping dictionary for online news. In *SDAD 2012 The 1st International Workshop on Sentiment Discovery from Affective Data*. page 28.

Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. 2009. Classifier chains for multi-label classification. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. pages 254–269.

Carlo Strapparava and Rada Mihalcea. 2008. Learning to identify emotions in text. In *Proceedings of the 2008 ACM symposium on Applied computing*. ACM, pages 1556–1560.

Vladimir Naumovich Vapnik and Vlamimir Vapnik. 1998. *Statistical learning theory*, volume 1. Wiley New York.

Yichen Wang and Aditya Pal. 2015. Detecting emotions in social media: A constrained optimization approach. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)*. pages 996–1002.

Miao Xu, Yu Feng Li, and Zhi Hua Zhou. 2013. Multi-label learning with pro loss. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*.

Min Ling Zhang. 2009. M l-rbf : Rbf neural networks for multi-label learning. *Neural Processing Letters* 29(2):61–74.

Min Ling Zhang. 2011. Lift: multi-label learning with label-specific features. In *International Joint Conference on Artificial Intelligence*. pages 1609–1614.

Min Ling Zhang and Zhi Hua Zhou. 2006. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge Data Engineering* 18(10):1338–1351.

Min Ling Zhang and Zhi Hua Zhou. 2007. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition* 40(7):2038–2048.

Min-Ling Zhang and Zhi-Hua Zhou. 2014. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering* 26(8):1819–1837.

Deyu Zhou, Xuan Zhang, Yin Zhou, Quan Zhao, and Xin Geng. 2016. Emotion distribution learning from texts. In *Conference on Empirical Methods in Natural Language Processing*. pages 638–647.

Solving Data Sparsity for Aspect based Sentiment Analysis using Cross-linguality and Multi-linguality

Md Shad Akhtar*, Palaash Sawant+, Sukanta Sen*, Asif Ekbal* and Pushpak Bhattacharyya*

* *Department of Computer Science & Engineering*
Indian Institute of Technology Patna, India
{shad.pcs15,sukanta.pcs15,asif,pb}@iitp.ac.in
+ *Goa University*, palaash77@gmail.com

Abstract

Efficient word representations play an important role in solving various problems related to Natural Language Processing (NLP), data mining, text mining etc. The issue of data sparsity poses a great challenge in creating efficient word representation model for solving the underlying problem. The problem is more intensified in resource-poor scenario due to the absence of sufficient amount of corpus. In this work, we propose to minimize the effect of data sparsity by leveraging bilingual word embeddings learned through a parallel corpus. We train and evaluate Long Short Term Memory (LSTM) based architecture for aspect level sentiment classification. The neural network architecture is further assisted by the hand-crafted features for the prediction. We show the efficacy of the proposed model against state-of-the-art methods in two experimental setups i.e. multi-lingual and cross-lingual.

1 Introduction

Sentiment analysis (Pang and Lee, 2005) tries to automatically extract the subjective information from a user written textual content and classifies it into one of the predefined set of classes, e.g. *positive*, *negative*, *neutral* or *conflict*. Sentiment analysis performed on coarser level (i.e. document or sentence level) does not provide enough information for a user who is critical of finer details such as *battery life* of a laptop or *service* of a restaurant etc. Aspect level sentiment analysis (ABSA) (Pontiki et al., 2014) serves such a purpose, which first identifies the features (or aspects) mentioned in the text and then classifies it into one of the target classes. For example, the following review is for a restaurant where the writer shares her/his experience. Though s/he likes the *food* but certainly not happy with the *service*.

*One of the best **food** we had in a while but the **service** was very disappointing.*

Analyzing such reviews on sentence level will reflect only an overall sentiment (i.e. *conflict*) of the sentence ignoring critical information such as *food* and *service* qualities. However, ABSA will first identify all the aspects in the text (i.e. *food* and *service*) and then associate *positive* with *food* and *negative* with *service*. Identification of aspect terms is also known as aspect term extraction or opinion target extraction. In this work, we focus on the second problem i.e. aspect level sentiment classification.

Literature survey suggests a wide range of research on sentiment analysis (at the document or sentence level) is being carried out in recent years (Turney, 2002; Kim and Hovy, 2004; Jagtap and Pawar, 2013; Poria et al., 2016; Kaljahi and Foster, 2016; Gupta et al., 2015). However, most of these researches are focused on resource-rich language like English. Like many other Natural Language Processing (NLP) problems, research on sentiment analysis involving Indian languages (e.g. Hindi, Bengali etc.) are very limited (Joshi et al., 2010; Bakliwal et al., 2012; Kumar et al., 2015; Balamurali et al., 2012; Singhal and Bhattacharyya, 2016). Due to the scarcity of various qualitative resources and/or tools in such languages, the problems have become more challenging and non-trivial to solve. The research on ABSA involving Indian languages has started only very recently, for e.g. (Akhtar et al., 2016a,b).

2 Motivation and Problem Definition

Indian languages are resource-constrained in nature as there is a lack of ready availability of different qualitative lexical resources and tools. In a supervised machine learning framework, good amount of training data always have a great impact on the overall system performance. Low-resource languages (such as the Indian [Hindi etc.]) usu-

Proceedings of NAACL-HLT 2018, pages 572–582
New Orleans, Louisiana, June 1 - 6, 2018.

ally suffer due to the non-availability of sufficient training data instances. In order to solve the data and resource scarcity problem in one language, researchers often utilize cross-lingual setup to leverage the resource-richness of other languages by projecting the task into a common problem space (Zhou et al., 2016; Balamurali et al., 2012; Singhal and Bhattacharyya, 2016; Barnes et al., 2016). The projection is often performed with the help of machine translation or bilingual dictionaries.

In recent times, deep learning (DL) techniques have shown success in solving several NLP problems. A good word representation is the essence of any deep learning approach. In the absence of qualitative word embeddings, it turns out to be a non-trivial task for any DL framework to effectively learn hidden features (e.g. lexical, syntactic, semantics etc.), which may effect the performance. The quality of word embeddings can be preserved by employing state-of-the-art distributed word representation models such as Word2Vec (Mikolov et al., 2013) or GloVe (Pennington et al., 2014) provided a huge corpus to train on. Due to this limitation, quality of word embeddings in Indian languages usually are not at par with that of resource-rich languages like English.

Data sparsity in word representation (i.e. absence of representation of any word) is another problem that often has to be dealt with. In order to solve any NLP task, out-of-vocabulary (OOV) words in a word embedding model pose a serious challenge to the underlying learning algorithm. For a missing word representation, the literature suggests two possible solutions: a) zero vector (Bahdanau et al., 2017) or b) random vector (Dhingra et al., 2017). However, in both the cases the resultant vector could be completely out of context and often does not fit well with others. Further, word embedding of a word in a source language has absolutely no correlation with the word embedding of the same word (translated) in the target language, hence, it cannot be directly used for training and/or testing in a cross-lingual setup. The prime motivation of the work is to minimize the effect of *data sparsity* and thereby, enabling any deep learning framework to effectively learn its hidden features.

In this paper, we propose to solve the *data sparsity* problem in a resource-scarce language scenario (here, primarily Hindi and also French embeddings)[1]. We hypothesize that addressing data sparsity in an intelligent manner would yield increased performance. We utilize bi-lingual word embedding (Luong et al., 2015) trained on English-Hindi and English-French parallel corpus to bridge the language divergence in the vector space. The proposed method is based on a deep learning (DL) architecture named Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997). We try to establish our hypothesis through experiments on *aspect based sentiment classification* task in both the setups i.e. multi-lingual and cross-lingual for *English-Hindi* and *English-French* language pairs. Aspect based sentiment classification deals with assigning the sentiment polarity (i.e. *positive*, *negative*, *neutral* or *conflict*) to the aspect terms. For evaluation, we use the datasets provided in (Akhtar et al., 2016a) for Hindi, SemEval-2014 shared task on ABSA (Pontiki et al., 2014) for English and SemEval-2016 shared task on ABSA (Pontiki et al., 2016) dataset for French.

2.1 Contributions

Major contributions of our current work are as follows: **a)** we train and use bilingual embeddings on Amazon product review corpus consisting of parallel sentences of English-Hindi and English-French, which serve as a bridge between the two languages; **b)** we propose to solve the problem of *data sparsity* in low-resource language word embedding by utilizing the word embedding created on resource-rich language; and **c)** to further improve the system's prediction we extract and use various English side semantic features of the machine translated words.

As we already mentioned, the research on ABSA involving Indian languages are limited. Some of the recent works include the one reported in (Akhtar et al., 2016a,b). The authors in (Barnes et al., 2016) employed bilingual word embeddings for sentiment classification in a cross-lingual setup. To the best of our knowledge, our current attempt is the very first of its kind to employ bilingual word embeddings for a multi-lingual scenario. Our proposed system differs with the existing systems in the following ways.

1. **Setup**: System (Barnes et al., 2016) defines a cross-lingual setup while (Singhal and Bhat-

[1] We use French to show how generic our proposed approach is. Compared to English, French does not have enough sentiment annotated data

tacharyya, 2016) is multi-lingual in nature. In contrast, our proposed system is applied to both multi-lingual and cross-lingual setups.

2. **Approach**: System (Akhtar et al., 2016a) defines classical feature driven approach while the system (Barnes et al., 2016) utilized bilingual word embeddings as feature values to train a Support Vector Machine (SVM) classifier. Rest of the systems (Akhtar et al., 2016b; Singhal and Bhattacharyya, 2016) (including the proposed one) are based on deep neural network architecture. However, the techniques employed are very much different. Akhtar et al. (2016b) is a CNN-SVM based system with the assistance of multi-objective optimized features, while Singhal and Bhattacharyya (2016) is a CNN based system that translate the source language texts into target language text (English) for training and evaluation. In comparison, our proposed method employ LSTM to solve the data sparsity problem in both multi-lingual as well as cross-lingual setups.

3. **Problem addressed**: Authors in (Singhal and Bhattacharyya, 2016) focused on sentence level sentiment classification while our present work focuses on fine-grained sentiment classification at the aspect level.

4. **Word Embeddings:** The proposed system employs shared vector-space bilingual word embeddings for training and testing while (Singhal and Bhattacharyya, 2016) projected the source language train & test data into target language using machine translation and utilizes target side pre-computed word vectors for training the system. Whereas, the system reported in (Akhtar et al., 2016b) employed mono-lingual word embeddings for training and evaluation.

5. **Data Sparsity:** The system of (Akhtar et al., 2016b) does not address the problem of data sparsity, while our proposed system tries to minimize the effect of data sparsity. Our proposed system tackles the data sparsity problem by replacing the OOV word with its translated form which usually happens to be its closest neighbor in the shared vector space, hence, the semantic closeness is preserved to an extent. Whereas, system (Singhal and Bhattacharyya, 2016) addressed the data sparsity by translating every word of the source language into target language which may introduce loss of sentiment in the target language as a side-effect (Mohammad et al., 2016).

6. **Hand-crafted Features:** The proposed system employs much richer set of lexicon based features than that of (Singhal and Bhattacharyya, 2016). Also, we do not augment polar words in the training instances as done in (Singhal and Bhattacharyya, 2016), rather we use sentiment scores of these lexicons as features themselves in the training and testing instances. Whereas, the authors in (Akhtar et al., 2016b) obtained an optimized feature vector through the application of multi-objective genetic algorithm.

3 Proposed Methodology

We propose to use a Long Short Term Memory (LSTM) architecture on top of bilingual word embeddings for the prediction. LSTM is a special kind of recurrent neural network (RNN) which efficiently captures long term dependencies. Bidirectional LSTM is an extended version of LSTM which takes both forward and backward sequences into account. Our model consists of two bidirectional LSTM layers followed by two fully-connected layers and an output layer.

3.1 Bilingual Word Embedding

We employ bilingual word embeddings (Luong et al., 2015) trained on a parallel *English-Hindi* (and *English-French*) corpus. We generate a parallel corpus for Amazon product review datasets[2] (consisting of approx. 7.2M sentences) using an in-house product review domain based *English→Hindi* (*English→French*) Statistical Machine Translation (SMT) system (*English→Hindi*: 39.5 BLEU score and *English→French*: 37.9 BLEU score). We employ widely used and standard machine translation tool *Moses* (Koehn et al., 2007) to train the phrase-based SMT system. The alignment information are obtained from the *mosesdecoder* (Koehn et al., 2007) during translation of the reviews.

The parallel corpus along with the alignment information are used to train two (English and Hindi)

[2]http://snap.stanford.edu/data/other.html

Skip-Gram word2vec (Mikolov et al., 2013) models which share the common vector space. If a word W_S is aligned to word W_T, then the context information C_T of target word W_T is also used as context of the source word W_S along with its own context C_S for computing word vectors. By utilizing the context information of both source and target side, resultant word embeddings of W_S and W_T are semantically closer to each other in the vector space.

Bilingual skip-gram model creates two separate word embeddings, i.e. one each for source (Hindi) and target language (English). First, we extract word representations for all the words in a sentence from the Hindi bilingual word embeddings. Subsequently at the second step we translate all the OOV words (words whose representations are missing in Hindi bilingual embeddings) into English and then perform another lookup in English embeddings. For instance, if embedding of a word 'अच्छाlachcha' is unknown we translate it in English as '*good*', and use its word embeddings in place of the source word 'अच्छाlachcha'. Thus the missing representation of OOV word is replaced by its translated target side representation. Since, both English and Hindi word embeddings share a common vector space, this replacement strategy proves to be an effective technique. In our case, we observe a reduction of approximately 65% and 37% OOV words, respectively for Hindi and French by our proposed replacement strategy. Consequently, an increase in accuracy value is observed during evaluation.

Hindi is a morphologically rich language. Many inflected words in Hindi share a common translated word in English. For example, based on the gender of the subject Hindi has two forms for word 'goes': 'जाता है | jAtA hai' (male) or 'जाती है | jAtI hai' (female). Therefore, if representation of one word (जाता है | jAtA hai) is missing in Hindi embedding we can still find its representation in English through its translation i.e. 'goes'. Bilingual embedding also helps in addressing the spelling variation cases. For e.g. two differently spelled words in Hindi such as 'कम्बिनेशन | kambineshana' and 'कंबीनेशन | kaMbIneshana' translate to an English word 'combination'.

We repeat the above process for *English-French* language pair to obtain two (English and French) word2vec models. We also released computed bilingual word embeddings for the research community[3].

3.2 Features

We employ various hand-crafted features to assist the network. We try to leverage the effectiveness of English side resources by translating a word into English and then extracting its feature representation. We use following set of features in our task. It should be noted that we do not include any lexical or syntactic features during training as distributed word embedding models are good at capturing such features. So, during the training phase, network adapts its weights to learn the relevant set of these features from the word embeddings itself.

1. **Bing Liu** (Ding et al., 2008) **& MPQA** (Wiebe and Mihalcea, 2006) **lexicons**: We define a feature that marks the positivity/negativity scores of the words in a sentence. We assign a score of +1 & -1, respectively to each positive and negative word in the sentence. For unseen words, we use score as 0. We extract one such feature from each lexicon.

2. **SentiWordNet** (Baccianella et al., 2010): Three features are extracted for every word denoting its positivity (posScore), negativity (negScore) and objectivity (1 - [posScore + negScore]) scores, respectively.

3. **Semantic Orientation (SO)** (Hatzivassiloglou and McKeown, 1997): Semantic orientation defines the association of a word *w.r.t.* its positivity and negativity. Semantic orientation (SO) of a word is the difference of point-wise mutual information of a word w in positive and negative reviews. We calculate the SO score of each word in the context window of size ± 5 and take the cumulative SO score as the feature value.

3.3 Cross-lingual and Multi-lingual Setups

We evaluate our proposed approach for two setups i.e. multi-lingual and cross-lingual setups. In multi-lingual setup, the proposed model is trained and evaluated on datasets of the same language i.e. Hindi or French. We pre-process our datasets to reduce the effect of data sparsity by utilizing the resource-rich language i.e. English. In contrast, the

[3]Bi-lingual word embeddings available at http://www.iitp.ac.in/~ai-nlp-ml/resources.html

cross-lingual setup employs dataset of resource-rich language (i.e. English) for training and during evaluation Hindi or French dataset is used. Similar to the multi-lingual setup, we pre-process the test dataset to reduce the effect of data sparsity in cross-lingual setup as well.

An overall schema of the proposed methodology is depicted in Figure 1 for both multi-lingual and cross-lingual setups. Figures 1a and 1b show the training architectures for the cross-lingual and multi-lingual scenarios, respectively. Since our test datasets for both the variants are in Hindi (or French), testing scenario for cross-lingual and multi-lingual setups are also the same as represented in Figure 1c.

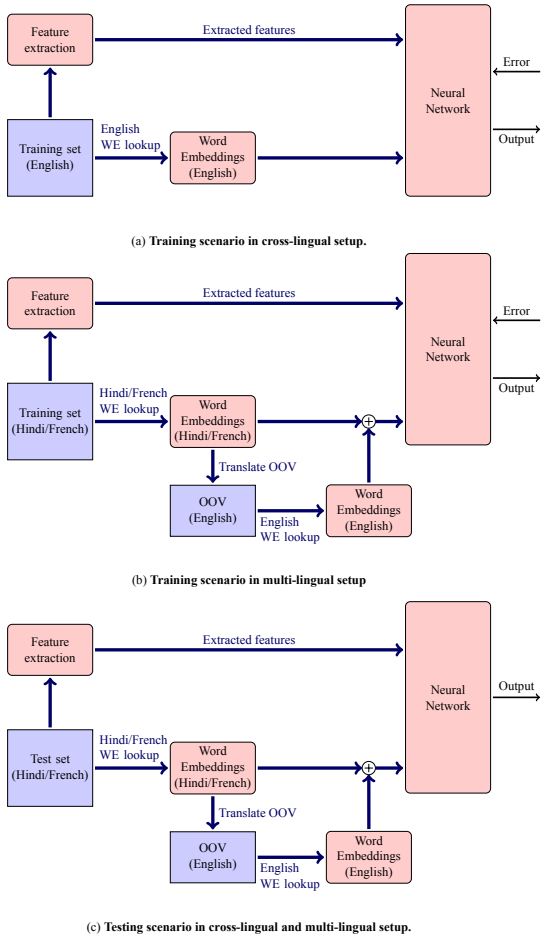


Figure 1: Proposed schema for *English-Hindi* and *English-French* language pairs.

3.4 Neural Network Architecture

For the successful marriage of word embeddings and extracted features, we try three different architectures as depicted in Figure 2. In the first architecture ($A1$, Figure 2a), we concatenate extracted features of each word of an instance with the corresponding word representations and pass it through a LSTM network followed by dense and output layers. In the second architecture ($A2$, Figure 2b), we do not combine features and word representations together. Rather, we learn sentence embeddings through a LSTM network and then concatenate it with the extracted features before feeding to the dense layer. Finally, in the third architecture ($A3$, Figure 2c), we train separate LSTMs for the extracted features and word embeddings. Subsequently, we merge their representations at the dense layer. The choice of separate LSTMs for the hand-crafted features in architecture $A3$ is driven by the fact that the dimension of a word embedding is usually very high as compared to its corresponding hand-crafted features. If trained together, as in architecture $A1$, extracted features of low dimension usually get overshadowed by the high-dimensional word embeddings. Thus making it non-trivial for the network to learn from the extracted features. Further, to exploit the sequence information of words in a sentence we pass hand-crafted features of each word through a separate LSTM layer. For example, in the following review sentence, there are two positive words ('*liking*' and *'recommending'*) and only one negative word ('*far*'). In a model that takes into account only the simple polar word score, the sentence would have high relevance towards the positive sentiment. However, the sequence information of the phrase "*far from liking and recommending*" dictates the negative sentiment of the sentence.

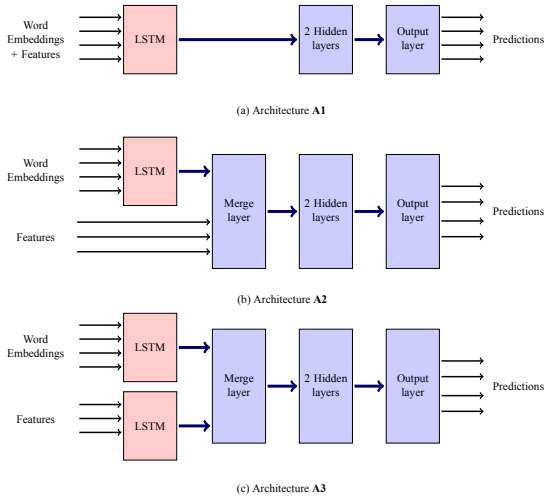


Figure 2: Neural Network Architectures.

"*I'm far from liking and recommending this phone to anyone.*"

In contrast to $A3$, architecture $A2$ does not rely on the sequence information of the extracted features and let the network to learn on its own.

We use 300 dimension word embeddings for the experiments. Each LSTM layer contains 100 neurons while two dense layers contain 100 and 50 neurons respectively.

4 Experimental Results

In this section, we describe the datasets, experimental setup, results and provide necessary analysis.

4.1 Datasets

We use Hindi ABSA dataset released by (Akhtar et al., 2016a) for our evaluation purpose. A total of 5,417 review sentences are present along with 4,509 aspect terms. Each aspect term belongs to one of the four sentiment classes: '*positive*', '*negative*', '*neutral*' and '*conflict*'. We split the dataset into 70%, 10% and 20% as training, development and test, respectively for the experiment. For French case, we use the SemEval-2016 shared task on ABSA (Pontiki et al., 2016) restaurant dataset. It consists of 2,429 review sentences and 3,482 aspect terms. In cross-lingual setup, we utilize English dataset of SemEval-2014 shared task on ABSA (Pontiki et al., 2014) for training and Hindi ABSA dataset for testing. The English dataset comprises of product reviews in two domains i.e. restaurant and laptop. However, we only employ laptop domain dataset as most of the reviews in Hindi ABSA datasets belong to the electronics domain. For training in cross-lingual setup, we combine the training and gold test dataset together. In total, there are 3,845 review sentences comprising of 3,012 aspect terms. For English-French case, we use English restaurant dataset of SemEval-2016 shared task on ABSA (Pontiki et al., 2016) for the training and French ABSA dataset (Pontiki et al., 2016) for evaluation. The SemEval-2016 English restaurant dataset contains 3,365 aspect terms across 2,676 review sentences.

4.2 Experiments

We use Python based neural network library, Keras[4] for implementation. For English-Hindi, all the four classes (namely *positive*, *negative*, *neutral* and *conflict*) were considered, whereas for English-French three classes (all except *conflict* class) were used for classification. Since there is no false class, we use accuracy value as metric to measure the performance of the system. Also, we utilize accuracy value for the direct comparison with the existing state-of-the-art systems. LSTM network is trained with early stopping criteria on (i.e. preserving best learned parameter at each epoch). We set the number of epochs and patience value as 100 & 20 respectively. In other words, we run the experiments for maximum 100 epochs and if validation loss does not reduce for consecutive 20 epochs training stops and reports the best epoch attained so far. As activation function, we utilize *'tanh'* at the intermediate layers, while for classification, we use *'softmax'* at the output layer. To prevent the network from over-fitting, we incorporate an efficient regularization technique called *'Dropout'* (Srivastava et al., 2014). At each layer of training, dropout skips few hidden neurons randomly. We fix dropout rate to be 45% during training while for optimization we use *'adam'* optimizer (Kingma and Ba, 2014).

Experimental results for aspect sentiment classification in multi-lingual and cross-lingual setups are reported in Figure 3 for both the language pairs. In total, we evaluate our model for four cases i.e. **a.** *En-Hi multi-lingual*, **b.** *En-Hi cross-lingual*, **c.** *En-Fr multi-lingual* and **d.** *En-Fr cross-lingual* scenarios. The non-root four-boxed nodes report performance of the respective methods for the four cases. The left subtree represents LSTM based baseline system that utilizes monolingual word embedding (WE) (i.e. word2vec model trained only on 7.2M Hindi and French sentences respectively). Whereas the right subtree represents usage of bilingual word embeddings in all the cases. Comparison between monolingual WE and bilingual WE shows competing results. Monolingual WE (a_M: 63.64%) in multi-lingual scenario performs better than the bilingual WE (a_B: 62.51%) for *English-Hindi* case, while bilingual WE (c_B: 70.89%) reports better performance as compared with monolingual WE (a_M: 66.29%) for *English-French* case. We observe a performance loss of approx. 1 point with bilingual embeddings for English-Hindi case. However, after addressing the problem of data sparsity (i.e. when OOV words are translated and corresponding English word embeddings are computed) the same LSTM network reports an improved accuracy value of 64.83% (a_{BO}) for English-Hindi case, thus observ-

[4] http://keras.io

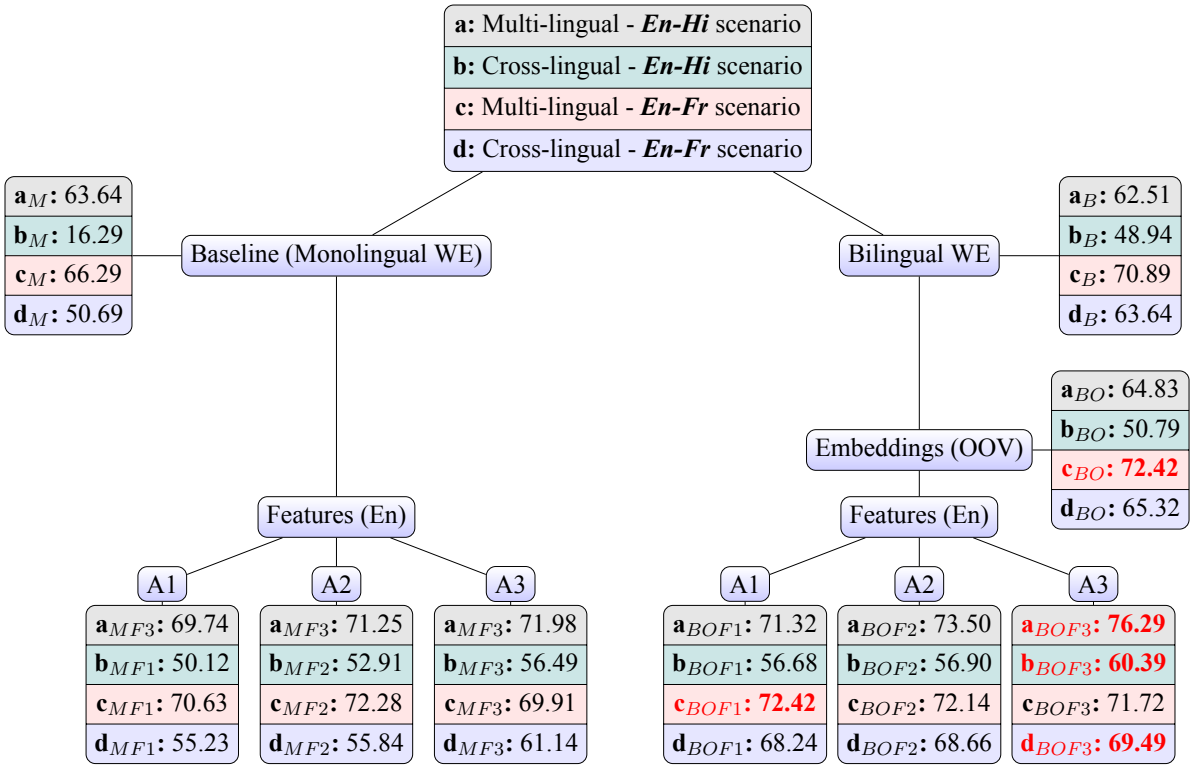


Figure 3: **Aspect classification in Multi-lingual and Cross-lingual setups for English-Hindi and English-French scenarios:** *Left subtree* represents various baselines and their corresponding results. *Right subtree* represents the proposed approach at different levels. Four-box rectangles at non-root levels show accuracy values for **a.** *multi-lingual (En-Hi)*, **b.** *cross-lingual (En-Hi)*, **c.** *multi-lingual (En-Fr)* & **d.** *cross-lingual (En-Fr)* scenarios respectively. **OOV:** Out-of-vocabulary words. **A1:** Word embeddings and extracted features are combined and fed into single LSTM network. **A2:** Extracted features are directly merged with LSTM output of word embedding. **A3:** One LSTM network each for word embeddings and extracted features. Subscripts **M:** *monolingual WE*; **B:** *bilingual WE*; **O:** *Embeddings(OOV)*; **F:** *Features*; **1,2,3:** *Architecture A1, A2 & A3* respectively.

ing a performance increase of more than 2 points. For English-French case, we also observe the improvement with embeddings of OOVs. This suggests that the richness of target language (English) word embeddings helps the system to efficiently solve the problem encountered in resource-poor source language. Since the resources are limited for resource-poor language we try to leverage the high-quality lexicon features of English in our system. Consequently, we introduce the extracted features of Section 3.2 to the network. For English-Hindi multi-lingual scenario, the performance increments from $A1$ to $A2$ to $A3$ indicate that the resource-richness of English language plays a crucial role in classification. While we incorporate English side lexicon features for English-French multi-lingual scenario, we observe no performance improvement like the others. For this case, our system reports an accuracy of 72.42% with (c_{BOF1}) and without (c_{BO}) the use of extra features.

Results of cross-lingual setup for English-Hindi case, where we train the network utilizing English dataset and evaluate the model on Hindi dataset, are reported in row 2 of the four-boxed nodes in Figure 3. The baseline model for cross-lingual setups (left subtree of Figure 3) employs monolingual word embeddings of English and Hindi for training and testing respectively. Since the vector spaces of two different languages are completely unrelated, it is no surprise that the baseline system achieves merely 16.29% (b_M) accuracy. Using only the bilingual word embeddings the system achieves 48.94% (b_B) accuracy. By increasing the coverage of input word embeddings using machine translation the proposed system obtains an increased accuracy of 50.79% (b_{BO}). This improvement in accuracy, again, justifies the use of translated words for obtaining the word embeddings. Further, with the inclusion of target-side lexicon based features our proposed approach reports a significant performance improvement of approximately 6-10 points for all the three archi-

tectures (b_{BOF1}, b_{BOF2} & b_{BOF3}).

Results of English-French cross-lingual scenario are reported in row 4 of the four-boxed nodes in Figure 3. We observe similar phenomenon in cross-lingual setup with the English-French case as well. The baseline system, where we utilize separate monolingual WE for training and testing in English and French respectively, reports an accuracy of 50.69% (d_M), while employing bilingual embeddings the system obtains a sharp jump of approx. 13 points with an accuracy value of 63.64% (d_B). Further, with the inclusion of OOV words and lexicon features performance of the system improves to 65.32% (d_{BO}) and 69.49% (d_{BOF3}), respectively.

We observe four phenomena from these results: i) use of lexicon-based features is the driving force in predicting the sentiment; ii) qualitative lexicons of the resource-rich language can assist in solving the problems of resource-poor languages; iii) embeddings of the OOV words improves the performance of the system with or without assistance of extra features; and iv) use of separate LSTMs (one for word embeddings and the other for features) helps the network to efficiently extract relevant features for prediction without interfering each other (except for the multi-lingual English-French scenario).

4.3 Comparative Analysis

Comparative results reported in Figure 4 show that our proposed system clearly outperforms the baseline model in both the setups and for both the language pairs. In multi-lingual setup, we compare the proposed model against three state-of-the-art systems (Akhtar et al., 2016a; Singhal and Bhattacharyya, 2016; Akhtar et al., 2016b) for English-Hindi case. An accuracy of 65.96% was reported by the system (Akhtar et al., 2016b), while the system (Singhal and Bhattacharyya, 2016) obtained an accuracy of 68.31%. However, our proposed system reports an accuracy of 76.29%, which is approx. 10% & 8% higher compared to the systems of (Akhtar et al., 2016b) and (Singhal and Bhattacharyya, 2016) respectively. In English-French case, our proposed system reports an improvement of approx. 6 points over the baseline. For cross-lingual setup in English-Hindi case, we compare our proposed method with the state-of-the-art system proposed in (Barnes et al., 2016; Singhal and Bhattacharyya, 2016). On the same dataset their systems reported to have achieved an accuracies

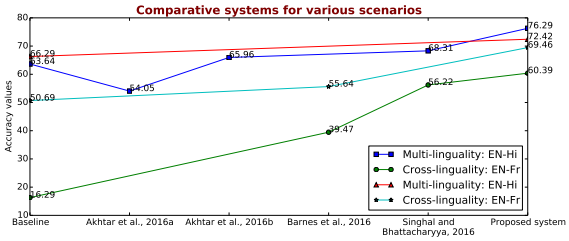


Figure 4: Comparison with the baseline and state-of-the-art methods.

of 39.47% & 56.22% as compared to 60.39% of our proposed system. In English-French case, the system proposed in (Barnes et al., 2016) obtains accuracy value of 55.64% against 69.49% in our proposed architecture. Statistical significance tests (*t*-test) confirm that performance increments in the proposed model are significant w.r.t. state-of-the-art methods with *p*-value=0.03 and *p*-value=0.01 respectively in multi-lingual and cross-lingual setups.

4.4 Discussions

The prime motivation of our current work is to minimize the effect of *data sparsity* while learning through deep neural network architecture. For this, we propose to use bilingual embeddings computed from a parallel corpus, which is created utilizing a MT system. Similarly, absence of a large aligned corpus in resource-poor language can be addressed through the application of a MT system. Since, the MT system is not fully accurate, there must be some errors introduced while translating. This, in turn, affects the bilingual word embedding. Another limitation of our work is that 7.2M sentences is not a big number in terms of word embedding computation. However, the underlying method performs considerably better compared to the state-of-the-art systems, even with all these constraints.

To show the effectiveness of bilingual embeddings in minimizing data sparsity, we also experiment with a mono-lingual Hindi embeddings computed on 53M sentences. Following the proposed approach (except computing embeddings for OOV words), we obtain an accuracy of 77.74% in aspect classification task. Table 1 shows comparison with mono-lingual and multi-lingual approach for classification. Despite all those limitations discussed above (i.e. SMT error & corpus size), the proposed method with bilingual embeddings (76.29%) performs considerably at par against the monolin-

gual embeddings created from a very large corpus of 53M (77.74%). However, the monolingual WE computed using the same amount of corpus (i.e. 7.2M sentences) produces an accuracy of only 63.64%. Further with the help of lexicon based features accuracy of this system increases to 70.86% (compared to 76.29% of our proposed model). It is also to be observed that performance of the system is improved by just including representations of the OOV words. Performance of the proposed system would have been much better if we would not have above mentioned limitations.

Models	Bilingual	Models	Monolingual	
	Size = 7.2M		Size = 7.2M	Size = 53M
Bilingual	62.51	Monolingual	63.64	68.74
Bilingual + Embedding (OOV)	64.83			
Bilingual + Embedding (OOV) + Feature (Eng)	76.29	Monolingual + Feature (Eng)	70.86	77.74

Table 1: Comparative analysis of monolingual embeddings and bilingual embeddings in multi-lingual setup.

4.5 Error Analysis

We perform error analysis on the obtained results. Quantitatively, '*neutral*' is the most problematic class in both multi-lingual and cross-lingual setups. It mainly confuses with '*positive*' class. Approximately, 20% & 40% of '*neutral*' instances are tagged as '*positive*' in multi-lingual and cross-lingual setups, respectively. Our system does not predict '*conflict*' class at all, possibly due to the insufficient number of instances for training. Qualitatively, following are the few cases where our system performs below par.

- **Lack of polar information inside context:** Our system finds it challenging to classify sentiment of the aspect terms whose polar information lie outside the context window. In the following sentence aspect term is 'वज़न |weight' and the actual sentiment towards it is positive. The polar information 'तुलना में लगभग आधा |about half as compared' and 'हल्का |lighter' are far from the aspect term, hence, not captured within the context window.
  Devanagari: इसका वज़न नए आईपैड की तुलना में लगभग आधा है और यह अन्य उपलब्ध 7-इंच टेबलेट्स से भी हल्का है।
  Transliteration: isakA vaZana nae AIpaiDa kI tulanA meM lagabhaga AdhA hai aura yaha anya upalabdha 7-iMcha TebaleTsa se bhI halkA hai.
  Translation: Its weight is about half as compared to the new iPad and it is lighter than other available 7-inch tablets.

- **Implicit sentiment:** Presence of implicit sentiment is not correctly classified by the proposed system. Following review contains 'बनावट |built' as an aspect term and its negative sentiment is derived from the phrase 'प्लास्टिक फील |plastic feel'.
  Devanagari: इस टेबलेट की बनावट काफी प्लास्टिक फील देता है।
  Transliteration: isa TebaleTa kI banAvaTa kAphI plAsTika phIla detA hai.
  Translation: The built of this tablet gives a fairly plastic feel.

5 Conclusion

In this paper, we present a deep learning based LSTM architecture built on top of bilingual word embeddings for aspect level sentiment classification. Bilingual word embeddings try to bridge the language barrier between a resource-rich and resource-poor languages in a shared vector space. We propose to reduce the effect of data sparsity in a resource-poor language word embeddings by projecting OOV words into target side and utilize the target side word embeddings. In addition, we also exploit various resources of English for assisting the proposed model. We show the effectiveness of the proposed method in two different setups, i.e. multi-lingual and cross-lingual. Experimental results show that the proposed system outperforms various state-of-the-art systems in both the setups. In future, we would like to explore the application of proposed method in another aspect level sentiment analysis task known as aspect term extraction or opinion target extraction.

6 Acknowldgements

Asif Ekbal acknowledges Young Faculty Research Fellowship (YFRF), supported by Visvesvaraya PhD scheme for Electronics and IT, Ministry of Electronics and Information Technology (MeitY), Government of India, being implemented by Digital India Corporation (formerly Media Lab Asia).

References

Md Shad Akhtar, Asif Ekbal, and Pushpak Bhattacharyya. 2016a. Aspect based Sentiment Analysis in Hindi: Resource Creation and Evaluation.

In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016), May 23-28, 2016*. European Language Resources Association (ELRA), Portorož, Slovenia, pages 2703–2709.

Md Shad Akhtar, Ayush Kumar, Asif Ekbal, and Pushpak Bhattacharyya. 2016b. A Hybrid Deep Learning Architecture for Sentiment Analysis. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING 2016): Technical Papers, December 11-16, 2016*. Osaka, Japan, pages 482–493.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC 2010), May 17-23, 2010*. European Language Resources Association (ELRA), Valletta, Malta, pages 2200–2204.

Dzmitry Bahdanau, Tom Bosc, Stanislaw Jastrzebski, Edward Grefenstette, Pascal Vincent, and Yoshua Bengio. 2017. Learning to Compute Word Embeddings On the Fly. *CoRR* abs/1706.00286. http://arxiv.org/abs/1706.00286.

Akshat Bakliwal, Piyush Arora, and Vasudeva Varma. 2012. Hindi Subjective Lexicon: A Lexical Resource For Hindi Polarity Classification. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC 2012), May 21-27, 2012*. Istanbul, Turkey, pages 1189–1196.

A. R. Balamurali, Aditya Joshi, and Pushpak Bhattacharyya. 2012. Cross-Lingual Sentiment Analysis for Indian Languages using Linked WordNets. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING): Posters, 8-15 December 2012*. Mumbai, India, pages 73–82.

Jeremy Barnes, Patrik Lambert, and Toni Badia. 2016. Exploring Distributional Representations and Machine Translation for Aspect-based Cross-lingual Sentiment Classification. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING 2016): Technical Papers, December 11-16, 2016*. Osaka, Japan, pages 1613–1623.

Bhuwan Dhingra, Hanxiao Liu, Ruslan Salakhutdinov, and William W. Cohen. 2017. A Comparative Study of Word Embeddings for Reading Comprehension. *CoRR* abs/1703.00993. http://arxiv.org/abs/1703.00993.

Xiaowen Ding, Bing Liu, and Philip S. Yu. 2008. A Holistic Lexicon-based Approach to Opinion Mining. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*. ACM, New York, NY, USA, WSDM '08, pages 231–240.

Deepak Kumar Gupta, Kandula Srikanth Reddy, Asif Ekbal, et al. 2015. PSO-ASent: Feature Selection Using Particle Swarm Optimization for Aspect Based Sentiment Analysis. In *Natural Language Processing and Information Systems (NLDB 2015), June 17-19 2015*. Springer, Passau, Germany, pages 220–233.

Vasileios Hatzivassiloglou and Kathleen R. McKeown. 1997. Predicting the Semantic Orientation of Adjectives. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Madrid, Spain, pages 174–181. https://doi.org/10.3115/976909.979640.

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

VS Jagtap and Karishma Pawar. 2013. Analysis of Different Approaches to Sentence-level Sentiment Classification. *International Journal of Scientific Engineering and Technology (ISSN: 2277-1581) Volume* 2:164–170.

Aditya Joshi, AR Balamurali, and Pushpak Bhattacharyya. 2010. A Fall-back Strategy for Sentiment Analysis in Hindi: a Case Study. In *Proceedings of the 8th International Conference on Natural Language Processing (ICON 2010)*. Kharagpur, India.

Rasoul Kaljahi and Jennifer Foster. 2016. Detecting Opinion Polarities using Kernel Methods. In *Proceedings of the Workshop on Computational Modelling of People's Opinions, Personality, and Emotions in Social Media*. Osaka, Japan, pages 60–69.

Soo-Min Kim and Eduard Hovy. 2004. Determining the sentiment of opinions. In *Proceedings of the 20th international conference on Computational Linguistics*. Association for Computational Linguistics, page 1367.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980. http://arxiv.org/abs/1412.6980.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*. Association for Computational Linguistics, pages 177–180.

Ayush Kumar, Sarah Kohail, Asif Ekbal, and Chris Biemann. 2015. IIT-TUDA: System for Sentiment Analysis in Indian Languages Using Lexical Acquisition. In *Mining Intelligence and Knowledge Exploration*, Springer, pages 684–693.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual Word Representations with Monolingual Quality in Mind. In *NAACL Workshop on Vector Space Modeling for NLP*. Denver, United States, pages 151–159.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781* .

Saif M. Mohammad, Mohammad Salameh, and Svetlana Kiritchenko. 2016. How Translation Alters Sentiment. *Journal of Artificial Intelligence Research* 55(1):95–130. http://dl.acm.org/citation.cfm?id=3013558.3013562.

Bo Pang and Lillian Lee. 2005. Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '05, pages 115–124. https://doi.org/10.3115/1219840.1219855.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar, pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammad AL-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphee De Clercq, Veronique Hoste, Marianna Apidianaki, Xavier Tannier, Natalia Loukachevitch, Evgeniy Kotelnikov, Núria Bel, Salud María Jiménez-Zafra, and Gülşen Eryiğit. 2016. SemEval-2016 Task 5: Aspect Based Sentiment Analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. Association for Computational Linguistics, San Diego, California, pages 19–30. http://www.aclweb.org/anthology/S16-1002.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. SemEval-2014 Task 4: Aspect Based Sentiment Analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Dublin, Ireland, pages 27–35.

Soujanya Poria, Erik Cambria, and Alexander Gelbukh. 2016. Aspect Extraction for Opinion Mining with a Deep Convolutional Neural Network. *Knowledge-Based Systems* 108:42–49.

Prerana Singhal and Pushpak Bhattacharyya. 2016. Borrow a Little from your Rich Cousin: Using Embeddings and Polarities of English Words for Multilingual Sentiment Classification. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING 2016): Technical Papers, December 11-16, 2016*. Osaka, Japan, pages 3053–3062.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15:1929–1958. http://jmlr.org/papers/v15/srivastava14a.html.

P. D. Turney. 2002. Thumbs up or thumbs down?: Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Association for Computational Linguistics (ACL)*. Philadelphia, USA, pages 417–424.

Janyce Wiebe and Rada Mihalcea. 2006. Word Sense and Subjectivity. In *Proceedings of the 21st International Conference on Computational Linguistics (COLING) and the 44th Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL-44, pages 1065–1072. https://doi.org/10.3115/1220175.1220309.

Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. 2016. Cross-Lingual Sentiment Classification with Bilingual Document Representation Learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*. Berlin, Germany, pages 1403–1412.

SRL4ORL: Improving Opinion Role Labeling Using Multi-Task Learning With Semantic Role Labeling

Ana Marasović and **Anette Frank**
Research Training Group AIPHES
Department of Computational Linguistics
Heidelberg University
{marasovic,frank}@cl.uni-heidelberg.de

Abstract

For over a decade, machine learning has been used to extract *opinion-holder-target structures* from text to answer the question *Who expressed what kind of sentiment towards what?*. Recent neural approaches do not outperform the state-of-the-art feature-based models for Opinion Role Labeling (ORL). We suspect this is due to the scarcity of labeled training data and address this issue using different multi-task learning (MTL) techniques with a related task which has substantially more data, i.e. Semantic Role Labeling (SRL). We show that two MTL models improve significantly over the single-task model for labeling of both holders and targets, on the development and the test sets. We found that the vanilla MTL model which makes predictions using only shared ORL and SRL features, performs the best. With deeper analysis we determine what works and what might be done to make further improvements for ORL.

1 Introduction

Fine-Grained Opinion Analysis (FGOA) aims to: (i) detect opinion expressions (O) that convey attitudes such as sentiments, agreements, beliefs or intentions (like *feared* in example (1)), (ii) measure their intensity (e.g. strong), (iii) identify their holders (H), i.e. entities that express an attitude (e.g. *it*), (iv) identify their targets (T), i.e. entities or propositions at which the attitude is directed (e.g. *violence*) and (v) classify their target-dependent attitude (e.g. negative sentiment)[1].

(1) Australia said [it]$_H$ [**feared**]$_{O_{neg}}$ [violence]$_T$ if voters thought the election had been stolen.

As the commonly accepted benchmark corpus MPQA (Wiebe et al., 2005) uses span-based annotations to represent *opinion entities* (opinions, holders and targets), the task is usually approached with sequence labeling techniques and the BIO encoding scheme (Choi et al., 2006; Yang and Cardie, 2013; Katiyar and Cardie, 2016). Initially pipeline models were proposed which first predict opinion expressions and then, given an opinion, label its *opinion roles*, i.e. holders and targets (Kim and Hovy, 2006; Johansson and Moschitti, 2013). Pipeline models have been substituted with so-called joint models that simultaneously identify all opinion entities, and predict which opinion role is related to which opinion (Choi et al., 2006; Yang and Cardie, 2013; Katiyar and Cardie, 2016). Recently an LSTM-based joint model was proposed (Katiyar and Cardie, 2016) that unlike the prior work (Choi et al., 2006; Yang and Cardie, 2013) does not depend on external resources (such as syntactic parsers or named entity recognizers). The neural variant does not outperform the feature-based CRF model (Yang and Cardie, 2013) in Opinion Role Labeling (ORL).

Both the neural and the CRF joint models achieve about 55% F1 score for predicting which targets relate to which opinions in MPQA. Thus, these models are not yet ready to answer the question this line of research is usually motivated with: *Who expressed what kind of sentiment towards what?*. Our goal is to investigate the limitations of neural models in solving different subtasks of FGOA on MPQA and to gain a better understanding of what is solved and what is next.

We suspect that one of the fundamental obstacles for neural models trained on MPQA is its small size. One way to address scarcity of labeled data is to use multi-task learning (MTL) with appropriate auxiliary tasks. A promising auxiliary task candidate for ORL is Semantic Role Labeling (SRL), the task of predicting predicate-argument structure of a sentence, which answers the question *Who did what to whom, where and when?*.

[1] Examples are drawn from MPQA (Wiebe et al., 2005).

Proceedings of NAACL-HLT 2018, pages 583–594
New Orleans, Louisiana, June 1 - 6, 2018.

	Australia	said	it	feared	violence	if	voters	thought	the	election	had	been	stolen	.
say.01	A0	-	A1	A1	A1	A1	A1	A1	A1	A1	A1	A1	A1	-
fear.01	-	-	**A0**	-	**A1**	AM-ADV	AM-ADV	AM-ADV	AM-ADV	AM-ADV	AM-ADV	AM-ADV	AM-ADV	-
think.01	-	-	-	-	-	-	A0	-	A1	A1	A1	A1	A1	-
steal.01	-	-	-	-	-	-	-	-	A1	A1	-	-	-	-

Table 1: Output of the SRL demo.

Table 1 illustrates the output of the SRL demo[2] for example (1), following the PropBank SRL scheme (Palmer et al., 2005)[3].

SRL4ORL. The semantic roles of the predicate *fear* (marked blue bold) correspond to the opinion roles H and T, according to MPQA. For this reason, the output of SRL systems has been commonly used for feature-based FGOA models (Kim and Hovy, 2006; Johansson and Moschitti, 2013; Choi et al., 2006; Yang and Cardie, 2013). Additionally, a considerable amount of training data is available for training SRL models (Table 2 in Sec. 3), which made neural SRL models successful (Zhou and Xu, 2015; Yang and Mitchell, 2017).

Obstacles. Although SRL is similar in nature to ORL, it cannot solve ORL for all cases (Ruppenhofer et al., 2008). In example (2) holder and target of the predicate *please* correspond to A1, A0 semantic roles respectively, wheres for the predicate *fear* in (1) holder and target correspond to A0, A1 respectively. We took into account this observation when deciding on an appropriate MTL model by splitting its parameters into shared and task-specific ones (i.e. hard-parameter sharing).

(2) $[\text{I}]_H^{A1}$ am very $[\textbf{pleased}]_{O_{pos}}$ that [the Council has now approved the Kyoto Protocol thus enabling the EU to proceed with its ratification$]_T^{A0}$.

A further obstacle for properly exploiting SRL training data with MTL could be specificities, inconsistency and incompleteness of the MPQA annotations. In example (3), *Rice* expressed his negative sentiment towards *the three countries in question* by *setting the criteria* which states something negative about those countries: they *are repressive and grave human rights violators [...]*. In this case, the model should not pick any local semantic role for the target.

(3) The criteria $[\textbf{set by}]_{O_{neg}}$ $[\text{Rice}]_H$ are the following: $[\text{the three countries in question}]_T$ are repressive and grave human rights violators, and aggressively seeking weapons [...].

In examples (4–5), the same opinion expression *concerned* realizes different scopes for the target. A model which exploits SRL knowledge could be biased to always label targets as complete SRL role constituents, as in example (5).

(4) Rice told us $[\text{the administration}]_H$ was $[\textbf{concerned}]_{O_{neg}}$ that $[\text{Iraq}]_T$ would take advantage of the 9/11 attacks.

(5) $[\text{The Chinese government}]_H$ is deeply $[\textbf{concerned}]_{O_{neg}}$ about [the sudden deterioration in the Middle East situation$]_T$, Tang said.

Regarding incompleteness, prior work (Katiyar and Cardie, 2016) has shown that their model makes reasonable predictions in sentences which do not have annotations at all, e.g. $[\text{mothers}]_H$ $[\textbf{care}]_O$ for $[\text{their young}]_T$, in: *From the fact that mothers care for their young, we can not deduce that they ought to do so, Hume argued.*

The examples above show that incorporating SRL knowledge via multi-task learning is a reasonable way to improve ORL, but at the same time they alert us that given the specificities of MPQA and ORL annotations in general, it is not obvious whether MTL can overcome divergences in the annotation schemes of opinion and semantic role labeling. We investigate this research question by adopting one of the recent successful architectures for SRL (Zhou and Xu, 2015) and experiment with different multi-task learning frameworks.

Our contributions are: (i) we adapt a recently proposed neural SRL model for ORL, (ii) we enhance the model using different MTL techniques with SRL to tackle the problem of scarcity of labeled data for ORL, (iii) we show that most of the MTL models improve the single-task model for labeling of both holders and targets on development and test sets, and two of them make yield significant improvements, (iv) by deeper analysis we provide a better understanding of what is solved and where to head next for neural ORL.

[2] http://barbar.cs.lth.se:8081

[3] Henceforth we use the PropBank SRL framework.

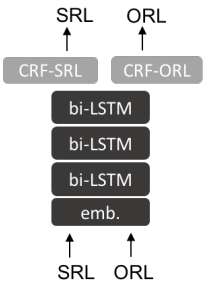


Figure 1: Fully-shared (FS) MTL.

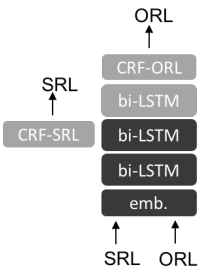


Figure 2: Hierarchical-MTL (H-MTL).

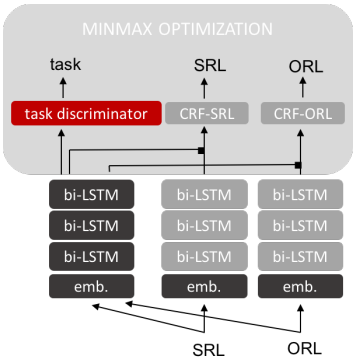


Figure 3: (Adversarial) state-private ((A)SP) MTL.

2 Neural MTL for SRL and ORL

Neural multi-task learning (MTL) receives a lot of attention and new MTL architectures emerge regularly. Yet there is no clear consensus which MTL architecture to use in which conditions. We experiment with well-received architectures that could adapt to different cases of ORL from Section 1.

As a general neural architecture for single- and multi-task learning we use the recently proposed SRL model (Zhou and Xu, 2015) (Z&X-STL) which successfully labels semantic roles without any syntactic guidance. This model consists of a stack of bi-directional LSTMs and a CRF which makes the final prediction. The inputs to the first LSTM are not only token embeddings but three additional features: embedding of the predicate, embedding of the context of the predicate and an indicator feature (1 if the current token is in the predicate context, 0 otherwise). Thus, every sentence is processed as many times as there are predicates in it. Adapting this model for labeling of opinion roles is straightforward, the only difference being that opinion expressions can be multi-words and only two opinion roles are assigned.

MTL techniques aim to learn several tasks jointly by leveraging knowledge from all tasks. In the context of neural networks, MTL is commonly used in such a way that it is predefined which layers have tied parameters and which are task-specific (i.e. hard-parameter sharing). There are various ways of defining which parameters should be shared and how to train them.

Fully-shared (FS) MTL model. A fully-shared model (Fig. 1) shares all parameters of the general model except the output layer. Each task has a task-specific output layer which makes the prediction based on the representation produced by the final LSTM. When training on a mini-batch of a certain task, parameters of the output layer of the other tasks are not updated. This model should be effective for constructions with a clear mapping between opinion and semantic roles such as $\{\text{H} \mapsto \text{A0}, \text{T} \mapsto \text{A1}\}$ as in example (1) (Sec. 1).

Hierarchical MTL (H-MTL) model. For NLP applications, often some given (high-level) task is supposed to benefit from another (low-level) task more than the other way around, e.g. parsing from POS tagging. This intuition lead to designing hierarchical MTL models (Søgaard and Goldberg, 2016; Hashimoto et al., 2017) in which predictions for low-level tasks are not made on the basis of the representation produced at the final LSTM, but on the representation produced by a lower-layer LSTM (Fig. 2). Task-specific layers atop shared layers could potentially give the model more power to distinguish or ignore certain semantic roles. If so, this MTL model is more suitable for examples like (2) and (3) (Sec. 1).

Shared-private (SP) MTL model. In the state-private model, in addition to the stack of shared LSTMs, each task has a stack of task-specific LSTMs (Liu et al., 2017) (Fig. 3). Representations at the outermost shared LSTM and the task-specific LSTM are concatenated and passed to the task-specific output layer. The ORL representation produced independently from SRL gives the model the ability to utilize the shared and entirely task-specific information. For labeling of targets, it is expected that for examples (1) & (5) the model relies mostly on the shared representation, for examples (2) & (4) on both shared and ORL-specific representations, and for example (3) solely on the ORL-specific representation.

Adversarial shared-private (ASP) model. The limitation of the SP model is that it does not prevent the shared layers from capturing task-specific features. To ensure this, ASP extends the

	task	train size	dev size	test size	$\lvert\mathcal{Y}\rvert$
CoNLL'05	SRL	90750	3248	6071	106
MPQA (4-CV)	ORL	3141.25	1055	1036.75	7
MPQA (10-CV)	ORL	3516.3	1326	349.3	7

Table 2: Datasets w/ nb. of SRL predicates/ORL opinions in train, dev & test set, size of label inventory.

SP model with a *task discriminator* (Liu et al., 2017). The task discriminator (Fig. 3, marked red) predicts to which task the current batch of data belongs, based on the representation produced by the shared LSTMs. If the shared LSTMs are task-invariant, the discriminator should perform badly. Thus, we update the shared parameters to maximize the discriminator's cross-entropy loss. At the same time we want the discriminator to challenge the shared LSTMs, so we update the discriminator's parameters to minimize its cross-entropy loss. This minmax optimization is known as *adversarial training* and recently it gained a lot of attention for NLP applications (Liu et al., 2017; Chen et al., 2017; Kim et al., 2017; Qin et al., 2017; Wu et al., 2017; Gui et al., 2017; Li et al., 2017; Zhang et al., 2017; Joty et al., 2017).

3 Experimental setup

3.1 Datasets

For SRL we use the newswire CoNLL-2005 shared task dataset (Carreras and Màrquez, 2005), annotated with PropBank predicate-argument structures. Sections 2-21 of the WSJ corpus (Charniak et al., 2000) are used for training and section 24 as dev set. The test set consists of section 23 of WSJ and 3 sections of the Brown corpus.

For ORL we use the manually annotated MPQA 2.0. corpus (Wiebe et al., 2005; Wilson, 2008). It mostly contains news documents, but also travel guides, transcripts of spoken conversations, e-mails, fundraising letters, textbook chapters and translations of Arabic source texts.

We report detailed pre-processing of MPQA[4] and data statistics in the Supplementary Material.

3.2 Evaluation metrics

For both tasks we adopt evaluation metrics from prior work. For SRL, precision is defined as the proportion of semantic roles predicted by a system which are correct, recall is the proportion of gold roles which are predicted by a system, F1 score is the harmonic mean of precision and recall.

In case of ORL, we report 10-fold CV[5] and repeated 4-fold CV with *binary F1 score* and *proportional F1 score*, for holders and targets separately. *Binary precision* is defined as the proportion of predicted holders (targets) that overlap with the gold holder (target), *binary recall* is the proportion of gold holders (targets) for which the model predicts an overlapping holder (target). *Proportional recall* measures the proportion of the overlap between a gold holder (target) and an overlapping predicted holder (target), *proportional precision* measures the proportion of the overlap between a predicted holder (target) and an overlapping gold holder (target). F1 scores are the harmonic means of the corresponding precision and recall.

3.3 Training details

We evaluate our models using two evaluation settings. First, we follow Katiyar and Cardie (2016) which set aside 132 documents for development and used the remaining 350 documents for 10-fold CV. However, in the 10-fold CV setting, the size of the tests sets is 3 times smaller than the dev set size (Table 2, row 3), and, consequently, results in high-variance estimates on the test sets. Therefore we additionally evaluate our models with 4-fold CV. We set aside 100 documents for development and use 25% of the remaining documents for testing. The resulting test sets are comparable in size to the dev set (Table 2, row 2). We run 4-fold CV twice with two different random seeds. We do not tune hyperparameters (HPs), but follow suggestions proposed in the thorough HP study for sequence labeling tasks (Reimers and Gurevych, 2017). HPs can be found in the Supplementary Material.

4 Results

We evaluate all models after every $\lceil \frac{\text{train size}}{\text{batch size}} \rceil$ iteration on the ORL dev set and save them if they achieve a higher arithmetic mean of proportional F1 scores of holders and targets on the ORL dev set. The saved models are used for testing.

We report the mean of F1 scores over 10 folds and the standard deviation (appears as a subscript) of all models in Table 3. We report the mean of F1 scores over 4 folds and 2 different seed (8 evalua-

[4]Examples how to use our scripts can be found at https://github.com/amarasovic/naacl-mpqa-srl4orl/blob/master/generate_mpqa_jsons.py.

[5]We used the same splits as the prior work (Katiyar and Cardie, 2016). We thank the authors for providing the splits.

	dev (MPQA)				test (MPQA)				
	holder		target			holder		target	
	binary F1	prop. F1	binary F1	prop. F1		binary F1	prop. F1	binary F1	prop. F1
Z&X-STL	$80.15_{1.10}$	$76.87_{1.26}$	$74.62_{0.67}$	$70.23_{1.04}$	Z&X-STL	$80.24_{2.91}$	$77.98_{2.90}$	$76.30_{2.55}$	$71.18_{2.55}$
FS-MTL	$83.68^{\bullet}_{0.44}$	$81.45^{\bullet}_{0.58}$	$\mathbf{76.23}^{\bullet}_{0.75}$	$\mathbf{73.01}^{\bullet}_{0.93}$	FS-MTL	$83.47^{\bullet}_{2.26}$	$81.80^{\bullet}_{2.26}$	$\mathbf{77.60}_{2.52}$	$\mathbf{73.77}_{2.28}$
H-MTL	$\mathbf{84.14}^{\bullet}_{0.72}$	$\mathbf{81.86}^{\bullet}_{0.48}$	$76.11^{\bullet}_{0.61}$	$72.55^{\bullet}_{0.73}$	H-MTL	$\mathbf{84.03}^{\bullet}_{2.65}$	$\mathbf{82.34}^{\bullet}_{2.51}$	$77.41_{2.14}$	$73.10_{1.96}$
SP-MTL	$82.18^{\bullet\diamond}_{0.89}$	$79.66^{\bullet\diamond}_{0.72}$	$74.99^{\diamond}_{1.17}$	$71.32^{\diamond}_{1.81}$	SP-MTL	$82.19^{\bullet}_{2.49}$	$80.11^{\bullet\diamond}_{2.36}$	$76.01_{3.03}$	$71.51_{3.34}$
ASP-MTL	$82.63^{\bullet\diamond}_{0.84}$	$80.20^{\bullet\diamond}_{0.99}$	$74.24^{\bullet\diamond}_{0.58}$	$70.16^{\diamond}_{1.29}$	ASP-MTL	$83.15^{\bullet}_{2.92}$	$81.12^{\bullet}_{2.66}$	$75.89_{2.66}$	$71.21_{2.78}$

Table 3: ORL 10-fold CV results.

	dev (MPQA)				test (MPQA)				
	holder		target			holder		target	
	binary F1	prop. F1	binary F1	prop. F1		binary F1	prop. F1	binary F1	prop. F1
Z&X-STL	$79.73_{1.19}$	$77.06_{1.14}$	$76.09_{0.94}$	$70.45_{1.07}$	Z&X-STL	$80.42_{1.92}$	$77.48_{2.06}$	$73.84_{1.17}$	$67.03_{2.13}$
FS-MTL	$\mathbf{83.58}^{\bullet}_{0.69}$	$\mathbf{82.16}^{\bullet}_{0.59}$	$\mathbf{78.32}^{\bullet}_{1.57}$	$\mathbf{75.09}^{\bullet}_{2.27}$	FS-MTL	$\mathbf{83.67}^{\bullet}_{1.52}$	$\mathbf{81.59}^{\bullet}_{1.50}$	$77.04^{\bullet}_{1.45}$	$73.01^{\bullet}_{2.53}$
H-MTL	$82.36^{\bullet\diamond}_{0.81}$	$80.84^{\bullet\diamond}_{0.98}$	$78.11^{\bullet}_{0.82}$	$74.89^{\bullet}_{1.33}$	H-MTL	$82.80^{\bullet}_{1.87}$	$80.40^{\bullet}_{1.91}$	$\mathbf{77.12}^{\bullet}_{1.34}$	$\mathbf{73.16}^{\bullet}_{1.78}$
SP-MTL	$82.21^{\bullet\diamond}_{0.79}$	$80.23^{\bullet\diamond}_{0.88}$	$76.14^{\diamond}_{1.18}$	$71.14^{\diamond}_{0.97}$	SP-MTL	$82.51^{\bullet}_{2.17}$	$80.03^{\bullet}_{2.00}$	$74.61^{\diamond}_{1.32}$	$68.70^{\diamond}_{2.32}$
ASP-MTL	$81.41^{\diamond}_{1.27}$	$79.39^{\bullet\diamond}_{1.45}$	$76.49_{1.39}$	$72.13^{\bullet}_{1.87}$	ASP-MTL	$81.77^{\diamond}_{1.74}$	$79.32^{\diamond}_{1.62}$	$74.92^{\bullet\diamond}_{0.84}$	$69.89^{\bullet}_{1.80}$

Table 4: ORL repeated 4-fold CV results.

tions) and the standard deviation of all models in Table 4. Evaluation metrics follow Section 3.2.

We mark significant difference between MTL models and the single-task (Z&X-STL) model, observed using a Kolmogorov-Smirnov significance test ($p < 0.05$) (Massey Jr, 1951), with $\bullet$ in superscript and between the FS-MTL model and other MTL models with $\diamond$.

STL vs. MTL. In the 10-fold CV evaluation setting (Table 3), the FS-MTL and the H-MTL models improve over the Z&X-STL model in all evaluation measures, for both holders and targets. When evaluated in the repeated 4-fold CV setting (Table 4), all MTL models improve over the Z&X-STL model in all evaluation measures, for both holders and targets.

The FS-MTL and the H-MTL models improve *significantly* in all evaluation measures, for both holders and targets, on both dev and test sets, when evaluated with repeated 4-fold CV. With 10-fold CV the improvements are also significant, except for targets on the test set. This is probably due to the small size of the test sets (Table 2, row 3), which results in a high-variance estimate. Indeed, standard deviations on the 10-fold CV test sets are always much higher compared to the dev set or to the test sets of 4-fold CV.

It is not surprising that larger improvements are visible in the labeling of holders. They are usually short, less ambiguous and often presented with the A0 semantic role, whereas annotating targets is a challenging task even for humans.[6]

Larger improvements are visible for proportional F1 score than for binary F1 score. That is, more data and SRL knowledge helps the model to better annotate the scope of opinion roles.

Comparing MTL models. In Section 2 we introduced MTL models with task-specific LSTM layers hypothesizing that these layers should give MTL models more power to adapt to a variety of potentially problematic cases that we illustrated in the Introduction. However, our results show that the FS-MTL model performs significantly better or comparable to MTL models that include task-specific layers. Reimers and Gurevych (2017) show that MTL is especially sensitive to the selection of HPs. Thus, a firm and solid comparison of the different MTL models requires thorough HP optimization, to properly control the number of parameters and regularization of the models. We leave HP optimization for future work.

5 Analysis

Our aim in this section is to analyze what the proposed models are good at, in which ways MTL improves over the single-task ORL model and what could be done to achieve further progress.

We evaluate the FS-MTL and the Z&X-STL

[6] Wilson (2008) reports annotator agreement for target labeling of 86.00 binary F1 score.

1	*Malinga* $_{FS,ZX}$ **said** according to the guidelines in the booklet, the election had been legitimate .
2	movie um-hum that 's interesting so that was a good movie too well do *you* $_{FS,ZX}$ **think** we've covered baseball i think so okay well have a good night
3	*The nation* $_{FS,ZX}$ should certainly **be concerned** about the plans to build a rocket launch pad , work on the infrastructure for which is due to start in 2002 , with launches beginning from 2004 .
4	Bam on Sunday said *she* $_{FS,ZX}$ **believed** Zimbabwe's election was not free and fair , adding they were not in line with international standards as well as those of her organisation .
5	The majority report , endorsed only by the ANC , said *the observer mission* $_{FS,ZX}$ **had noted** that over three million Zimbabweans had cast their votes and this substantially represented the will of the people .

Table 5: The dev examples for which both models (FS-MTL, Z&X-STL) *correctly* predict the *holder* in 6/8 trials.

1	Indonesia **has come under pressure** from several quarters to *take tougher action against alleged terrorist leaders* but has played down the threat $_{ZXFS}$.
2	Mugabe even talked about his **desire** to *keep safeguarding Zimbabwe 's sovereignty and land* $_{ZX}$ *in spirit* $_{FS}$ *when he dies* , a dream which the veteran leader said forced him to sacrifice a bright teaching career in the 1950s to lead [...].
3	Under his blueprint , the government **hopes** to *stabilize the economy* through curtailing state expenditure , reforming public enterprises and expanding agriculture $_{FS,ZX}$.
4	He said those who thought the election process would be rigged were supporters of the MDC party , adding that they were prejudging and **wanted** to *direct the process* $_{FS,ZX}$.
5	People in the rural areas support the ruling party because our party has been genuine on **its policy** on *land reform* $_{FS,ZX}$.

Table 6: The dev examples for which both models (FS-MTL, Z&X-STL) *correctly* predict the *target* in 6/8 trials.

	easy	hard
% opinions that are predicates	91.32	93.33
% holders that are subjects	77.84	38.79
% holders that are A0 roles	74.10	33.33
avg. distance between holders & opinions	1.54	7.56

Table 7: Statistics of *holder* prediction.

	easy	hard
% opinions that are predicates	92.58	89.20
% target's heads that are objects	22.12	14.77
% targets that are A1 roles	70.62	42.61
% targets that are A2 roles	9.00	0.57
avg. distance between targets & opinions	2.29	8.46

Table 8: Statistics of *target* prediction.

models on the ORL dev set using 4-fold CV repeated twice with different seeds (8 evaluation trials). We say that a model predicts a role of a given opinion expression *correctly* if the model predicts a role that overlaps with the correct role in at least 6 out of 8 evaluation trials. If a model predicts a role that overlaps with the correct role in at most 2 out of 8 trials, we say that the model predicts the role *incorrectly*. The requirement on 6-8 (in)correct predictions reduces the risk of analyzing inconsistent predictions and enables us to draw firmer conclusions. We analyze the following scenarios:

(i) both the FS-MTL model and the Z&X-STL model make correct predictions (Tables 5–6)

(ii) the FS-MTL model makes a correct prediction, while the Z&X-STL makes an incorrect prediction (Tables 11–12)

(iii) both models make wrong predictions (Tables 9–10)

In the following, we categorize predictions in case (i) as *easy cases*, and predictions in case (iii) as *hard cases*.

In Tables 5–6 and 9–12, the opinion expression is bolded, the correct role is italicized, predictions of the FS-MTL model are colored blue (subscript *FS*), predictions of the Z&X-STL model are colored yellow (subscript *ZX*) and green marks predictions where both models agree. For simplicity, we show only holders or targets, although the models predict both roles jointly.

What works well? There are 668/1055 instances in the dev set for which both models predict holders correctly, and 663/1055 for targets.

Examples 1–5 in Table 5 suggest that holders that can be properly labeled by both models (*easy*

1	It would be entirely improper if , in *its* **defense of** Israel $_{FS}$, the United States continues to exert pressure on [...] .
2	Indonesia $_{FS,ZX}$ **has come under pressure** from *several quarters* to take tougher action against alleged terrorist leaders but has played down the threat .
3	Australia should adhere to the *Cardinal Principle of International Law* , which **states** that all nations in the world must first respect and promote the humanitarian interests and progress of all humankind .
4	*The department* said that it will cost $ 600 for an HIV/AIDS patient per year at this time , and the following years this cost is **expected** to stand at just $ 400/year for one patient as the production of such drugs becomes stable .
5	The Organisation of African Unity OAU $_{ZX}$ also backed Zimbabwean President Robert Mugabe 's re-election , with its observer team $_{FS,ZX}$ **describing** the poll as " transparent , credible , free and fair " .
6	Regarding the American proposed Anti-Missile Defense System too , neither Russia , China , Japan , nor even the European Union , had shown any enthusiasm ; rather *they* $_{FS}$ had all $_{FS,ZX}$ **expressed** their reserves on the project .
7	The president renewed his pledge to thwart terrorist groups $_{FS,ZX}$ *who* **want** to " mate up " with regimes hoping to acquire weapons of mass destruction and said " nations will come with us " if the US-led war on terrorism is extended .

Table 9: The dev examples for which both models (FS-MTL, Z&X-STL) *incorrectly* predict the *holder* in 6/8 trials.

1	*State-sanctioned land invasions* , several times **declared** illegal by Zimbabwe 's courts , as well as a drought have disrupted Zimbabwe 's food production and famine is already looming in much of the country .
2	But he **told** the nation $_{FS,ZX}$ that in spite of *stiff opposition to the agrarian reforms from powerful Western countries , especially the country 's former colonial power of Britain* , he would press ahead to seize farms from whites and [...] .
3	If the Europeans wish to influence Israel in the political arena – in *a direction* that many in Israel **would support wholeheartedly** – they will not be able to promote their positions in such a manner .
4	They $_{FS,ZX}$ are fully aware that these are dangerous *individuals* , he **said** during a press conference [...] .
5	And her little girl **just complained** , " I don't want to *wash the dishes* " .
6	During *President Bush's speech* , I **thought** of heckling $_{ZX}$; ' What are you going to do with the Kyoto Protocol ? $_{FS}$'
7	At first I **didn't want** to apply for it $_{FS,ZX}$, but the principal called me during the summer months and said , " Sandra the time is running out , you need to apply ".

Table 10: The dev examples for which both models (FS-MTL, Z&X-STL) *incorrectly* predict the *target* in 6/8 trials.

cases) are subjects of their governing heads or A0 roles. The statistics in Table 7 (col. 1, rows 2–3) supports this observation.[7] In contrast, holders that both models predict incorrectly (*hard* cases) are less frequently subjects or A0 roles (col. 2, rows 2–3). Also, *easy* holders are close to the corresponding opinion expression: the average distance is 1.54 tokens (Table 7, row 4), contrary to the *hard* holders with the average distance of 7.56.

Examples 1–5 in Table 6 suggest that targets that can be properly labeled by both models are objects of their governing heads or A1 roles. Table 8, row 3, shows that the majority of the *easy* targets are indeed A1 roles, in contrast to the *hard* targets. Similar to holders, the *easy* targets are in average 7 tokens closer to the opinion expression.

What to do for further improvement? There are 165/1055 instances in the dev set for which both models predict holders incorrectly, and 176 for targets.

As we have seen so far, many holders that are subjects or A0 roles, and targets that are A1 roles, are properly labeled by both models. However, a considerable amount of such holders and targets are not correctly predicted (Table 7–8, col. 2, rows 2–3). Thus our models do not work flawlessly for all such cases. A distinguishing property of the *hard* cases is the distance of the role from the opinion. Thus, future work should advance the model' s ability to capture long-range dependencies.

Examples in Table 9 demonstrate that holders, harder to label with our models, occur with the corresponding opinions in more complicated syntactic constructions. In the first example, the FS-MTL model does not recognize the possessive and is possibly biased towards picking the country (*Israel*), which occurs immediately after the opinion. In the second example, the opinion expression is

[7] The statistics is calculated using the output of mate-tools (Björkelund et al., 2010).

1	Yoshihisa Murasawa , a management consultant for Booz-Allen & Hamilton Japan Inc. , said *his firm* $_{FS,ZX}$ will likely be recommending acquisitions of Japanese companies more $_{ZX}$ often to foreign clients in the future .
2	*The source* $_{FS}$, interviewed by Interfax in Grozny , **expressed confidence** that that the command of the Russian forces in Chechnya would soon " be able to obtain documentary confirmation " that Khattab was dead .
3	*The Commonwealth team earlier* this week $_{FS}$ **said** that " the conditions in Zimbabwe did not adequately allow the free and fair expression of will by the electorate ".
4	Publishing such biased reports will only create **mistrust** among nations $_{FS}$ regarding the objectives and independence of the UN Commission on Human Rights .
5	The Inkatha Freedom Party , Democratic Alliance , New National Party , African Christian Democratic Party , the Pan Africanist Congress and the United Christian Democratic Party $_{ZX}$ had disagreed with the ANC $_{FS}$ **conclusion** .
6	The Nigerian leader , President Olusegun Obasanjo $_{ZX}$, had urged the *minister* $_{FS,ZX}$ not to attack Blair frontally over Britain 's negative position regarding Zimbabwe , but to deal [...] .
7	US diplomats $_{ZX}$ say *Bush* $_{FS,ZX}$ **will seek to support** Kim 's Nobel Prize winning policy by offering new talks with the North , while remaining firm about North Korea 's missile sales and its feared chemical and biological weapons programmes.

Table 11: The dev examples for which the FS-MTL model *correctly* predicts the *holder* in 6/8 trials, whereas the Z&X-STL model predicts *incorrectly* in 6/8 trials.

1	In most cases he **described** *the legal punishments* $_{FS}$ *like floggings and executions of murderers and major drug traffickers that are applied based on the Shria , or Islamic law as human rights violations* .
2	In another **verbal attack** Kharazi accused *the United States* $_{FS}$ of wanting to exercise " world dictatorship " since the " horrible attacks " of September 11 .
3	He said those who thought the election process would be rigged were supporters of the MDC party , adding that they **were prejudging** and wanted to direct the process $_{ZX}$.
4	However , the fact that certain countries have a more balanced view of the conflict $_{ZX}$ is not the only reason to doubt that *anti-Israeli decisions* $_{FS}$ will , in fact , **be adopted** .
5	But *his tough stand on P'yongyang* $_{FS}$ **has provoked concern** in Seoul $_{ZX}$, where President Kim Tae-chung , who is in the last year of his five-year term , has been trying to prise the hermit state out of isolation .

Table 12: The dev examples for which the FS-MTL model *correctly* predicts the *target* in 6/8 trials, whereas the Z&X-STL model predicts *incorrectly* in 6/8 trials.

a nominal predicate and the holder is its object. The sentence is in passive voice but the models probably interpret it in the active voice and thus make the wrong prediction. In the third example, the opinion expression is the head of the relative clause that modifies the holder. These examples raise the following questions: would improved consistency with syntax lead to improvements for ORL and could we train a dependency parsing model with SRL and ORL to help the models handle syntactically harder cases?

Example 4 shows that holders specific to the MPQA annotation schema are hard to label as they require inference skills: from *the department said*, we can defeasibly infer that it is *the department* who *expects [this cost] to stand at just $400/year [...]*. To handle such cases, it would be worth trying training our models jointly with models for recognizing textual entailment.

Examples 6–7 illustrate that some gap in performance stems from difficulties in processing MPQA. Example 5 has no gold holder, but the models make plausible predictions. For example 6, FS-MTL predicts the discontinuous holder *they ... all*, while MPQA allows only contiguous entities. Therefor our evaluation scripts interpret *they* and *all* as two separate holders and deem *all* as incorrect, resulting with lower precision. Finally, for example 7 our models make plausible predictions. However, the gold holder is always the entity from the coreference cluster that is the closest to the opinion.[8] The evaluation scripts needs to be extended such that predicting any entity from the coreference cluster is considered to be correct. To conclude, to better evaluate future developments, it would be worth curating MPQA instances with missing roles and extending evaluation to account for coreferent holders and discontinuous roles.

The examples in Table 10 demonstrate that dif-

[8] We followed the prior work (Katiyar and Cardie, 2016).

ficulties in labeling targets originate from similar reasons as for holders. Examples 1–3 demonstrate complex syntactic constructions, examples 4–6 MPQA-specific annotations that require inference and example 7 exemplifies a missing target.

How does MTL help? There are 18/1055 instances in the dev set for which the FS model predicts the holder correctly and the Z&X-STL model does not, and 19/1055 for targets.

For holders, for 9 out 18 of such examples, the Z&X-STL model does not predict anything (as in examples 2–5 in Table 11). From examples 1–5 we notice that SRL data helps to handle more complex syntactic constructions. From examples 5–7 we observed that using MTL with SRL helps to handle cases when more than one person or organization is present in the close neighborhood of the opinion. For targets, for 11 out of 18 cases the Z&X-STL model does not predict anything as in examples 1–2 in Table 11. We conclude that the greatest improvements from the FS-MTL model comes from having far fewer missing roles.

6 Related work

FGOA. Closest to our work are Yang and Cardie (2013) (Y&C) and Katiyar and Cardie (2016) (K&C). They as well label both holders and targets in MPQA. By contrast, our focus is on the task of ORL. We thus refrain from predicting opinion expressions first, to ensure a reproducible evaluation setup on a fixed set of gold opinion expressions. The MTL models we develop in this work will, however, be the basis for the full task in a later stage. Because of these differences, direct comparison to Y&C and K&C is not possible. However, if we compare our results we notice a big gap that demonstrates that opinion expression extraction is the import step in FGOA. Similar to K&C, Liu et al. (2015) jointly labels opinion expressions and their targets in reviews.

Some work focuses entirely on labeling of opinion expressions (Yang and Cardie, 2014; Irsoy and Cardie, 2014). Other work looks into specific subcategories of ORL: opinion role induction for verbal predicates (Wiegand and Ruppenhofer, 2015), categorization of opinion words into actor and speaker view (Wiegand et al., 2016b), opinion roles extraction on opinion compounds (Wiegand et al., 2016a). Wiegand and Ruppenhofer (2015) report 72.54 binary F1 score for labeling of holders in MPQA (results for targets are not reported).

Neural SRL. New neural SRL models have emerged (He et al., 2017; Yang and Mitchell, 2017; Marcheggiani and Titov, 2017) since we started this work. In future work we can improve our models with such new proposals.

Auxiliary tasks for MTL. Other work investigates under which conditions MTL is effective. Martínez Alonso and Plank (2017) show that the best auxiliary tasks have low kurtosis of labels (usually a small label set) and high entropy (labels occur uniformly). We show that the best MTL model for ORL is the model which uses shared layers only. Thus it seems reasonable to consider only a small and uniform SRL label set {A0, A1}.

Bingel and Søgaard (2017) show that MTL works when the main task has a flattening learning curve, but the auxiliary task curve is still steep. We notice such behavior in our learning curves.

7 Conclusions

We address the problem of scarcity of annotated training data for labeling of opinion holders and targets (ORL) using multi-task learning (MTL) with Semantic Role Labeling (SRL). We adapted a recently proposed neural SRL model for ORL and enhanced it with different MTL techniques. Two MTL models achieve significant improvements with all evaluation measures, for both holders and targets, on both dev and test set, when evaluated with repeated 4-fold CV. We recommend evaluation with comparable dev and test set sizes for future work, as this enables more reliable evaluation.

With deeper analysis we show that future developments should improve the ability of the models to capture long-range dependencies, investigate if consistency with syntax can improve ORL, and consider other auxiliary tasks such as dependency parsing or recognizing textual entailment. We emphasize that future improvements can be measured more reliably if opinion expressions with missing roles are curated and if the evaluation considers all mentions in opinion role coreference chains as well as discontinuous roles.

Acknowledgments

This work has been supported by the German Research Foundation as part of the Research Training Group Adaptive Preparation of Information from Heterogeneous Sources (AIPHES) under grant No. GRK 1994/1.

References

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, Valencia, Spain, pages 164–169. http://www.aclweb.org/anthology/E17-2026.

Anders Björkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*. Association for Computational Linguistics, pages 33–36.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*. Association for Computational Linguistics, Ann Arbor, Michigan, pages 152–164. http://www.aclweb.org/anthology/W/W05/W05-0620.

Eugene Charniak, Don Blaheta, Niyu Ge, Keith Hall, John Hale, and Mark Johnson. 2000. Bllip 1987-89 wsj corpus release 1. *Linguistic Data Consortium, Philadelphia* 36.

Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-criteria learning for chinese word segmentation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1193–1203. http://aclweb.org/anthology/P17-1110.

Yejin Choi, Eric Breck, and Claire Cardie. 2006. Joint extraction of entities and relations for opinion recognition. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Sydney, Australia, pages 431–439. http://www.aclweb.org/anthology/W/W06/W06-1651.

Tao Gui, Qi Zhang, Haoran Huang, Minlong Peng, and Xuanjing Huang. 2017. Part-of-speech tagging for twitter with adversarial neural networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 2401–2410. https://www.aclweb.org/anthology/D17-1255.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 446–456. https://www.aclweb.org/anthology/D17-1046.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 473–483. http://aclweb.org/anthology/P17-1044.

Ozan Irsoy and Claire Cardie. 2014. Opinion mining with deep recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 720–728. http://www.aclweb.org/anthology/D14-1080.

Richard Johansson and Alessandro Moschitti. 2013. Relational Features in Fine-Grained Opinion Analysis. *Computational Linguistics* 39:473–509.

Shafiq Joty, Preslav Nakov, Lluís Màrquez, and Israa Jaradat. 2017. Cross-language Learning with Adversarial Neural Networks. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. Association for Computational Linguistics, Vancouver, Canada, pages 226–237. http://aclweb.org/anthology/K17-1024.

Arzoo Katiyar and Claire Cardie. 2016. Investigating LSTMs for Joint Extraction of Opinion Entities and Relations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany, pages 919–929. http://www.aclweb.org/anthology/P16-1087.

Soo-Min Kim and Eduard Hovy. 2006. Extracting Opinions, Opinion Holders, and Topics Expressed in Online News Media Text. In *Proceedings of the Workshop on Sentiment and Subjectivity in Text*. Sydney, Australia, pages 1–8. http://www.aclweb.org/anthology/W/W06/W06-0301.

Young-Bum Kim, Karl Stratos, and Dongchan Kim. 2017. Adversarial adaptation of synthetic or stale data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1297–1307. http://aclweb.org/anthology/P17-1119.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 2147–2159. https://www.aclweb.org/anthology/D17-1229.

Pengfei Liu, Shafiq Joty, and Helen Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1433–1443. http://aclweb.org/anthology/D15-1168.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1–10. http://aclweb.org/anthology/P17-1001.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 1506–1515. https://www.aclweb.org/anthology/D17-1159.

Héctor Martínez Alonso and Barbara Plank. 2017. When is multitask learning effective? semantic sequence prediction under varying data conditions. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, Valencia, Spain, pages 44–53. http://www.aclweb.org/anthology/E17-1005.

Frank J. Massey Jr. 1951. The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American statistical Association* 46(253):68–78.

Martha Palmer, Paul Kingsbury, and Daniel Gildea. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics* 31:71–106.

Lianhui Qin, Zhisong Zhang, Hai Zhao, Zhiting Hu, and Eric Xing. 2017. Adversarial connective-exploiting networks for implicit discourse relation classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1006–1017. http://aclweb.org/anthology/P17-1093.

Nils Reimers and Iryna Gurevych. 2017. Optimal hyperparameters for deep lstm-networks for sequence labeling tasks. *arXiv preprint arXiv:1707.06799* .

Josef Ruppenhofer, Swapna Somasundaran, and Janyce Wiebe. 2008. Finding the Sources and Targets of Subjective Expressions. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC)*. Marrakech, Morocco, pages 2781–2788.

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

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating Expressions of Opinions and Emotions in Language. *Language Resources and Evaluation* 39:165–210.

Michael Wiegand, Christine Bocionek, and Josef Ruppenhofer. 2016a. Opinion holder and target extraction on opinion compounds – a linguistic approach. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 800–810. http://www.aclweb.org/anthology/N16-1094.

Michael Wiegand and Josef Ruppenhofer. 2015. Opinion holder and target extraction based on the induction of verbal categories. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Beijing, China, pages 215–225. http://www.aclweb.org/anthology/K15-1022.

Michael Wiegand, Marc Schulder, and Josef Ruppenhofer. 2016b. Separating actor-view from speaker-view opinion expressions using linguistic features. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 778–788. http://www.aclweb.org/anthology/N16-1092.

Theresa Ann Wilson. 2008. *Fine-Grained Subjectivity And Sentiment Analysis: Recognizing The Intensity, Polarity, And Attitudes Of Private States*. Ph.D. thesis, University of Pittsburgh.

Yi Wu, David Bamman, and Stuart Russell. 2017. Adversarial training for relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 1779–1784. https://www.aclweb.org/anthology/D17-1187.

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

Bishan Yang and Claire Cardie. 2014. Joint modeling of opinion expression extraction and attribute clas-

sification. *Transactions of the Association for Computational Linguistics* 2:505–516.

Bishan Yang and Tom Mitchell. 2017. A joint sequential and relational model for frame-semantic parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark, pages 1258–1267. https://www.aclweb.org/anthology/D17-1129.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1959–1970. http://aclweb.org/anthology/P17-1179.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China, pages 1127–1137. http://www.aclweb.org/anthology/P15-1109.

Approaching Neural Grammatical Error Correction as a Low-Resource Machine Translation Task

Marcin Junczys-Dowmunt
Microsoft
marcinjd@microsoft.com

Roman Grundkiewicz
University of Edinburgh
rgrundki@inf.ed.ac.uk

Shubha Guha
University of Edinburgh
sguha@ed-alumni.net

Kenneth Heafield
University of Edinburgh
kheafiel@inf.ed.ac.uk

Abstract

Previously, neural methods in grammatical error correction (GEC) did not reach state-of-the-art results compared to phrase-based statistical machine translation (SMT) baselines. We demonstrate parallels between neural GEC and low-resource neural MT and successfully adapt several methods from low-resource MT to neural GEC. We further establish guidelines for trustable results in neural GEC and propose a set of model-independent methods for neural GEC that can be easily applied in most GEC settings. Proposed methods include adding source-side noise, domain-adaptation techniques, a GEC-specific training-objective, transfer learning with monolingual data, and ensembling of independently trained GEC models and language models. The combined effects of these methods result in better than state-of-the-art neural GEC models that outperform previously best neural GEC systems by more than 10% M^2 on the CoNLL-2014 benchmark and 5.9% on the JFLEG test set. Non-neural state-of-the-art systems are outperformed by more than 2% on the CoNLL-2014 benchmark and by 4% on JFLEG.

1 Introduction

Most successful approaches to automated grammatical error correction (GEC) are based on methods from statistical machine translation (SMT), especially the phrase-based variant. For the CoNLL 2014 benchmark on grammatical error correction (Ng et al., 2014), Junczys-Dowmunt and Grundkiewicz (2016) established a set of methods for GEC by SMT that remain state-of-the-art. Systems (Chollampatt and Ng, 2017; Yannakoudakis et al., 2017) that improve on results by Junczys-Dowmunt and Grundkiewicz (2016) use their set-up as a backbone for more complex systems.

The view that GEC can be approached as a machine translation problem by translating from erroneous to correct text originates from Brockett et al. (2006) and resulted in many systems (e.g. Felice et al., 2014; Susanto et al., 2014) that represented the current state-of-the-art at the time.

In the field of machine translation proper, the emergence of neural sequence-to-sequence methods and their impressive results have lead to a paradigm shift away from phrase-based SMT towards neural machine translation (NMT). During WMT 2017 (Bojar et al., 2017) authors of pure phrase-based systems offered "unconditional surrender"[1] to NMT-based methods.

Based on these developments, one would expect to see a rise of state-of-the-art neural methods for GEC, but as Junczys-Dowmunt and Grundkiewicz (2016) already noted, this is not the case. Interestingly, even today, the top systems on established GEC benchmarks are still mostly phrase-based or hybrid systems (Chollampatt and Ng, 2017; Yannakoudakis et al., 2017; Napoles and Callison-Burch, 2017). The best "pure" neural systems (Ji et al., 2017; Sakaguchi et al., 2017; Schmaltz et al., 2017) are several percent behind.[2]

If we look at recent MT work with this in mind, we find one area where phrased-based SMT dominates over NMT: low-resource machine translation. Koehn and Knowles (2017) analyze the behavior of NMT versus SMT for English-Spanish systems trained on 0.4 million to 385.7 million words of parallel data, illustrated in Figure 1. Quality for NMT

[1] Ding et al. (2017) on their news translation shared task poster http://www.cs.jhu.edu/~huda/papers/jhu-wmt-2017.pdf

[2] After submission of this work, Chollampatt and Ng (2018) published impressive new results for neural GEC with some overlap with our methods. However, our results stay ahead on all benchmarks while using simpler models.

Proceedings of NAACL-HLT 2018, pages 595–606
New Orleans, Louisiana, June 1 - 6, 2018.

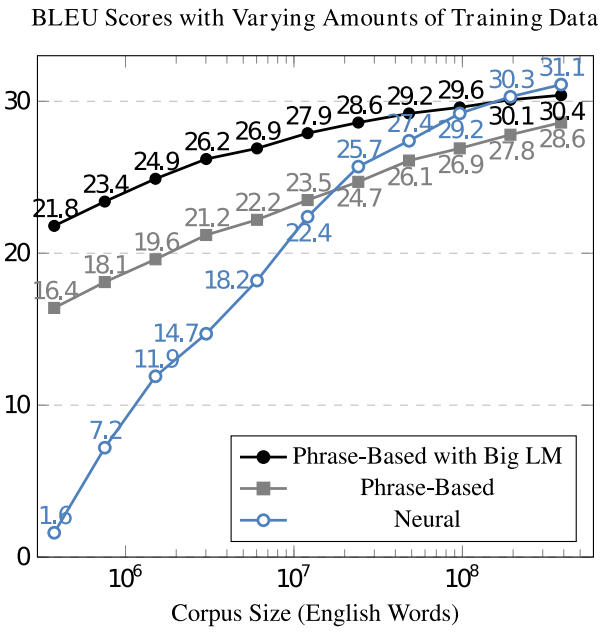


Figure 1: BLEU scores for English-Spanish systems trained on 0.4M to 385.7M words of parallel data. Source: Koehn and Knowles (2017)

Corpus	Sent.	Tokens	Public
NUCLE*	57.1K	1.2M	Yes
Lang-8 NAIST*	1.9M	25.0M	Yes
CLC FCE	30.9K	0.5M	Yes
CLC	1.9M	29.2M	No

Table 1: Statistics for existing GEC training data sets. Data sets marked with * are used in this work.

starts low for small corpora, outperforms SMT at a corpus size of about 15 million words, and with increasing size beats SMT with a large in-domain language model.

Table 1 lists existing training resources for the English as-a-second-language (ESL) grammatical error correction task. Publicly available resources, NUS Corpus of Learner English (NUCLE) by Dahlmeier et al. (2013), Lang-8 NAIST (Mizumoto et al., 2012) and CLC FCE (Yannakoudakis et al., 2011) amount to about 27M tokens. Among these the Lang-8 corpus is quite noisy and of low quality. The Cambridge Learner Corpus (CLC) by Nicholls (2003) — probably the best resource in this list — is non-public and we would strongly discourage reporting results that include it as training data as this makes comparisons difficult.

Contrasting this with Fig. 1, we see that for about 20M tokens NMT systems start outperforming SMT models without additional large language models. Current state-of-the-art GEC systems based on SMT, however, all include large-scale in-domain language models either following the steps outlined in Junczys-Dowmunt and Grundkiewicz (2016) or directly re-using their domain-adapted Common-Crawl language model.

It seems that the current state of neural methods in GEC reflects the behavior for NMT systems trained on smaller data sets. Based on this, we conclude that we can think of GEC as a low-resource, or at most mid-resource, machine translation problem. This means that techniques proposed for low-resource (neural) MT should be applicable to improving neural GEC results.

In this work we show that adapting techniques from low-resource (neural) MT and SMT-based GEC methods allows neural GEC systems to catch up to and outperform SMT-based systems. We improve over the previously best-reported neural GEC system (Ji et al., 2017) on the CoNLL 2014 test set by more than 10% M^2, over a comparable pure SMT system by Junczys-Dowmunt and Grundkiewicz (2016) by 6%, and outperform the state-of-the-art result of Chollampatt and Ng (2017) by 2%. On the JFLEG data set, we report the currently best results, outperforming the previously best pure neural system (Sakaguchi et al., 2017) by 5.9% GLEU and the best reported results (Chollampatt and Ng, 2017) by 3% GLEU.

In Section 2, we describe our NMT-based baseline for GEC, and follow recommendations from the MT community for a trustable neural GEC system. In Section 3, we adapt neural models to make better use of sparse error-annotated data, transferring low-resource MT and GEC-specific SMT methods to neural GEC. This includes a novel training objective for GEC. We investigate how to leverage monolingual data for neural GEC by transfer learning in Section 4 and experiment with language model ensembling in Section 5. Section 6 explores deep NMT architectures. In Section 7, we provide an overview of the experiments and how results relate to the JFLEG benchmark. We also recommend a model-independent toolbox for neural GEC.

2 A trustable baseline for neural GEC

In this section, we combine insights from Junczys-Dowmunt and Grundkiewicz (2016) for grammatical error correction by phrase-based statistical machine translation and from Denkowski and Neubig (2017) for trustable results in neural machine translation to propose a trustable baseline for neural grammatical error correction.

Test/Dev set	Sent.	Annot.	Metric
CoNLL-2013 test	1,381	1	M^2
CoNLL-2014 test	1,312	2	M^2
JFLEG dev	754	4	GLEU
JFLEG test	747	4	GLEU

Table 2: Statistics for test and development data.

2.1 Training and test data

To make our results comparable to state-of-the-art results in the field of GEC, we limit our training data strictly to public resources. In the case of error-annotated data, as marked in Table 1, these are the NUCLE (Dahlmeier et al., 2013) and Lang-8 NAIST (Mizumoto et al., 2012) data sets. We do not include the FCE corpus (Yannakoudakis et al., 2011) to maintain comparability to Junczys-Dowmunt and Grundkiewicz (2016) and Chollampatt and Ng (2017). We strongly urge the community to not use the non-public CLC corpus for training, unless contrastive results without this corpus are provided as well.

We choose the CoNLL-2014 shared task test set (Ng et al., 2014) as our main benchmark and the test set from the 2013 edition of the shared task (Ng et al., 2013) as a development set. For these benchmarks we report MaxMatch (M^2) scores (Dahlmeier and Ng, 2012). Where appropriate, we will provide results on the JFLEG dev and test sets (Napoles et al., 2017) using the GLEU metric (Sakaguchi et al., 2016) to demonstrate the generality of our methods. Table 2 summarizes test/dev set statistics for both tasks.

For most our experiments, we report M^2 on CoNLL-2013 test (Dev) and precision (Prec.), recall (Rec.), M^2 (Test) on the CoNLL-2014 test set.

2.2 Preprocessing and sub-words

As both benchmarks, CoNLL and JFLEG, are provided in NLTK-style tokenization (Bird et al., 2009), we use the same tokenization scheme for our training data. We truecase line beginnings and escape special characters using scripts included with Moses (Koehn et al., 2007). Following Sakaguchi et al. (2017), we apply the Enchant[3] spell-checker to the JFLEG data before evaluation. No spell-checking is used for the CoNLL test sets.

We follow the recommendation by Denkowski and Neubig (2017) to use byte-pair encoding (BPE) sub-word units (Sennrich et al., 2016b) to solve the large-vocabulary problem of NMT. This is a well established procedure in neural machine translation and has been demonstrated to be generally superior to UNK-replacement methods. It has been largely ignored in the field of grammatical error correction even when word segmentation issues have been explored (Ji et al., 2017; Schmaltz et al., 2017). To our knowledge, this is the first work to use BPE sub-words for GEC, however, an analysis on advantages of word versus sub-word or character level segmentation is beyond the scope of this paper. A set of 50,000 monolingual BPE units is trained on the error-annotated data and we segment training and test/dev data accordingly. Segmentation is reversed before evaluation.

2.3 Model and training procedure

Implementations of all models explored in this work[4] are available in the Marian[5] toolkit (Junczys-Dowmunt et al., 2018). The attentional encoder-decoder model in Marian is a re-implementation of the NMT model in Nematus (Sennrich et al., 2017b). The model differs from the model introduced by Bahdanau et al. (2014) by several aspects, the most important being the conditional GRU with attention for which Sennrich et al. (2017b) provide a concise description.

All embedding vectors consist of 512 units; the RNN states of 1024 units. The number of BPE segments determines the size of the vocabulary of our models, i.e. 50,000 entries. Source and target side use the same vocabulary. To avoid overfitting, we use variational dropout (Gal and Ghahramani, 2016) over GRU steps and input embeddings with probability 0.2. We optimize with Adam (Kingma and Ba, 2014) with an average mini-batch size of ca. 200. All models are trained until convergence (early-stopping with a patience of 10 based on development set cross-entropy cost), saving model checkpoints every 10,000 mini-batches. The best eight model checkpoints w.r.t. the development set M^2 score of each training run are averaged element-wise (Junczys-Dowmunt et al., 2016) resulting in a final single model. During decoding we use a beam-size of 24 and normalize model scores by length.[6]

[3] https://github.com/AbiWord/enchant

[4] Models, system configurations and outputs are available from https://github.com/grammatical/neural-naacl2018

[5] https://github.com/marian-nmt/marian

[6] We used a larger beam-size than usual due to experiments with re-ranking of n-best lists not included in the paper. We did not see any differences compared to smaller beams.

	CoNLL				JFLEG	
Run	Dev	Prec.	Rec.	Test	Dev	Test
1	20.2	68.6	11.8	34.9	47.6	52.3
2	21.3	64.6	10.3	31.5	47.1	51.8
3	21.7	64.8	10.6	32.0	47.1	52.4
4	22.0	67.1	10.9	33.0	47.1	52.0
Avg	21.3	–	–	32.9	47.2	52.1
Ens	19.3	70.8	9.5	30.9	47.0	52.5

Table 3: Instable results for multiple baseline runs versus average and ensemble — for the CoNLL benchmark.

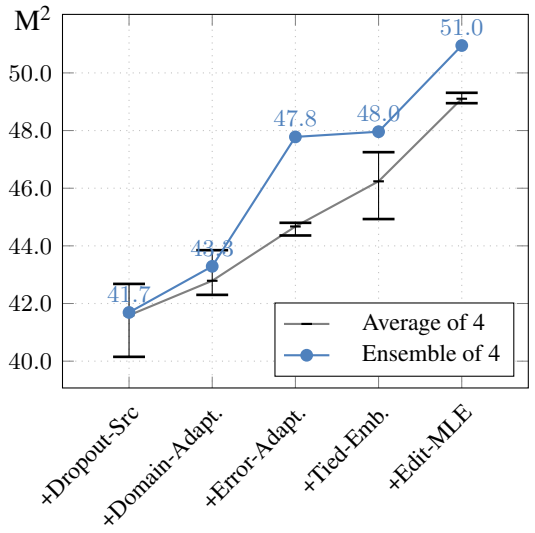
2.4 Optimizer instability

Junczys-Dowmunt and Grundkiewicz (2016) noticed that discriminative parameter tuning for GEC by phrase-based SMT leads to unstable M^2 results between tuning runs. This is a well-known effect for SMT parameter tuning and Clark et al. (2011) recommend reporting results for multiple tuning runs. Junczys-Dowmunt and Grundkiewicz (2016) perform four tuning runs and calculate parameter centroids following Cettolo et al. (2011).

Neural sequence-to-sequence training is discriminative optimization and as such prone to instability. We already try to alleviate this by averaging over eight best checkpoints, but as seen in Table 3, results for M^2 remain unstable for runs with differently initialized weights. An amplitude of 3 points M^2 on the CoNLL-2014 test set is larger than most improvements reported in recent papers. None of the recent works on neural GEC account for instability, hence it is unclear if observed outcomes are actual improvements or lucky picks among by-products of instability. We therefore strongly suggest to provide results for multiple independently trained models. Otherwise improvements of less than 2 or 3 points of M^2 remain doubtful. Interestingly, GLEU on the JFLEG data seems to be more stable than M^2 on CoNLL data.

2.5 Ensembling of independent models

Running multiple experiments to provide averaged results seems prohibitively expensive, but Denkowski and Neubig (2017) and others (e.g. Sutskever et al., 2014; Sennrich et al., 2017a) show that ensembling of independently trained models leads to consistent rewards for MT. For our baseline in Table 3 the opposite seems to be true for M^2. This is likely the reason why no other work on neural GEC mentions results for ensembles.



Model	Dev	Prec.	Rec.	Test
Baseline	19.3	70.8	9.5	30.9
+Dropout-Src.	27.5	72.4	15.5	41.7
+Domain-Adapt.	30.0	69.2	17.3	43.3
+Error-Adapt.	34.5	70.8	20.8	47.8
+Tied-Emb.	33.0	73.0	20.2	48.0
+Edit-MLE	37.6	65.3	27.1	51.0

Table 4: Results (M^2) on the CoNLL benchmark for GEC-specific adaptations.

On closer inspection, however, we see that the drop in M^2 for ensembles is due to a precision bias. M^2 being an F-score penalizes increasing distance between precision and recall. The increase in precision for ensembles is to be expected and we see it later consistently for all experiments. Ensembles choose corrections for which all independent models are fairly confident. This leads to fewer but better corrections, hence an increase in precision and a drop in recall. If the models are weak as our baseline, this can result in a lower score. It would, however, be unwise to dismiss ensembles, as we can use their bias towards precision to our advantage whenever they are combined with methods that aim to increase recall. This is true for nearly all remaining experiments.

3 Adaptations for GEC

The methods described in this section turn our baseline into a more GEC-specific system. Most have been inspired by techniques from low-resource MT or closely related domain-adaptation techniques for NMT. All modifications are applied incrementally, later models include enhancements from the previous ones.

3.1 Source-word dropout as corruption

GEC can be treated as a denoising task where grammatical errors are corruptions that have to be reduced. By introducing more corruption on the source side during training we can teach the model to reduce trust into the source input and to apply corrections more freely. Dropout is one way to introduce noise, but for now we only drop out single units in the embedding or GRU layers, something the model can easily recover from. To make the task harder, we add dropout over source words, setting the full embedding vector for a source word to $1/p_{\text{src}}$ with a probability of p_{src}. During our experiments, we found $p_{\text{src}} = 0.2$ to work best.

Table 4 show impressive gains for this simple method (+Dropout-Src.). Results for the ensemble match the previously best results on the CoNLL-2014 test set for pure neural systems (without the use of an additional monolingual language model) by Ji et al. (2017) and Schmaltz et al. (2017).

3.2 Domain adaptation

The NUCLE corpus matches the domain of the CoNLL benchmarks perfectly. It is however much smaller than the Lang-8 corpus. A setting like this seems to be a good fit for domain-adaptation techniques. Sennrich et al. (2016a) oversample in-domain news data in a larger non-news training corpus. We do the same by adding the NUCLE corpus ten times to the training corpus. This can also be seen as similar to Junczys-Dowmunt and Grundkiewicz (2016) who tune phrase-based SMT parameters on the entire NUCLE corpus. Respectable improvements on both CoNLL test sets (+Domain-Adapt. in Table 4) are achieved.

3.3 Error adaptation

Junczys-Dowmunt and Grundkiewicz (2016) noticed that when tuning on the entire NUCLE corpus, even better results can be achieved if the error rate of NUCLE is adapted to the error rate of the original dev set. In NUCLE only 6% of tokens contain errors, while the CoNLL-2013 test set has an error-rate of about 15%. Following Junczys-Dowmunt and Grundkiewicz (2016), we remove correct sentences from the ten-fold oversampled NUCLE data greedily until an error-rate of 15% is achieved. This can be interpreted as a type of GEC-specific domain adaptation. We mark this method as +Domain-Adapt. in Table 4 and report for the ensemble the so far strongest results for any neural GEC system on the CoNLL benchmark.

	CoNLL				JFLEG	
Λ	Dev	Prec.	Rec.	Test	Dev	Test
1	33.5	67.5	20.8	46.6	48.9	53.9
3	36.8	59.8	28.8	49.2	51.2	56.5
5	36.2	54.0	30.8	47.0	50.9	55.7

Table 5: Results for model type +Tied-Emb. trained with edit-weighted MLE and chosen Λ.

3.4 Tied embeddings

Press and Wolf (2016) showed that parameter tying between input and output embeddings[7] for language models leads to improved perplexity. Similarly, three-way weight-tying between source, target and output embeddings for neural machine translation seems to improve translation quality in terms of BLEU while also significantly decreasing the number of parameters in the model. In monolingual cases like GEC, where source and target vocabularies are (mostly) equal, embedding-tying seems to arise naturally. Output layer, decoder and encoder embeddings all share information which may further enhance the signal from corrective edits. The M^2 scores for +Tied-Emb. in Table 4 are inconclusive, but we see improvements in conjunction with later modifications.

3.5 Edit-weighted MLE objective

Previously, we applied error-rate adaptation to strengthen the signal from corrective edits in the training data. In this section, we investigate the effects of directly modifying the training loss to incorporate weights for corrective edits.

Assuming that each target token y_j has been generated by a source token x_i, we scale the loss for each target token y_j by a factor Λ if y_j differs from x_i, i.e. if y_j is part of an edit. Hence, log-likelihood loss takes the following form:

$$L(x, y, a) = -\sum_{t=1}^{T_y} \lambda(x_{a_t}, y_t) \log P(y_t | x, y_{<t}),$$

$$\lambda(x_{a_t}, y_t) = \begin{cases} \Lambda & \text{if } x_{a_t} \neq y_t \\ 1 & \text{otherwise} \end{cases},$$

where (x, y) is a training sentence pair and a is a word alignment $a_t \in \{0, 1, \ldots, T_x\}$ such that source token x_{a_t} generates target token y_t. Alignments are computed for each sentence pair with fast-align (Dyer et al., 2013).

[7] Output embeddings are encoded in the last output layer of a neural language or translation model.

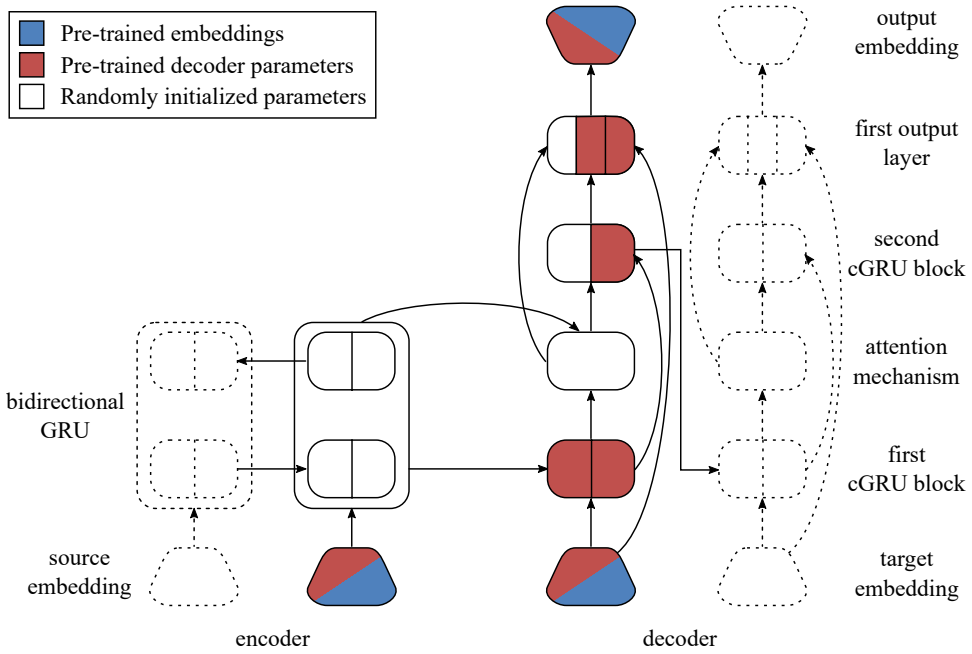


Figure 2: Parameters pretrained on monolingual data are marked with colors. Blue indicates pre-trained embeddings with word2vec, red parameters have been pre-trained with the GRU-based language model only. All embedding layers have tied parameters.

This is comparable to reinforcement learning towards GLEU as introduced by Sakaguchi et al. (2017) or training against diffs by Schmaltz et al. (2017). In combination with previous modifications, edit-weighted Maximum Likelihood Estimation (MLE) weighting seem to outperform both methods. The parameter Λ introduces an additional hyper-parameter that requires tuning for specific tasks and affects the precision/recall trade-off. Table 5 shows $\Lambda = 3$ seems to work best among the tested values when chosen to maximize M^2 on the CoNLL-2013 dev set.

For this setting, we achieve our strongest results of 50.95 M^2 on the CoNLL benchmark (system +Edit-MLE) yet. This outperforms the results of a phrase-based SMT system with a large domain-adapted language model from Junczys-Dowmunt and Grundkiewicz (2016) by 1% M^2 and is the first neural system to beat this strong SMT baseline.

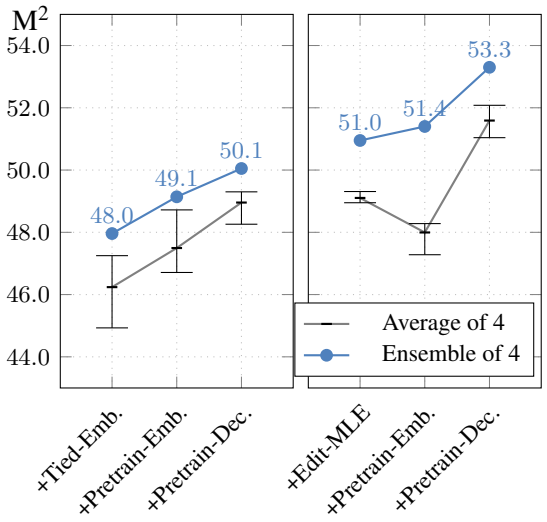
4 Transfer learning for GEC

Many ideas in low-resource neural MT are rooted in transfer learning. In general, one first trains a neural model on high-resource data and then uses the resulting parameters to initialize parameters of a new model meant to be trained on low-resource data only. Various settings are possible, e.g. initializing from models trained on large out-of-domain data and continuing on in-domain data (Miceli Barone et al., 2017) or using related language pairs (Zoph et al., 2016). Models can also be partially initialized by pre-training monolingual language models (Ramachandran et al., 2017) or only word-embeddings (Gangi and Federico, 2017). In GEC, Yannakoudakis et al. (2017) apply pre-trained monolingual word-embeddings as initializations for error-detection models to re-rank SMT n-best lists. Approaches based on pre-training with monolingual data appear to be particularly well-suited to the GEC task. Junczys-Dowmunt and Grundkiewicz (2016) published 300GB of compressed monolingual data used in their work to create a large domain-adapted Common-Crawl n-gram language model.[8] We use the first 100M lines. Preprocessing follows section 2.2 including BPE segmentation.

4.1 Pre-training embeddings

Similarly to Gangi and Federico (2017) or Yannakoudakis et al. (2017), we use Word2vec (Mikolov et al., 2013) with standard settings to create word vectors. Since weights between source, target and output embeddings are tied, these embeddings are inserted once into the model, but affect computations three-fold, see the blue elements in Figure 2. The remaining parameters of the model are initialized randomly. We refer to this adaptation as +Pretrain-Emb.

[8] https://github.com/grammatical/baselines-emnlp2016



Model	Dev	Prec.	Rec.	Test
+Tied-Emb.	33.0	73.0	20.2	48.0
+Pretrain-Emb.	35.5	69.1	22.8	49.1
+Pretrain-Dec.	36.2	69.1	23.8	50.1
+Edit-MLE	37.6	65.3	27.1	51.0
+Pretrain-Emb.	38.2	64.4	28.4	51.4
+Pretrain-Dec.	40.3	65.2	32.2	54.1

Table 6: Results (M^2) on the CoNLL benchmark set for GEC-specific adaptations.

4.2 Pre-training decoder parameters

Following Ramachandran et al. (2017), we first train a GRU-based language model on the monolingual data. The architecture of the language model corresponds as much as possible to the structure of the decoder of the sequence-to-sequence model. All pieces that rely on the attention mechanism or the encoder have been removed. After training for two epochs, all red parameters (including embedding layers) in Figure 2 are copied from the language model to the decoder. Remaining parameters are initialized randomly. This configuration is called +Pretrain-Dec. We pretrain each model separately to make sure that all weights have been initialized randomly.

4.3 Results for transfer learning

Table 6 summarizes the results for our transfer learning experiments. We compare the effects of pre-training with and without the edit-weighted MLE objective and can see that pre-training has significantly positive effects in both settings.

The last result of 53.3% M^2 on the CoNLL-2014 benchmark matches the currently highest reported numbers (53.14% M^2) by Chollampatt and Ng (2017) for a much more complex system and outperforms the highest neural GEC system (Ji et al., 2017) by 8% M^2.

Model	Dev	Prec.	Rec.	Test
+Tied-Emb	33.0	73.0	20.2	48.0
+GRU-LM	40.2	59.8	36.2	52.9
+Edit-MLE	37.6	65.3	27.1	51.0
+GRU-LM	40.3	61.9	34.5	53.4
+Pretrain-Dec.	40.3	65.2	32.2	54.1
+GRU-LM	41.6	62.2	36.6	54.6

Table 7: Ensembling with a neural language model.

5 Ensembling with language models

Phrase-based SMT systems benefit naturally from large monolingual language models, also in the case of GEC as shown by Junczys-Dowmunt and Grundkiewicz (2016). Previous work (Xie et al., 2016; Ji et al., 2017) on neural GEC used n-gram language models to incorporate monolingual data. Xie et al. (2016) built a large 5-gram model and integrated it directly into their beam search algorithm, while Ji et al. (2017) re-use the language model provided by Junczys-Dowmunt and Grundkiewicz (2016) for n-best list re-ranking.

We already combined monolingual data with our GEC models via pre-training, but exploiting separate language models is attractive as no additional training is required. Here, we reuse the neural language model created for pre-training.

Similarly to Xie et al. (2016), the score $s(y|x)$ for a correction y of sentence x is calculated as

$$s(y|x) = \frac{1}{|y|}\left[\sum_{i=1}^{4} \log P_i(y|x) + \alpha \log P_{\text{LM}}(y)\right],$$

where $P_i(y|x)$ is a translation probability for the i-th model in an ensemble of 4. $P_{\text{LM}}(y)$ is the language model probability for y weighted by α. We normalize by sentence length $|y|$. Using the dev set, we choose α that maximizes this score via linear search in range $[0, 2]$ with step 0.1.

Table 7 summarizes results for language model ensembling with three of our intermediate configurations. All configurations benefit from the language model in the ensemble, although gains for the pre-trained model are rather small.

6 Deeper NMT models

So far we analyzed model-independent[9] methods — only training data, hyper-parameters, parameter initialization, and the objective function were modified. In this section we investigate if these techniques can be generalized to deeper or different architectures.

6.1 Architectures

We consider two state-of-the-art NMT architectures implemented in Marian:

Deep RNN A deep RNN-based model (Miceli Barone et al., 2017) proposed by Sennrich et al. (2017a) for their WMT 2017 submissions. This model is based on the shallow model we used until now. It has single layer RNNs in the encoder and decoder, but increases depth by stacking multiple GRU-style blocks inside one RNN cell. A single RNN step passes through all blocks before recursion. The encoder RNN contains 4 stacked GRU blocks, the decoder 8 (1 + 7 due to the conditional GRU). Following Sennrich et al. (2017a), we enable layer-normalization in the RNN-layers. State and embedding dimensions used throughout this work and in Sennrich et al. (2017a) are the same.

Transformer The self-attention-based model by Vaswani et al. (2017). We base our model on their default architecture of 6 complex attention/self-attention blocks in the encoder and decoder and use the same model dimensions — embeddings vector size is 512 (as before), filter size is 2048.

6.2 Training settings

As the deep models are less reliably trained with asynchronous SGD, we change the training algorithm to synchronous SGD and for both models follow the recipe proposed in Vaswani et al. (2017), with an effective base learning rate of 0.0003, learning rate warm-up during the first 16,000 iterations, and an inverse square-root decay after the warm-up. As before, we average the best 8 checkpoints. We increase dropout probability over RNN layers to 0.3 for Deep-RNN and similarly set dropout between transformer layers to 0.3. Source-word dropout as a noising technique remains unchanged.

[9]The pre-training procedure however needs to be adapted to model architecture if we want to take advantage of every shared parameter, otherwise matching parameter subsets could probably be used successfully.

Model	Dev	Prec.	Rec.	Test
+Pretrain-Dec.	40.3	65.2	32.2	54.1
+GRU-LM	41.6	62.2	36.6	54.6
+Deep-RNN	41.1	64.3	35.2	55.2
+Deep-RNN-LM	41.9	61.3	40.2	55.5
+Transformer	41.5	63.0	38.9	56.1
+Transformer-LM	42.9	61.9	40.2	55.8

Table 8: Shallow (Pretrain-Dec.) versus deep ensembles, with and without corresponding language models.

6.3 Pre-training deep models

We reuse all methods included up to +Pretrain-Dec. The pre-training procedure as described in section 4.1 needs to be modified in order to maximize the number of pre-trained parameters for the larger model architectures. Again, we train decoder-only models as typical language models by removing all elements that depend on the encoder, including attention-mechanisms over the source context. We can keep the decoder self-attention layers in the transformer model. We train for two epochs on our monolingual data reusing the hyper-parameters for the parallel case above.

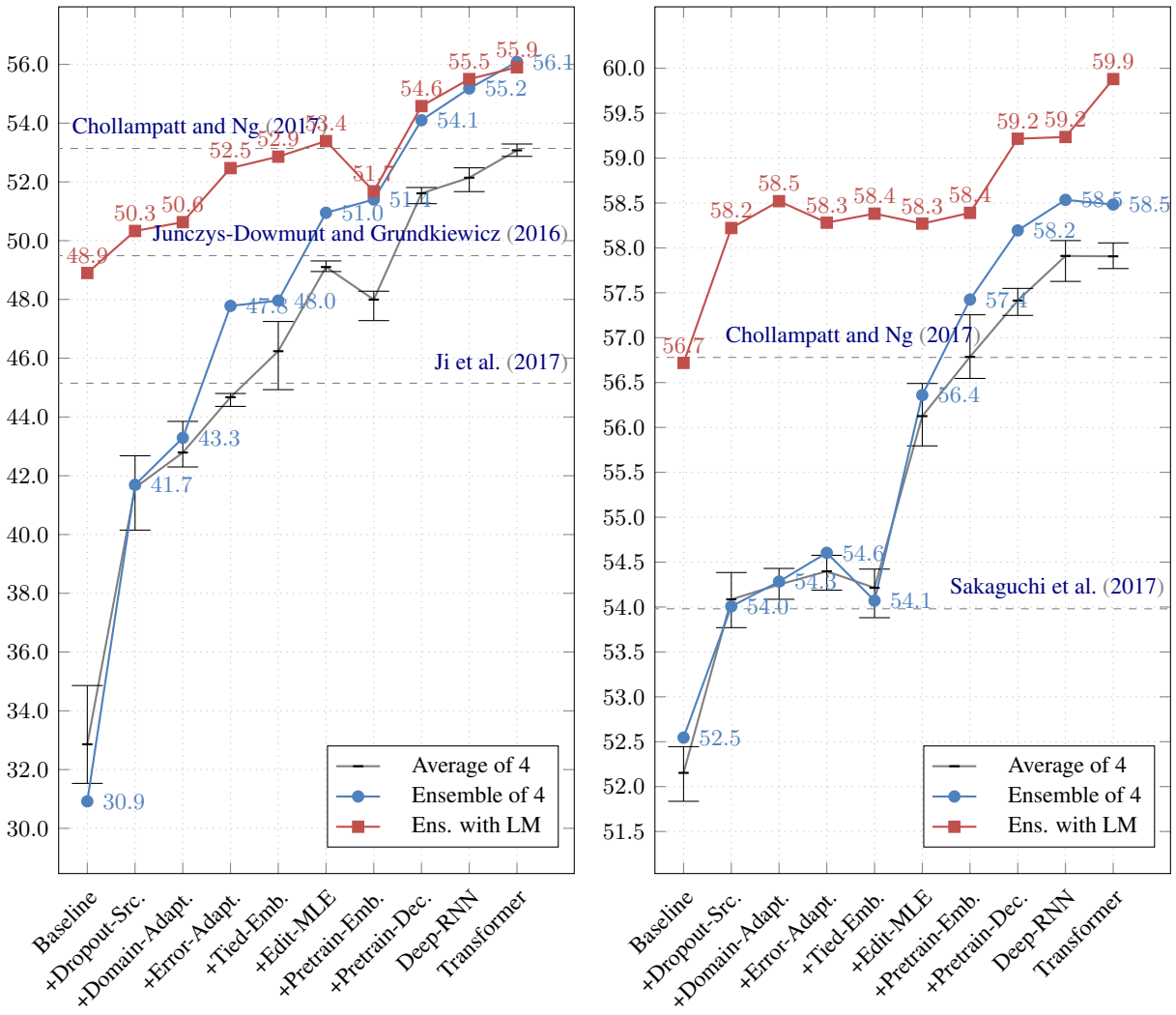
6.4 Results

Table 8 summarizes the results for deeper models on the CoNLL dev and test set. Both deep models improve significantly over the shallow model with the transformer model reaching our best result reported on the CoNLL 2014 test set. For that test set it seems that ensembling with language models that were used for pre-training is ineffective when measured with M^2; while on the JFLEG data measured with GLEU we see strong improvements (Fig. 3b).

7 A standard tool set for neural GEC

We summarize the results for our experiments in Figure 3 and provide results on the JFLEG test set. Weights for the independent language model in the full ensemble were chosen on the respective dev sets for both tasks. Comparing results according to both benchmarks and evaluation metrics (M^2 for CoNLL, GLEU for JFLEG), it seems we can isolate the following set of reliable methods for state-of-the-art neural grammatical error correction:

- Ensembling neural GEC models with monolingual language models;
- Dropping out entire source embeddings;



(a) CoNLL-2014 test set (M^2)

(b) JFLEG test set (GLEU)

Figure 3: Comparison on the CoNLL-2014 test set and JFLEG test for all investigated methods.

- Weighting edits in the training objective during optimization (+Edit-MLE);
- Pre-training on monolingual data;
- Ensembling of independently trained models;
- Domain and error adaptation (+Domain-Adapt., Error-Adapt.) towards a specific benchmark;
- Increasing model depth.

Combinations of these generally[10] model-independent methods helped raising the performance of pure neural GEC systems by more than 10% M^2 on the CoNLL 2014 benchmark, also outperforming the previous state-of-the-art (Chollampatt and Ng, 2017), a hybrid phrase-based system with a complex spell-checking system by 2%. We also showed that a pure neural system can easily outperform a strong pure phrase-based SMT system (Junczys-Dowmunt and Grundkiewicz, 2016) when similarly adapted to the GEC task.

On the JFLEG benchmark we outperform the previously-best pure neural system (Sakaguchi et al., 2017) by 5.9% GLEU (4.5% if no monolingual data is used). Improvements over SMT-based system like Napoles and Callison-Burch (2017)[11] and Chollampatt and Ng (2017) are significant and constitute the new state-of-the-art on the JFLEG test set.

Acknowledgments

This work was partially funded by Facebook. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Facebook.

[10] Increasing depth or changing the architecture to the Transformer model is clearly not model-independent.

[11] Results based on errata from https://github.com/cnap/smt-for-gec#errata

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *The 3rd International Conference on Learning Representations (ICLR2015)*.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition.

Ondrej Bojar, Christian Buck, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Julia Kreutzer, Varvara Logacheva, Christof Monz, Matteo Negri, Aurélie Névéol, Mariana Neves, Matt Post, Stefan Riezler, Artem Sokolov, Lucia Specia, Marco Turchi, and Karin Verspoor, editors. 2017. *Proceedings of the Second Conference on Machine Translation*. Association for Computational Linguistics, Copenhagen, Denmark. http://www.aclweb.org/anthology/W17-47.

Chris Brockett, William B. Dolan, and Michael Gamon. 2006. Correcting ESL errors using phrasal SMT techniques. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, USA, pages 249–256.

Mauro Cettolo, Nicola Bertoldi, and Marcello Federico. 2011. Methods for smoothing the optimizer instability in SMT. In *MT Summit XIII: the Thirteenth Machine Translation Summit*. pages 32–39.

Shamil Chollampatt and Hwee Tou Ng. 2017. Connecting the dots: Towards human-level grammatical error correction. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*. Association for Computational Linguistics, Copenhagen, Denmark, pages 327–333. http://www.aclweb.org/anthology/W17-5037.

Shamil Chollampatt and Hwee Tou Ng. 2018. A multilayer convolutional encoder-decoder neural network for grammatical error correction. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA - Short Papers*. pages 176–181. http://www.aclweb.org/anthology/P11-2031.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 568–572.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner english: The NUS corpus of learner english. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*. pages 22–31.

Michael Denkowski and Graham Neubig. 2017. Stronger baselines for trustable results in neural machine translation. In *The First Workshop on Neural Machine Translation (NMT)*. Vancouver, Canada. http://www.phontron.com/paper/denkowski17wnmt.pdf.

Shuoyang Ding, Huda Khayrallah, Philipp Koehn, Matt Post, Gaurav Kumar, and Kevin Duh. 2017. The JHU machine translation systems for WMT 2017. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*. Association for Computational Linguistics, Copenhagen, Denmark, pages 276–282. http://www.aclweb.org/anthology/W17-4724.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *HLT-NAACL*. The Association for Computational Linguistics, pages 644–648.

Mariano Felice, Zheng Yuan, Øistein E. Andersen, Helen Yannakoudakis, and Ekaterina Kochmar. 2014. Grammatical error correction using hybrid systems and type filtering. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*. Association for Computational Linguistics, Baltimore, Maryland, pages 15–24. http://www.aclweb.org/anthology/W14-1702.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, Curran Associates, Inc., pages 1019–1027.

Mattia Antonino Di Gangi and Marcello Federico. 2017. Can monolingual embeddings improve neural machine translation? In *Proceedings of the Fourth Italian Conference on Computational Linguistics (CLiC-it 2017), Rome, Italy, December 11-13, 2017.*. http://ceur-ws.org/Vol-2006/paper040.pdf.

Jianshu Ji, Qinlong Wang, Kristina Toutanova, Yongen Gong, Steven Truong, and Jianfeng Gao. 2017. A nested attention neural hybrid model for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 753–762.

Marcin Junczys-Dowmunt, Tomasz Dwojak, and Hieu Hoang. 2016. Is neural machine translation ready for deployment? a case study on 30 translation directions. In *Proceedings of the 9th International Workshop on Spoken Language Translation (IWSLT)*. Seattle, WA. http://workshop2016.iwslt.org/downloads/IWSLT_2016_paper_4.pdf.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Phrase-based machine translation is state-of-the-art for automatic grammatical error correction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1546–1556. https://aclweb.org/anthology/D16-1161.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in C++. *arXiv preprint arXiv:1804.00344* https://arxiv.org/abs/1804.00344.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *Proceedings of the 3rd International Conference on Learning Representations (ICLR)* .

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Annual Meeting of the Association for Computational Linguistics*. The Association for Computer Linguistics.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*. Association for Computational Linguistics, Vancouver, pages 28–39.

Antonio Valerio Miceli Barone, Barry Haddow, Ulrich Germann, and Rico Sennrich. 2017. Regularization techniques for fine-tuning in neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 1489–1494. https://www.aclweb.org/anthology/D17-1156.

Antonio Valerio Miceli Barone, Jindřich Helcl, Rico Sennrich, Barry Haddow, and Alexandra Birch. 2017. Deep architectures for neural machine translation. In *Proceedings of the Second Conference on Machine Translation, Volume 1: Research Papers*. Association for Computational Linguistics, Copenhagen, Denmark. http://www.statmt.org/wmt17/pdf/WMT10.pdf.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781.

Tomoya Mizumoto, Yuta Hayashibe, Mamoru Komachi, Masaaki Nagata, and Yu Matsumoto. 2012. The effect of learner corpus size in grammatical error correction of ESL writings. In *Proceedings of COLING 2012*. pages 863–872.

Courtney Napoles and Chris Callison-Burch. 2017. Systematically adapting machine translation for grammatical error correction. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*. Association for Computational Linguistics, Copenhagen, Denmark, pages 345–356. http://www.aclweb.org/anthology/W17-5039.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. JFLEG: A fluency corpus and benchmark for grammatical error correction. In *Proceedings of the 2017 Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Valencia, Spain. https://arxiv.org/abs/1702.04066.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*. Association for Computational Linguistics, Baltimore, Maryland, pages 1–14. http://www.aclweb.org/anthology/W14-1701.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*. Association for Computational Linguistics, Sofia, Bulgaria, pages 1–12. http://www.aclweb.org/anthology/W13-3601.

Diane Nicholls. 2003. The Cambridge Learner Corpus: Error coding and analysis for lexicography and ELT. In *Proceedings of the Corpus Linguistics 2003 conference*. volume 16, pages 572–581.

Ofir Press and Lior Wolf. 2016. Using the output embedding to improve language models. *CoRR* abs/1608.05859.

Prajit Ramachandran, Peter Liu, and Quoc Le. 2017. Unsupervised pretraining for sequence to sequence learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 383–391. https://www.aclweb.org/anthology/D17-1039.

Keisuke Sakaguchi, Courtney Napoles, Matt Post, and Joel Tetreault. 2016. Reassessing the goals of grammatical error correction: Fluency instead of grammaticality. *Transactions of the Association for Computational Linguistics* 4:169–182. https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/800.

Keisuke Sakaguchi, Matt Post, and Benjamin Van Durme. 2017. Grammatical error correction with neural reinforcement learning. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Asian Federation of Natural Language Processing, Taipei, Taiwan, pages 366–372. http://www.aclweb.org/anthology/I17-2062.

Allen Schmaltz, Yoon Kim, Alexander Rush, and Stuart Shieber. 2017. Adapting sequence models for sentence correction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 2807–2813. https://www.aclweb.org/anthology/D17-1298.

Rico Sennrich, Alexandra Birch, Anna Currey, Ulrich Germann, Barry Haddow, Kenneth Heafield, Antonio Valerio Miceli Barone, and Philip Williams. 2017a. The University of Edinburgh's neural MT systems for WMT17. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*. Association for Computational Linguistics, Copenhagen, Denmark, pages 389–399. http://www.aclweb.org/anthology/W17-4739.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. 2017b. Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Valencia, Spain, pages 65–68. http://aclweb.org/anthology/E17-3017.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Edinburgh neural machine translation systems for WMT16. In *Proceedings of the First Conference on Machine Translation*. Association for Computational Linguistics, Berlin, Germany, pages 371–376. http://www.aclweb.org/anthology/W/W16/W16-2323.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1715–1725. http://www.aclweb.org/anthology/P16-1162.

Raymond Hendy Susanto, Peter Phandi, and Hwee Tou Ng. 2014. System combination for grammatical error correction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 951–962. https://doi.org/10.3115/v1/D14-1102.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. MIT Press, Cambridge, MA, USA, NIPS'14, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, Curran Associates, Inc., pages 5998–6008. http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, and Andrew Y Ng. 2016. Neural language correction with character-based attention. *arXiv preprint arXiv:1603.09727* .

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading ESOL texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pages 180–189.

Helen Yannakoudakis, Marek Rei, Øistein E. Andersen, and Zheng Yuan. 2017. Neural sequence-labelling models for grammatical error correction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 2785–2796. https://www.aclweb.org/anthology/D17-1296.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *EMNLP*. The Association for Computational Linguistics, pages 1568–1575.

Robust Cross-lingual Hypernymy Detection using Dependency Context

Shyam Upadhyay[1*], **Yogarshi Vyas**[2*], **Marine Carpuat**[2], **Dan Roth**[1]
[1] Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA
[2] Department of Computer Science, University of Maryland, College Park, MD
shyamupa@seas.upenn.edu, yogarshi@cs.umd.edu,
marine@cs.umd.edu, danroth@seas.upenn.edu

Abstract

Cross-lingual Hypernymy Detection involves determining if a word in one language ("fruit") is a hypernym of a word in another language ("pomme" i.e. apple in French). The ability to detect hypernymy cross-lingually can aid in solving cross-lingual versions of tasks such as textual entailment and event coreference. We propose BISPARSE-DEP, a family of unsupervised approaches for cross-lingual hypernymy detection, which learns sparse, bilingual word embeddings based on dependency contexts. We show that BISPARSE-DEP can significantly improve performance on this task, compared to approaches based only on lexical context. Our approach is also robust, showing promise for low-resource settings: our dependency-based embeddings can be learned using a parser trained on related languages, with negligible loss in performance. We also crowd-source a challenging dataset for this task on four languages – Russian, French, Arabic, and Chinese. Our embeddings and datasets are publicly available.[1]

1 Introduction

Translation helps identify correspondences in bilingual texts, but other asymmetric semantic relationships can improve language understanding when translations are not exactly equivalent. One such relationship is *cross-lingual hypernymy* – identifying that *écureuil* ("squirrel" in French) is a kind of *rodent*, or ворона ("crow" in Russian) is a kind of *bird*. The ability to detect hypernyms across languages serves as a building block in a range of cross-lingual tasks, including Recognizing Textual Entailment (RTE) (Negri et al., 2012, 2013), constructing multilingual taxonomies (Fu et al., 2014), event coreference across multilingual news sources (Vossen et al., 2015), and evaluating Machine Translation output (Padó et al., 2009).

Building models that can robustly identify hypernymy across the spectrum of human languages is a challenging problem, that is further compounded in low resource settings. At first glance, translating words to English and then identifying hypernyms in a monolingual setting may appear to be a sufficient solution. However, this approach cannot capture many phenomena. For instance, the English words *cook*, *leader* and *supervisor* can all be hypernyms of the French word *chef*, as the French word does not have a exact translation in English covering its possible usages. However, translating *chef* to *cook* and then determining hypernymy monolingually precludes identifying *leader* or *supervisor* as a hypernyms of *chef*. Similarly, language-specific usage patterns can also influence hypernymy decisions. For instance, the French word *chroniqueur* translates to *chronicler* in English, but is more frequently used in French to refer to journalists (making *journalist* its hypernym).[2]

This motivates approaches that *directly* detect hypernymy in the cross-lingual setting by extending distributional methods for detecting monolingual hypernymy, as in our prior work (Vyas and Carpuat, 2016). State-of-the-art distributional approaches (Roller and Erk, 2016; Shwartz et al., 2017) for detecting monolingual hypernymy require syntactic analysis (eg. dependency parsing), which may not available for many languages. Additionally, limited training resources make unsupervised methods more desirable than supervised hypernymy detection approaches (Roller and Erk,

[*] These authors contributed equally.
[1] https://github.com/yogarshi/bisparse-dep/
[2] All examples are from our dataset.

Proceedings of NAACL-HLT 2018, pages 607–618
New Orleans, Louisiana, June 1 - 6, 2018.

2016). Furthermore, monolingual distributional approaches cannot be applied directly to the cross-lingual task, because the vector spaces of two languages need to be aligned using a cross-lingual resource (a bilingual dictionary, for instance).

We tackle these challenges by proposing BISPARSE-DEP - a family of robust, unsupervised approaches for identifying cross-lingual hypernymy. BISPARSE-DEP uses a cross-lingual word embedding model learned from a small bilingual dictionary and a variety of monolingual syntactic context extracted from a dependency parsed corpus. BISPARSE-DEP exhibits robust behavior along multiple dimensions. In the absence of a dependency treebank for a language, it can learn embeddings using a parser trained on related languages. When exposed to less monolingual data, or a lower quality bilingual dictionary, BISPARSE-DEP degrades only marginally. In all these cases, it compares favorably with models that have been supplied with all necessary resources, showing promise for low-resource settings. We extensively evaluate BISPARSE-DEP on a new crowd-sourced cross-lingual dataset, with over 2900 hypernym pairs, spanning four languages from distinct families – French, Russian, Arabic and Chinese – and release the datasets for future evaluations.

2 Related Work

Cross-lingual Distributional Semantics Cross-lingual word embeddings have been shown to encode semantics across languages in tasks such as word similarity (Faruqui and Dyer, 2014) and lexicon induction (Vulić and Moens, 2015). Our works stands apart in two aspects (1) In contrast to tasks involving similarity and synonymy (symmetric relations), the focus of our work is on detecting *asymmetric* relations across languages, using cross-lingual embeddings. (2) Unlike most previous work, we use dependency context instead of lexical context to induce cross-lingual embeddings, which allows us to abstract away from language specific word order, and (as we show) improves hypernymy detection.

More closely related is our prior work (Vyas and Carpuat, 2016) where we used lexical context based embeddings to detect cross-lingual lexical entailment. In contrast, the focus of this work is on hypernymy, a more well-defined relation than entailment. Also, we improve upon our previous approach by using dependency based embeddings (§6.1), and show that the improvements hold even when exposed to data scarce settings (§6.3).

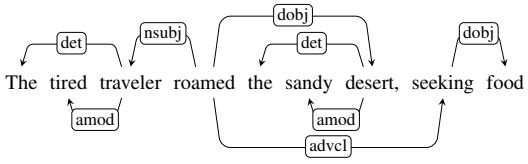


Figure 2: Example Dependency Tree.

We also do a more comprehensive evaluation on four languages paired with English, instead of just French.

Dependency Based Embeddings In monolingual settings, dependency based embeddings have been shown to outperform window based embeddings on many tasks (Bansal et al., 2014; Hill et al., 2014; Melamud et al., 2016). Roller and Erk (2016) showed that dependency embeddings can help in recovering Hearst patterns (Hearst, 1992) like "animals such as cats", which are known to be indicative of hypernymy. Shwartz et al. (2017) demonstrated that dependency based embeddings are almost always superior to window based embeddings for identifying hypernyms in English. Our work uses dependency based embeddings in a cross-lingual setting, a less explored research direction. A key novelty of our work also lies in its use of syntactic transfer to derive dependency contexts. This scenario is more relevant in a cross-lingual setting, where treebanks might not be available for many languages.

3 Our Approach – BISPARSE-DEP

We propose BISPARSE-DEP, a family of approaches that uses sparse, bilingual, dependency based word embeddings to identify cross-lingual hypernymy.

Figure 1 shows an overview of the end-to-end pipeline of BISPARSE-DEP. The two key components of this pipeline are: (1) *Dependency based contexts* (§3.1), which help us generalize across languages with minimal customization by abstracting away language-specific word order. We also discuss how to extract such contexts in the absence of a treebank in the language (§3.2) using a (weak) dependency parser trained on related languages. (2) *Bilingual sparse coding* (§3.3), which allows us to align dependency based word embeddings in a shared semantic space using a small bilingual dictionary. The resulting sparse bilingual embeddings can then be used with a unsupervised entailment scorer (§3.4) to predict hypernymy for cross-lingual word pairs.

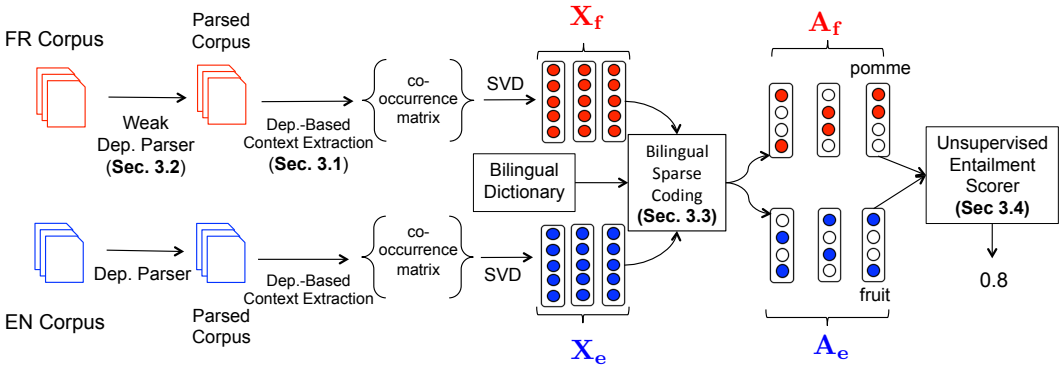


Figure 1: The BISPARSE-DEP approach, which learns sparse bilingual embeddings using dependency based contexts. The resulting sparse embeddings, together with an unsupervised entailment scorer, can detect hypernyms across languages (e.g., *pomme* is a *fruit*).

3.1 Dependency Based Context Extraction

The context of a word can be described in multiple ways using its syntactic neighborhood in a dependency graph. For instance, in Figure 2, we describe the context for a target word (*traveler*) in the following two ways:

- FULL context (Padó and Lapata, 2007; Baroni and Lenci, 2010; Levy and Goldberg, 2014): Children and parent words, concatenated with the label and direction of the relation (eg. *roamed#nsubj*$^{-1}$ and *tired#amod* are contexts for *traveler*).
- JOINT context (Chersoni et al., 2016): Parent concatenated with each of its siblings (eg. *roamed#desert* and *roamed#seeking* are contexts for *traveler*).

These two contexts exploit different amounts of syntactic information – JOINT does not require labeled parses, unlike FULL. The JOINT context combines parent and sibling information, while FULL keeps them as distinct contexts. Both encode directionality into the context, either through label direction or through sibling-parent relations.

We use word-context co-occurrences generated using these contexts in a distributional semantic model (DSM) in lieu of window based contexts to generate dependency based embeddings.

3.2 Dependency Contexts without a Treebank

Using dependency contexts in multilingual settings may not always be possible, as dependency treebanks are not available for many languages. To circumvent this issue, we use related languages to train a weak dependency parser.

We train a *delexicalized* parser using treebanks of related languages, where the word form based features are turned off, so that the parser is trained on purely non-lexical features (e.g. POS tags). The rationale behind this is that related languages show common syntactic structure that can be transferred to the original language, with delexicalized parsing (Zeman and Resnik, 2008; McDonald et al., 2011, *inter alia)* being one popular approach.[3]

3.3 Bilingual Sparse Coding

Given a dependency based co-occurrence matrix described in the previous section(s), we generate BISPARSE-DEP embeddings using the framework from our prior work (Vyas and Carpuat, 2016), which we henceforth call BISPARSE. BISPARSE generates sparse, bilingual word embeddings using a dictionary learning objective with a sparsity inducing l_1 penalty. We give a brief overview of this approach, the full details of which can be found in our prior work.

For two languages with vocabularies v_e and v_f, and monolingual dependency embeddings $\mathbf{X_e}$ and $\mathbf{X_f}$, BISPARSE solves the following objective:

$$\begin{aligned}\underset{\mathbf{A_e},\mathbf{D_e},\mathbf{A_f},\mathbf{D_f}}{\text{argmin}} & \sum_{i=1}^{v_e} \frac{1}{2}||\mathbf{A_e}_i\mathbf{D_e}^{\mathrm{T}} - \mathbf{X_e}_i||_2^2 + \lambda_e||\mathbf{A_e}_i||_1 \\ & + \sum_{j=1}^{v_f} \frac{1}{2}||\mathbf{A_f}_j\mathbf{D_f}^{\mathrm{T}} - \mathbf{X_f}_j||_2^2 + \lambda_f||\mathbf{A_f}_j||_1 \\ & + \sum_{i,j} \frac{1}{2}\lambda_x \mathbf{S}_{ij}||\mathbf{A_e}_i - \mathbf{A_f}_j||_2^2 \qquad (1) \\ \text{s.t. } \mathbf{A_k} > \mathbf{0} \qquad & \|\mathbf{D_k}_i\|_2^2 \leq 1 \qquad \mathbf{k} \in \{\mathbf{e}, \mathbf{f}\}\end{aligned}$$

where $\mathbf{S}$ is a translation matrix, and $\mathbf{A_e}$ and $\mathbf{A_f}$

[3]More sophisticated techniques for transferring syntactic knowledge have been proposed (Ammar et al., 2016; Rasooli and Collins, 2017), but we prioritize simplicity and show that a simple delexicalized parser is effective.

are sparse matrices which are bilingual representations in a shared semantic space. The translation matrix $\mathbf{S}$ (of size $v_e \times v_f$) captures correspondences between the vocabularies (of size v_e and v_f) of two languages. For instance, each row of $\mathbf{S}$ can be a one-hot vector that identifies the word in ***f*** that is most frequently aligned with the ***e*** word for that row in a large parallel corpus, thus building a one-to-many mapping between the two languages.

3.4 Unsupervised Entailment Scorer

A variety of scorers can be used to quantify the directional relationship between two words, given feature representations of these words (Lin, 1998; Weeds and Weir, 2003; Lenci and Benotto, 2012). Once the BISPARSE-DEP embeddings are constructed, we use *BalAPinc* (Kotlerman et al., 2009) to score word pairs for hypernymy. *BalAPinc* is based on the distributional inclusion hypothesis (Geffet and Dagan, 2005) and computes the geometric mean of 1) *LIN* (Lin, 1998), a symmetric score that captures similarity, and 2) *APinc*, an asymmetric score based on average precision.

4 Crowd-Sourcing Annotations

There is no publicly available dataset to evaluate models of hypernymy detection across multiple languages. While ontologies like Open Multilingual WordNet (OMW) (Bond and Foster, 2013) and BabelNet (Navigli and Ponzetto, 2012) contain cross-lingual links, these resources are semi-automatically generated and hence contain noisy edges. Thus, to get reliable and high-quality test beds, we collect evaluation datasets using CrowdFlower[4]. Our datasets span four languages from distinct families - French (Fr), Russian (Ru), Arabic (Ar) and Chinese (Zh) - paired with English.

To begin the annotation process, we first pool candidate pairs using hypernymy edges across languages from OMW and BabelNet, along with translations from monolingual hypernymy datasets (Baroni and Lenci, 2011; Baroni et al., 2012; Kotlerman et al., 2010).

4.1 Annotation Setup

The annotation task requires annotators to be fluent in both English and the non-English language. To ensure only fluent speakers perform the task, for each language, we provide task instructions in the non-English language itself. Also, we restrict the task to annotators verified by CrowdFlower to have those language skills. Finally, annotators also need to pass a quiz based on a small amount of gold standard data to gain access to the task.

[4] http://crowdflower.com

pair	#crowdsourced	#pos (= #neg)
French-English	2115	763
Russian-English	2264	706
Arabic-English	2144	691
Chinese-English	2165	806

Table 1: Crowd-sourced dataset statistics. #pos (#neg) denote positives (negatives) in the evaluation set. We deliberately under-sample negatives to have a balanced evaluation set.

Annotators choose between three options for each word pair (p_f, q_e), where p_f is a non-English word and q_e is a English word : "p_f is a kind of q_e", "q_e is a part of p_f" and "none of the above". Word pairs labeled with the first option are considered as positive examples while those labeled as "none of the above" are considered as negative.[5] The second option was included to filter out meronymy examples that were part of the noisy pool. We leave it to the annotator to infer whether the relation holds between any senses of p_f or q_e, if either of them are polysemous.

For every candidate hypernym pair (p_f, q_e), we also ask annotators to judge its reversed and translated *hyponym* pair (q_f, p_e). For instance, if $(citron, food)$ is a hypernym candidate, we also show annotators $(aliments, lemon)$ which is a potential hyponym candidate (*potential*, because as mentioned in §1, translation need not preserve semantic relationships). The purpose of presenting the hyponym pair, (q_f, p_e), is two-fold. First, it emphasizes the directional nature of the task. Second, it identifies hyponym pairs, which we use as negative examples. The hyponym pairs are challenging since differentiating them from hypernyms truly requires detecting asymmetry.

Each pair was judged by at least 5 annotators, and judgments with 80% agreement (at least 4 annotators agree) are considered for the final dataset. This is a stricter condition than certain monolingual hypernymy datasets - for instance, EVALution (Santus et al., 2015) - where agreement by 3 annotators is deemed sufficient. Inter-annotator agreement measured using Fleiss' Kappa (Fleiss, 1971) was 58.1 (French), 53.7 (Russian), 53.2 (Arabic) and 55.8 (Chinese). This indicates moderate agreement, on par with agreement obtained on related fine-grained semantic tasks (Pavlick et al., 2015). We cannot compare with monolin-

[5] We collected more negative pairs than positive, but sampled so as to keep a balanced dataset for ease of evaluation. We will release all annotated pairs along with the dataset.

gual hypernymy annotator agreement as, to the best of our knowledge, such numbers are not available for existing test sets. Dataset statistics are shown in Table 1.

We observed that annotators were able to agree on pairs containing polysemous words where hypernymy holds for some sense. For instance, for the French-English pair (*avocat*, *professional*), the French word *avocat* can either mean *lawyer* or *avocado*, but the pair was annotated as a positive example. Hence, we leave it to the annotators to handle polysemy by choosing the most appropriate sense.

4.2 Two Evaluation Test Sets

To verify if the crowdsourced hyponyms are challenging negative examples we create two evaluation sets. Both share the (crowdsourced) positive examples, but differ in their negatives:

- HYPER-HYPO – negative examples are the crowdsourced hyponyms.
- HYPER-COHYPO – negative examples are *cohyponyms* drawn from OMW.

Cohyponyms are words sharing a common hypernym. For instance, *bière* ("beer" in French) and *vodka* are cohyponyms since they share a common hypernym in *alcool/alcohol*. We choose cohyponyms for the second test set because: (a) They require differentiating between similarity (a symmetric relation) and hypernymy (an asymmetric relation). For instance, *bière* and *vodka* are highly similar yet, they do not have a hypernymy relationship. (b) Cohyponyms are a popular choice of negative examples in many entailment datasets (Baroni and Lenci, 2011).

5 Experimental Setup

5.1 Data and Evaluation Setup

Training BISPARSE-DEP requires a dependency parsed monolingual corpus, and a translation matrix for jointly aligning the monolingual vectors. We compute the translation matrix using word alignments derived from parallel corpora (see corpus statistics in Table **??**). While we use parallel corpora to generate the translation matrix to be comparable to baselines (§5.2), we can obtain the matrix from any bilingual dictionary.

The monolingual corpora are parsed using `Yara Parser` (Rasooli and Tetreault, 2015), trained on the corresponding treebank from the Universal Dependency Treebank (McDonald et al., 2013) (UDT-v1.4). `Yara Parser` was chosen as it is fast, and competitive with state-of-the-art parsers (Choi et al., 2015). The monolingual corpora was POS-tagged using TurboTagger (Martins et al., 2013). We induce dependency contexts for words by first thresholding the language vocabulary to the top 50,000 nouns, verbs and adjectives. A co-occurrence matrix is computed over this vocabulary using the context types in §3.1.

Inducing Dependency Contexts The entries of the word-context co-occurrence matrix are reweighted using Positive Pointwise Mutual Information (Bullinaria and Levy, 2007). The resulting matrix is reduced to 1000 dimensions using SVD (Golub and Kahan, 1965).[6] These vectors are used as $\mathbf{X_e}, \mathbf{X_f}$ in the setup from §3.3 to generate 100 dimensional sparse bilingual vectors.

Evaluation We use accuracy as our evaluation metric, as it is easy to interpret when the classes are balanced (Turney and Mohammad, 2015). Both evaluation datasets – HYPER-HYPO and HYPER-COHYPO – are split into 1:2 dev/test splits. *BalAPinc* has two tunable parameters - 1) a threshold that indicates the *BalAPinc* score above which all examples are labeled as positive, 2) the maximum number of features to consider for each word. We use the tuning set to tune the two parameters as well as the various hyper-parameters associated with the models.

5.2 Contrastive Approaches

We compare our BISPARSE-DEP embeddings with the following approaches:

MONO-DEP **(Translation baseline)** For word pair (p_f, q_e) in test data, we translate p_f to English using the most common translation in the translation matrix. Hypernymy is then determined using sparse, dependency based embeddings in English.

BISPARSE-LEX **(Window context)** Predecessor of the BISPARSE-DEP model from our previous work (Vyas and Carpuat, 2016). This model induces sparse, cross-lingual embeddings using window based context.

BIVEC+ **(Window context)** Our extension of the BIVEC model of Luong et al. (2015). BIVEC generates dense, cross-lingual embeddings using window based context, by substituting aligned word pairs within a window in parallel sentences. By default, BIVEC only trains using parallel data,

[6] Chosen based on preliminary experiments with {500,1000,2000,3000} dimensional vectors for En-Fr.

Language	Parallel Data	#sent.	Monolingual Data	#sent.
English	–	–	Wackypedia (Baroni et al., 2009)	43M
Arabic	ISI (Munteanu and Marcu, 2007) NewsCommentary, Wikipedia (Tiedemann, 2012)	1.1M	Arabic Gigaword 3.0 (Graff, 2007)	17M
Chinese	FBIS (LDC2003E14)	9.5M	Chinese Gigaword 5.0 (Parker, 2011)	58M
French	Europarl (Koehn, 2005) NewsCommentary♦, Wikipedia (Tiedemann, 2012)	2.7M	Wikipedia♣	20M
Russian	Yandex-1M♠	1.6M	Wikipedia♣	22M

♦ = www.statmt.org/wmt15/training-parallel-nc-v10.tgz, ♣ = dumps.wikimedia.org/xxwiki/20161201/
♠ = translate.yandex.ru/corpus

Table 2: Training data statistics for different languages. Note that while we use parallel corpora for computing translation dictionaries, our approach does not require it, and can work with any bilingual dictionary.

and so we initialize it with monolingually trained window based embeddings to ensure fair comparison.

CL-DEP **(Dependency context)** The model from Vulić (2017), which induces dense, dependency based cross-lingual embeddings by translating syntactic word-context pairs using the most common translation, and jointly training a `word2vecf`[7] model for both languages. Vulić (2017) showed improvements for word similarity and bilingual lexicon induction. We report the first results using CL-DEP on this task.

5.3 Evaluating Robustness of BISPARSE-DEP

We investigate how robust BISPARSE-DEP is when exposed to data scarce settings. Evaluating on a truly low resource language is complicated by the fact that obtaining an evaluation dataset for such a language is difficult. Therefore, we simulate such settings for the languages in our dataset in multiple ways.

No Treebank If a treebank is not available for a language, dependency contexts have to be induced using treebanks from other languages (§3.2), which can affect the quality of the dependency-based embeddings. To simulate this, we train a delexicalized parser for the languages in our dataset. We use treebanks from Slovenian, Ukrainian, Serbian, Polish, Bulgarian, Slovak and Czech (40k sentences) for training the Russian parser, and treebanks from English, Spanish, German, Portuguese, Swedish and Italian (66k sentences) for training the French parser. UDT does not (yet) have languages in the same family as Arabic or Chinese, so for the sake of completeness, we train Arabic and Chinese parsers on delexicalized treebanks of the language itself. After delexicalized training, the Labeled Attachment Score (LAS) on the UDT test set dropped by several points for all languages – from 76.6% to 60.0% for Russian, 83.7% to 71.1% for French, from 76.3% to 62.4% for Arabic and from 80.3% to 53.3% for Chinese. The monolingual corpora are then parsed with these weaker parsers, and co-ocurrences and dependency contexts are computed as before.

Subsampling Monolingual Data To simulate low-resource behavior along another axis, we subsample the monolingual corpora used by BISPARSE-DEP to induce monolingual vectors, $\mathbf{X_e}, \mathbf{X_f}$. Specifically, we learn $\mathbf{X_e}$ and $\mathbf{X_f}$ using progressively smaller corpora.

Quality of Bilingual Dictionary We study the impact of the quality of the bilingual dictionary used to create the translation matrix $\mathbf{S}$. This experiment involves using increasingly smaller parallel corpora to induce the translation dictionary.

6 Experiments

We aim to answer the following questions – **(a)** Are dependency based embeddings superior to window based embeddings for identifying cross-lingual hypernymy? (§6.1) **(b)** Does directionality in the dependency context help cross-lingual hypernymy identification? (§6.2) **(c)** Are our models robust in data scarce settings (§6.3)? **(d)** Is the answer to **(a)** predicated on the choice of entailment scorer? (§6.4)?

6.1 Dependency v/s Window Contexts

We compare the performance of models described in §5.2 with the BISPARSE-DEP (FULL and JOINT) models. We evaluate the models on the two test splits described in §4.2 – HYPER-HYPO and HYPER-COHYPO.

[7] bitbucket.org/yoavgo/word2vecf/

Model ↓ En With →	Ru	Zh	Ar	Fr	Avg.
Translation Baseline					
MONO-DEP	50.1	52.3	51.8	54.5	52.2
Win. Based					
BISPARSE-LEX	56.6	53.7	50.9	52.0	53.3
BIVEC+	55.8	52.0	51.5	53.4	53.2
Dep. Based					
CL-DEP	**60.2**	54.4	**56.7***	53.8	**56.3**
BISPARSE-DEP (Full)	59.0	55.9	52.6	56.6	56.0
BISPARSE-DEP (Joint)	53.8	**57.0***	52.4	**59.9***	55.8
BISPARSE-DEP (Unlab)	55.9	51.2	53.3	55.9	54.1

(a) Performance on HYPER-HYPO.

Model ↓ En With →	Ru	Zh	Ar	Fr	Avg.
Translation Baseline					
MONO-DEP	58.7	50.0	65.1	56.9	57.7
Win. Based					
BISPARSE-LEX	**63.8**	55.8	65.8	63.2	62.2
BIVEC+	55.9	64.9	62.2	54.1	58.3
Dep. Based					
CL-DEP	56.2	62.7	63.1	61.0	60.0
BISPARSE-DEP (Full)	63.6	**67.3**	**66.8***	**66.7***	**66.1**
BISPARSE-DEP (Joint)	60.6	63.6	65.9	64.9	63.8
BISPARSE-DEP (Unlab)	58.6	66.7	62.4	61.5	62.4

(b) Performance on HYPER-COHYPO.

Table 3: Comparing the different approaches from §5.2 with our BISPARSE-DEP approach on HYPER-HYPO and HYPER-COHYPO (random baseline= 0.5). **Bold** denotes the best score for each language, and the * on the best score indicates a statistically significant ($p < 0.05$) improvement over the next best score, using McNemar's test (McNemar, 1947). Across both datasets, BISPARSE-DEP models outperform window based models and the translation baseline on an average.

Hyper-Hypo Results Table 3a shows the results on HYPER-HYPO. First, the benefit of cross-lingual modeling (as opposed to translation) is evident in that almost all models (except CL-DEP on French) outperform the translation baseline. Among dependency based models, BISPARSE-DEP (FULL) and CL-DEP consistently outperform both window models, while BISPARSE-DEP (JOINT) outperforms them on all except Russian. BISPARSE-DEP (JOINT) is the best model overall for two languages (French and Chinese), CL-DEP for one (Arabic), with no statistically significant differences between BISPARSE-DEP (JOINT) and CL-DEP for Russian. This confirms that dependency context is more useful than window context for cross-lingual hypernymy detection.

Hyper-Cohypo Results The trends observed on HYPER-HYPO also hold on HYPER-COHYPO i.e. dependency based models continue to outperform window based models (Table 3b).

Overall, BISPARSE-DEP (FULL) performs best in this setting, followed closely by BISPARSE-DEP (JOINT). This suggests that the sibling information encoded in JOINT is useful to distinguish hypernyms from hyponyms (HYPER-HYPO results), while the dependency labels encoded in FULL help to distinguish hypernyms from co-hyponyms. Also note that all models improve significantly on the HYPER-COHYPO set, suggesting that discriminating hypernyms from cohyponyms is easier than discriminating them from hyponyms.

While the BISPARSE-DEP models were generally performing better than window models on both test sets, CL-DEP was not as consistent (e.g., it was worse than the best window model on HYPER-COHYPO). As shown by Turney and Mohammad (2015), *BalAPinc* is designed for sparse embeddings and is likely to perform poorly with dense embeddings. This explains the relatively inconsistent performance of CL-DEP.

Besides establishing the challenging nature of our crowd-sourced set, the experiments on HYPER-COHYPO and HYPER-HYPO also demonstrate the ability of the BISPARSE-DEP models to discriminate between different lexical semantic relations (viz. hypernymy and cohyponymy) in a cross-lingual setting. We will investigate this ability more carefully in future work.

6.2 Ablating Directionality in Context

The context described by the FULL and JOINT BISPARSE models encodes directional information (§3.1) either in the form of label direction (FULL), or using sibling information (JOINT). Does such directionality in the context help to capture the asymmetric relationship inherent to hypernymy? To answer this, we evaluate a third BISPARSE-DEP model which uses UNLABELED dependency contexts. This is similar to the FULL context, except we do not concatenate the label of the relation to the context word (parent or children). For instance, for *traveler* in Fig. 2, contexts will be *roamed* and *tired*.

Experiments on both HYPER-HYPO and HYPER-COHYPO (bottom row, Tables 3a and 3b) highlight that directional information is indeed essential - UNLABELED almost always performs worse than FULL and JOINT, and in many cases worse than even window based models.

Model ↓ En With →	Ru	Zh	Ar	Fr	Avg.
Hyper-Hypo					
Best Win.	56.6	53.7	51.5	53.4	53.8
Delex.	59.1*	55.1*	54.6*	56.1*	56.2
Best Dep.	60.2	57.0*	56.7*	59.9*	58.5
Hyper-Cohypo					
Best Win.	63.8	64.9	65.8	63.2	64.4
Delex.	59.4	65.7*	67.5*	66.3*	64.7
Best Dep.	63.6*	67.3*	66.8*	66.7	66.1

Table 4: **Robustness in absence of a treebank**: The delexicalized model is competitive to the best dependency based and the best window based models on both test sets. For each dataset, * indicates a statistically significant ($p < 0.05$) improvement over the next best model in that column, using McNemar's test (McNemar, 1947).

6.3 Evaluating Robustness of BISPARSE-DEP

No Treebank We run experiments (Table 4) for all languages with a version of BISPARSE-DEP that use the FULL context type for both English and the non-English (target) language, but the target language contexts are derived from a corpus parsed using a delexicalized parser (§5.3).

This model compares favorably on all language pairs against the best window based and the best dependency based model. In fact, it almost consistently outperforms the best window based model by several points, and is only slightly worse than the best dependency-based model.

Further analysis revealed that the good performance of the delexicalized model is due to the relative robustness of the delexicalized parser on frequent contexts in the co-occurrence matrix. Specifically, we found that in French and Russian, the most frequent contexts were derived from `amod`, `nmod`, `nsubj` and `dobj` edges.[8] For instance, the `nmod` edge appears in 44% of Russian contexts and 33% of the French contexts. The delexicalized parser predicts both the label and direction of the `nmod` edge correctly with an F1 of 68.6 for Russian and 69.6 for French. In contrast, a fully-trained parser achieves a F1 of 76.7 for Russian and 76.8 for French for the same edge.

Small Monolingual Corpus In Figure 4, we use increasingly smaller monolingual corpora (10%, 20%, 40%, 60% and 80%) sampled at random to induce the monolingual vectors for BISPARSE-DEP (FULL) model. Trends (Figure 4) indicate that BISSPARSE-DEP models that use only 40% of the original data remain competitive with the BISSPARSE-LEX model that has access to the full data. Robust performance with smaller monolingual corpora is helpful since large-enough monolingual corpora are not always easily available.

[8]Together they make up at least 70% of the contexts.

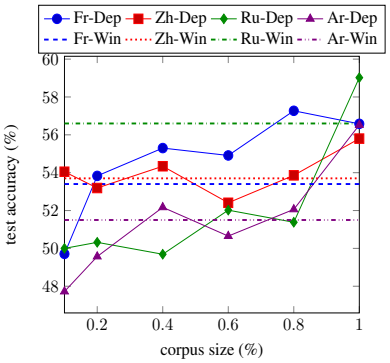


Figure 3: **Robustness to Small Corpus** For most languages, BISPARSE-DEP outperforms the corresponding best window based model for each language on HYPER-HYPO, with about 40% of the monolingual corpora.

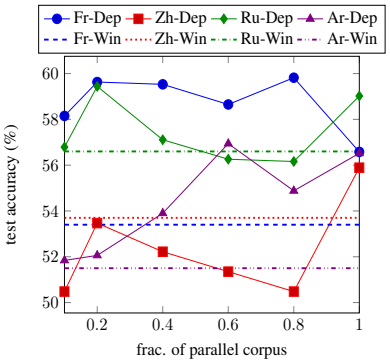


Figure 4: **Robustness to Noisy Dictionary** For most languages, BISPARSE-DEP outperforms the corresponding best window based model on HYPER-HYPO, with increasingly lower quality dictionaries.

Quality of Bilingual Dictionary Bilingual dictionaries derived from smaller amounts of parallel data are likely to be of lower quality than those derived from larger corpora. Hence, to analyze the impact of dictionary quality on BISPARSE-DEP (FULL), we use increasingly smaller parallel corpora to induce bilingual dictionaries used as the score matrix $\mathbf{S}$ (§3.3). We use the top 10%, 20%, 40%, 60% and 80% sentences from the parallel corpora. The trends in Figure 4 show that even with a lower quality dictionary, BISPARSE-DEP performs better than BISPARSE-LEX.

6.4 Choice of Entailment Scorer

We change the entailment scorer from *BalAPinc* to *SLQS* (Santus et al., 2014) and redo experiments from §6.1 to see if the conclusions drawn depend

on the choice of the entailment scorer. *SLQS* is based on the distributional informativeness hypothesis, which states that hypernyms are less "informative" than hyponyms, because they occur in more general contexts. The informativeness E_u of a word u is defined to be the median entropy of its top N dimensions, $E_u = median_{k=1}^{N} H(c_k)$, where $H(c_i)$ denotes the entropy of dimension c_i. The *SLQS* score for a pair (u, v) is the relative difference in entropies,

$$SLQS(u \rightarrow v) = 1 - \frac{E_u}{E_v}$$

Recent work (Shwartz et al., 2017) has found *SLQS* to be more successful than other metrics in monolingual hypernymy detection.

The trends observed in these experiments are consistent with those in §6.1 – both BISPARSE-DEP models still outperform window-based models. Also, the delexicalized version of BISPARSE-DEP outperforms the window-based models, showing that the robust behavior demonstrated in §6.3 is also invariant across metrics.

We also found that using *BalAPinc* led to better results than *SLQS* . For both BISPARSE-DEP models, *BalAPinc* wins across the board for two languages (Russian and Chinese), and wins half the time for the other two languages compared to *SLQS* . We leave detailed comparison of these and other scores to future work.

7 Conclusion

We introduced BISPARSE-DEP, a new distributional approach for identifying cross-lingual hypernymy, based on cross-lingual embeddings derived from dependency contexts. We showed that using BISPARSE-DEP is superior for the cross-lingual hypernymy detection task, when compared to standard window based models and a translation baseline. Further analysis also showed that BISPARSE-DEP is robust to various low-resource settings. In principle, BISPARSE-DEP can be used for any language that has a bilingual dictionary with English and a "related" language with a treebank. We also introduced crowd-sourced cross-lingual hypernymy datasets for four languages for future evaluations.

Our approach has the potential to complement existing work on creating cross-lingual ontologies such as BabelNet and the Open Multilingual Wordnet, which are noisy because they are compiled semi-automatically, and have limited language coverage. In general, distributional approaches can help refine ontology construction for any language where sufficient resources are available.

It remains to be seen how our approach performs for other language pairs beyond simluated low-resource settings. We anticipate that replacing our delexicalized parser with more sophisticated transfer strategies (Rasooli and Collins, 2017; Aufrant et al., 2016) might be beneficial in such settings.While our delexicalized parsing based approach exhibits robustness, it can benefit from more sophisticated approaches for transfer parsing (Rasooli and Collins, 2017; Aufrant et al., 2016) to improve parser performance. We aim to explore these and other directions in the future.

Acknowledgments

The authors would like to thank the members of the CLIP lab at the University of Maryland, members of the Cognitive Computation Group at the University of Pennsylvania, and the anonymous reviewers from EMNLP/CoNLL 2017 and NAACL 2018 for their constructive feedback. YV and MC were funded in part by research awards from Amazon, Google, and the Clare Boothe Luce Foundation. SU and DR were supported by Contract HR0011-15-2-0025 with the US Defense Advanced Research Projects Agency (DARPA).

References

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah Smith. 2016. Many languages, one parser. *Transactions of the Association for Computational Linguistics* 4:431–444.

Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. 2016. Zero-resource dependency parsing: Boosting delexicalized cross-lingual transfer with linguistic knowledge. In *Proc. of COLING*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 119–130. http://aclweb.org/anthology/C16-1012.

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proc. of ACL*. Association for Computational Linguistics, Baltimore, Maryland, pages 809–815. http://www.aclweb.org/anthology/P14-2131.

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. In *Proc. of EACL*. Association for Computational Linguistics, pages 23–32. http://www.aclweb.org/anthology/E12-1004.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The WaCky wide web:

a collection of very large linguistically processed web-crawled corpora. *Proc. of LREC* .

Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics* 36(4):673–721.

Marco Baroni and Alessandro Lenci. 2011. How we BLESSed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop*.

Francis Bond and Ryan Foster. 2013. Linking and extending an open multilingual wordnet. In *Proc. of ACL*. Association for Computational Linguistics, pages 1352–1362. http://www.aclweb.org/anthology/P13-1133.

John A. Bullinaria and Joseph P. Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods* pages 510–526.

Vicente Ivan Sanchez Carmona and Sebastian Riedel. 2017. How well can we predict hypernyms from word embeddings? a dataset-centric analysis. In *Proc. of EACL*. Association for Computational Linguistics, Valencia, Spain, pages 401–407. http://www.aclweb.org/anthology/E17-2064.

Emmanuele Chersoni, Enrico Santus, Alessandro Lenci, Philippe Blache, and Chu-Ren Huang. 2016. Representing verbs with rich contexts: an evaluation on verb similarity. In *Proc. of EMNLP*. Association for Computational Linguistics, Austin, Texas, pages 1967–1972. https://aclweb.org/anthology/D16-1205.

Jinho D Choi, Joel R Tetreault, and Amanda Stent. 2015. It depends: Dependency parser comparison using a web-based evaluation tool. In *Proc. of ACL*. Association for Computational Linguistics, Beijing, China, pages 387–396. http://www.aclweb.org/anthology/P15-1038.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proc. of EACL*. Association for Computational Linguistics, Gothenburg, Sweden, pages 462–471. http://www.aclweb.org/anthology/E14-1049.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin* 76(5):378.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning semantic hierarchies via word embeddings. In *Proc. of ACL*. Association for Computational Linguistics, Baltimore, Maryland, pages 1199–1209. http://www.aclweb.org/anthology/P14-1113.

Maayan Geffet and Ido Dagan. 2005. The Distributional Inclusion Hypotheses and Lexical Entailment. In *Proc. of ACL*.

Tom Goldstein, Christoph Studer, and Richard Baraniuk. 2014. A Field Guide to Forward-Backward Splitting with a FASTA Implementation. *arXiv eprint* abs/1411.3.

Gene Golub and William Kahan. 1965. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the SIAM* .

David Graff. 2007. Arabic gigaword 3rd edition, LDC2003T40. LDC, University of Pennsylvania.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proc. of COLING*. Association for Computational Linguistics, pages 539–545. https://doi.org/10.3115/992133.992154.

Felix Hill, Roi Reichart, and Anna Korhonen. 2014. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *arXiv preprint arXiv:1408.3456* .

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proc. of MT Summit*.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. 2009. Directional distributional similarity for lexical expansion. In *Proc. of the ACL-IJCNLP*.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering* .

Alessandro Lenci and Giulia Benotto. 2012. Identifying hypernyms in distributional semantic spaces. In *Proc. of the 6th Workshop on Semantic Evaluation*.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proc. of ACL*. Association for Computational Linguistics, Baltimore, Maryland, pages 302–308. http://www.aclweb.org/anthology/P14-2050.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proc. of ACL*. Association for Computational Linguistics, pages 768–774. https://doi.org/10.3115/980691.980696.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual word representations with monolingual quality in mind. In *Proc. of the Workshop on Vector Space Modeling for NLP*.

Andre Martins, Miguel Almeida, and Noah A. Smith. 2013. Turning on the Turbo: Fast Third-Order Non-Projective Turbo Parsers. In *Proc. of ACL*. Association for Computational Linguistics, pages 617–622. http://www.aclweb.org/anthology/P13-2109.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao

Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proc. of ACL*.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proc. of EMNLP*. Association for Computational Linguistics, pages 62–72. http://www.aclweb.org/anthology/D11-1006.

Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika* 12(2):153–157.

Oren Melamud, David McClosky, Siddharth Patwardhan, and Mohit Bansal. 2016. The Role of Context Types and Dimensionality in Learning Word Embeddings. In *Proc. of NAACL-HLT*. Association for Computational Linguistics, San Diego, California, pages 1030–1040. http://www.aclweb.org/anthology/N16-1118.

Dragos Stefan Munteanu and Daniel Marcu. 2007. ISI Arabic-English Automatically Extracted Parallel Text LDC2007T08. LDC, University of Pennsylvania.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence* .

Matteo Negri, Alessandro Marchetti, Yashar Mehdad, Luisa Bentivogli, and Danilo Giampiccolo. 2012. Semeval-2012 Task 8: Cross-lingual Textual Entailment for Content Synchronization .

Matteo Negri, Alessandro Marchetti, Yashar Mehdad, Luisa Bentivogli, and Danilo Giampiccolo. 2013. Semeval-2013 Task 8: Cross-lingual Textual Entailment for Content Synchronization .

Sebastian Padó, Michel Galley, Dan Jurafsky, and Chris Manning. 2009. Robust machine translation evaluation with entailment features. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*. Association for Computational Linguistics, pages 297–305.

Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics* .

Robert Parker. 2011. Chinese Gigaword 5th Edition, LDC2011T13. LDC, University of Pennsylvania.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. *Proceedings of ACL-IJCNLP 2015* pages 425–430.

Mohammad Sadegh Rasooli and Michael Collins. 2017. Cross-lingual syntactic transfer with limited resources. *Transactions of the Association for Computational Linguistics* 5:279–293. https://transacl.org/ojs/index.php/tacl/article/view/922.

Mohammad Sadegh Rasooli and Joel R. Tetreault. 2015. Yara Parser: A Fast and Accurate Dependency Parser. *CoRR* abs/1503.06733.

Stephen Roller and Katrin Erk. 2016. Relations such as Hypernymy: Identifying and Exploiting Hearst Patterns in Distributional Vectors for Lexical Entailment. In *Proc. of EMNLP*. Association for Computational Linguistics, pages 2163–2172. http://www.aclweb.org/anthology/D16-1234.

Enrico Santus, Alessandro Lenci, Qin Lu, and Sabine Schulte im Walde. 2014. Chasing hypernyms in vector spaces with entropy. In *Proc. of EACL*. Association for Computational Linguistics, Gothenburg, Sweden, pages 38–42. http://www.aclweb.org/anthology/E14-4008.

Enrico Santus, Frances Yung, Alessandro Lenci, and Chu-Ren Huang. 2015. EVALution 1.0: an evolving semantic dataset for training and evaluation of distributional semantic models. In *Proceedings of the 4th Workshop on Linked Data in Linguistics (LDL-2015)*. Association for Computational Linguistics, Beijing, China, pages 64–69. http://www.aclweb.org/anthology/W15-4208.

Vered Shwartz, Enrico Santus, and Dominik Schlechtweg. 2017. Hypernyms under siege: Linguistically-motivated artillery for hypernymy detection. In *Proc. of EACL*. Association for Computational Linguistics, Valencia, Spain, pages 65–75. http://www.aclweb.org/anthology/E17-1007.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proc. of LREC*.

Peter D Turney and Saif M Mohammad. 2015. Experiments with three approaches to recognizing lexical entailment. *Natural Language Engineering* .

Piek Vossen, Egoitz Laparra, Itziar Aldabe, and German Rigau. 2015. Interoperability of cross-lingual and cross-document event detection. In *Proc. of the 3rd Workshop on EVENTS at the NAACL-HLT*.

Ivan Vulić. 2017. Cross-lingual syntactically informed distributed word representations. In *Proc. of EACL*. Association for Computational Linguistics, Valencia, Spain, pages 408–414. http://www.aclweb.org/anthology/E17-2065.

Ivan Vulić and Marie-Francine Moens. 2015. Bilingual word embeddings from non-parallel document-aligned data applied to bilingual lexicon induction. In *Proc. of ACL*. Association for Computational Linguistics, Beijing, China, pages 719–725. http://www.aclweb.org/anthology/P15-2118.

Yogarshi Vyas and Marine Carpuat. 2016. Sparse Bilingual Word Representations for Cross-lingual Lexical Entailment. Association for Computational Linguistics, San Diego, California, pages 1187–1197. http://www.aclweb.org/anthology/N16-1142.

Julie Weeds and David Weir. 2003. A general framework for distributional similarity. In *Proc. of EMNLP*. http://www.aclweb.org/anthology/W03-1011.

Daniel Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In *IJCNLP*. http://www.aclweb.org/anthology/I08-3008.

Noising and Denoising Natural Language: Diverse Backtranslation for Grammar Correction

Ziang Xie, Guillaume Genthial, Stanley Xie, Andrew Y. Ng, Dan Jurafsky
Computer Science Department, Stanford University
{zxie,genthial,stanxie,ang}@cs.stanford.edu,jurafsky@stanford.edu

Abstract

Translation-based methods for grammar correction that directly map noisy, ungrammatical text to their clean counterparts are able to correct a broad range of errors; however, such techniques are bottlenecked by the need for a large parallel corpus of noisy and clean sentence pairs. In this paper, we consider synthesizing parallel data by noising a clean monolingual corpus. While most previous approaches introduce perturbations using features computed from local context windows, we instead develop error generation processes using a neural sequence transduction model trained to translate clean examples to their noisy counterparts. Given a corpus of clean examples, we propose beam search noising procedures to synthesize additional noisy examples that human evaluators were nearly unable to discriminate from nonsynthesized examples. Surprisingly, when trained on additional data synthesized using our best-performing noising scheme, our model approaches the same performance as when trained on additional nonsynthesized data.

1 Introduction

Correcting noisy, ungrammatical text remains a challenging task in natural language processing. Ideally, given some piece of writing, an error correction system would be able to fix minor typographical errors, as well as grammatical errors that involve longer dependencies such as nonidiomatic phrasing or errors in subject-verb agreement. Existing methods, however, are often only able to correct highly local errors, such as spelling errors or errors involving articles or prepositions.

Classifier-based approaches to error correction are limited in their ability to capture a broad range of error types (Ng et al., 2014). Machine translation-based approaches—that instead translate noisy, ungrammatical sentences to clean, corrected sentences—can flexibly handle a large variety of errors; however, such approaches are bottlenecked by the need for a large dataset of source-target sentence pairs.

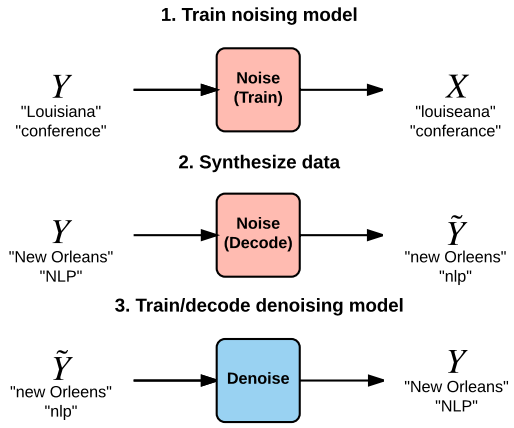


Figure 1: Overview of method. We first train a noise model on a seed corpus, then apply noise during decoding to synthesize data that is in turn used to train the denoising model.

To address this data sparsity problem, we propose methods for synthesizing noisy sentences from clean sentences, thus generating an additional artificial dataset of noisy and clean sentence pairs. A simple approach to noise clean text is to noise individual tokens or bigrams, for example by replacing each token with a random draw from the unigram distribution. This type of approach, however, tends to generate highly unrealistic noise and fails to capture phrase-level phenomena. Other rule-based approaches fail to capture a diverse set of error types.

We consider a method inspired by the backtranslation procedure for machine translation (Sennrich et al., 2015). Our method combines a neural sequence transduction trained on a seed corpus of clean→noisy pairs with *beam search*

Proceedings of NAACL-HLT 2018, pages 619–628
New Orleans, Louisiana, June 1 - 6, 2018.

noising procedures to produce more diversity in the decoded outputs. This technique addresses two issues with existing synthesis techniques for grammar correction:

1. By using a neural model trained end-to-end on a large corpus of noisy and clean sentences, the model is able to generate rich, diverse errors that better capture the noise distribution of real data.

2. By encouraging diversity through applying noise to hypotheses during decoding, we avoid what we refer to as the *one-to-many* problem, where decoding from a model trained on clean→noisy examples results in overly clean output, since clean subphrases still form the majority of noisy examples.

We perform experiments using several noising methods to validate these two claims, yielding gains on two benchmarks. Our main empirical result is that, starting with only clean news data and models trained on a parallel corpus of roughly 1.3 million sentences, we can train models with additional synthesized data that nearly match the performance of models trained on 3 million nonsynthesized examples.

2 Related work

Noising While for images, there are natural noising primitives such as rotations, small translational shifts, and additive Gaussian noise, similar primitives are not as well developed for text data. Similarly, while denoising autoencoders for images have been shown to help with representation learning (Vincent et al., 2010), similar methods for learning representations are not well developed for text. Some recent work has proposed noising—in the form of dropping or replacing individual tokens—as a regularizer when training sequence models, where it has been demonstrated to have a smoothing effect on the softmax output distribution (Bowman et al., 2015; Xie et al., 2017; Dai and Le, 2015; Kumar et al., 2015).

Grammar correction Recent work by Chollampatt and Ng (2018) has achieved impressive performance on the benchmarks we consider using convolutional encoder-decoder models. Previous work using data synthesis for grammatical error correction (GEC) has introduced errors by examining the distribution of error types, then applying errors according to those distributions together with lexical or part-of-speech features based on a small context window (Brockett et al., 2006; Felice, 2016). While these methods can introduce many possible edits, they are not as flexible as our approach inspired by the backtranslation procedure for machine translation (Sennrich et al., 2015). This is important as neural language models not explicitly trained to track long-range linguistic dependencies can fail to capture even simple noun-verb errors (Linzen et al., 2016). Recently, in the work perhaps most similar to ours, Rei et al. (2017) propose using statistical machine translation and backtranslation along with syntactic patterns for generating errors, albeit for the error detection task.

Neural machine translation Recent end-to-end neural network-based approaches to machine translation have demonstrated strong empirical results (Sutskever et al., 2014; Cho et al., 2014). Building off of these strong results on machine translation, we use neural encoder-decoder models with attention (Bahdanau et al., 2014) for both our data synthesis (noising) and grammar correction (denoising) models. Although many recent works on NMT have focused on improving the neural network architecture, the model architecture is orthogonal to the contributions in this work, where we instead focus on data synthesis. In parallel to our work, work on machine translation without parallel corpora has also explored applying noise to avoid copying when pretraining autoencoders by swapping adjacent words (Lample et al., 2017; Artetxe et al., 2017).

Diverse decoding Key to the data generation procedure we describe is adding noise to the scores of hypotheses during beam search–otherwise, decoded outputs tend to contain too few errors. This is inspired by work in dialogue, in which neural network models tend to produce common, overly generic responses such as "*I don't know*" (Sordoni et al., 2015; Serban et al., 2015). To mitigate this issue, Li et al. (2015) and others have proposed methods to increase the diversity of neural network outputs. We adopt a similar approach to Li et al. (2015) to generate noisier hypotheses during decoding.

3 Method

We first briefly describe the neural model we use, then detail the noising schemes we apply when synthesizing examples.

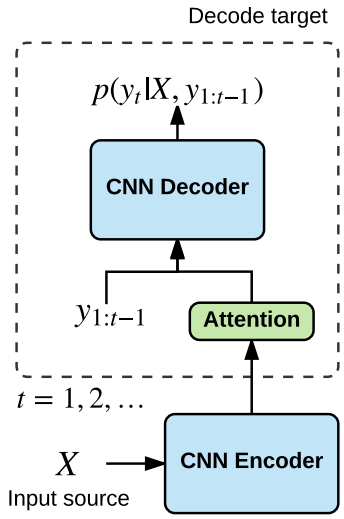


Figure 2: Model architecture used for both noising and denoising networks.

3.1 Model

In order to generate noisy examples as well as to translate ungrammatical examples to their corrected counterparts, we need to choose a sequence transduction model. Based off their strong empirical performance, we use a neural network-based model for this work.

Our method uses two neural encoder-decoder models:

1. The first is the *noising* model, which, given a clean sentence, is used to generate a noised version of that sentence. This model is trained on a *seed corpus* of parallel clean→noisy sentences.

2. The second is the *denoising* model, which, given a noisy, ungrammatical sentence, generates the clean, corrected sentence.

For both models, we use the same convolutional encoder-decoder to model

$$p(Y|X) = \prod_{t=1}^{T_Y} p(y_t|X, y_{1:t-1}; \theta)$$

where $X = (x_1, x_2, \ldots, x_{T_X})$ is the source sequence and $Y = (y_1, y_2, \ldots, y_{T_Y})$ the corresponding target sequence, and we minimize the training loss

$$\ell(\theta) = -\log \sum_{t=1}^{T_Y} p(y_t|X, y_{1:t-1}; \theta)$$

thus maximizing log-likelihood. The model architecture we use is similar to that described by Kalchbrenner et al. (2016) and Gehring et al. (2017). Gated convolutions are applied with masking—to avoid peeking at future inputs when training using teacher forcing—such that they form an autoregressive network similar to a recurrent neural network with gated hidden units. This architecture was selected so that training steps could be parallelized across the time dimension through the use of convolutions. However, we emphasize that the architecture is not a focus of this paper, and we would expect that RNN architectures with LSTM cells would achieve similar results. For simplicity and to avoid handling out-of-vocabulary words, we use character-level tokenization. Figure 2 illustrates the model architecture.

3.2 Noising

The amount of parallel data is often the limiting factor in the performance of neural network systems. In order to obtain more parallel examples for the grammar correction task, we take clean text Y and apply noise, yielding noisy text $\tilde{Y}$, then train a denoising model to map from $\tilde{Y}$ back to Y. The noising process used to generate $\tilde{Y}$ greatly affects final performance. First, we consider noising methods which we use as our baselines, as well as the drawbacks for each method.

- **appending clean examples**: We first consider simply appending clean examples with no noise applied to both the source and the target. The aim is for the decoder to learn a better language model when trained on additional clean text, similar to the motivation described in Dai and Le (2015). However, for the models we consider, the attention mechanism allows copying of source to target. Thus the addition of examples where source and target are identical data may also cause the model to become too conservative with edits and thus reduce the recall of the system.

- **token noising**: Here we simply consider a context window of at most two characters or words and allow word/character deletions and transpositions.

  First, for every *character* in each word we sample deletions, followed by transpositions. Then we sample deletions and transpositions for every *word* in the sentence. Deletion and transposition probabilities were selected such

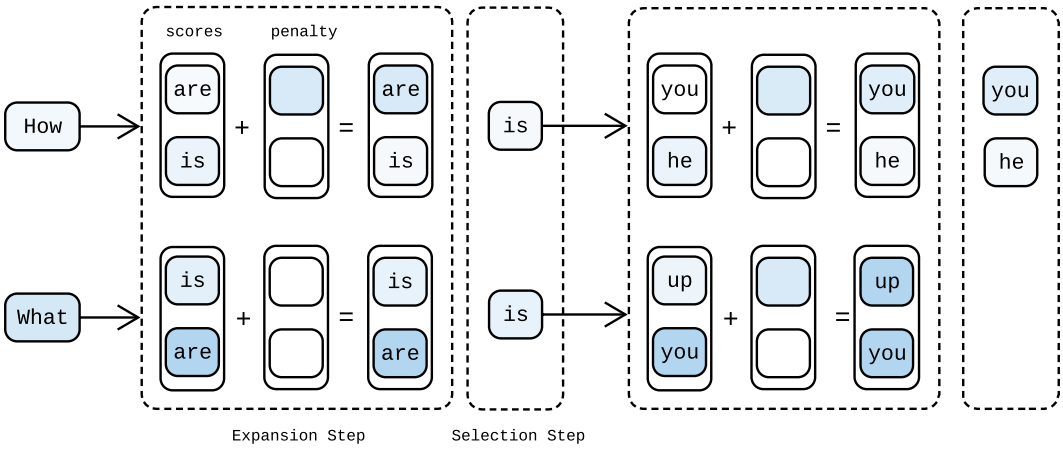


Figure 3: Illustration of random noising with beam width 2. Darker shading indicates less probable expansions. In this example, greedy decoding would yield "How are you". Applying noise penalties, however, results in the hypotheses "How is you/he". Note that applying a penalty does not always result in an expansion falling off the beam.

that overall character and word-level edit distances roughly matched the edit distances between clean and noisy examples in our parallel seed corpus. While this method is fast to apply, it tends to produce highly unrealistic errors leading to a mismatch between the synthesized and real parallel data.

- **reverse noising**: For reverse noising, we simply train a reverse model from $Y \rightarrow X$ using our parallel noisy-clean corpus and run standard beam search to generate noisy targets $\tilde{Y}$ from clean inputs Y. However, we find vanilla reverse noising tends to be too conservative. This is due to the *one-to-many* problem where a clean sentence has many possible noisy outputs which mostly consist of clean phrases. The output then contains far fewer errors on average than the original noisy text.

To address the drawback of the reverse noising scheme, we draw inspiration from ideas for increasing diversity of outputs in dialogue (Li et al., 2016). During the beam search procedure, we add noise to the scores of hypotheses on the beam to encourage decoding to stray from the greedy output. Recall that during beam search, we iteratively grow a set of hypotheses $\mathcal{H} = \{h_1, h_2, \ldots\}$, only keeping the top hypotheses after each step of decoding according to some scoring function $s(h)$. Extending the reverse noising scheme, the *beam search noising* schemes we consider are:

- **rank penalty noising** We directly apply the method of Li et al. (2016). At every step of the search procedure, siblings from the same parent are penalized by adding $k\beta_{\text{rank}}$ to their scores, where k is their rank (in descending log-likelihood) amongst their siblings and β_{rank} is a penalty hyperparameter corresponding to some log-probability.

- **top penalty noising** Only the top (most-probable) hypothesis h_{top} of the beam is penalized by adding β_{top} to its score $s(h_{\text{top}})$.

- **random noising** Every hypothesis is penalized by adding $r\beta_{\text{random}}$ to its score, where r is drawn uniformly from the interval $[0, 1]$. For sufficiently large β_{random}, this leads to a random shuffling of the ranks of the hypotheses according to their scores.

An illustration of the random noising algorithm is shown in Figure 3. Note that although rank penalty noising should encourage hypotheses whose parents have similar scores to remain on the beam, it can also tend to leave the hypothesis from greedy decoding on the beam in the case where softmax output distributions are highly peaked. This is much more of an issue for tasks that involve significant copying of source to target, such as grammar correction. Note also that the random noising can yield more diverse outputs than top penalty noising, depending on the probability with which each is applied. All of the beam search noising methods described are intended to increase the diversity and the amount of noise in the synthe-

Corpus	Sent. Pairs
CoNLL 2014	60K
Lang-8	1.3M
Lang-8 expanded	3.3M
synthesized (NYT 2007)	1.0M
base (CoNLL + L8)	1.3M
expanded (CoNLL + L8 expanded)	3.3M

Table 1: Summary of training corpora.

sized outputs $\tilde{Y}$. By performing beam search noising, we can produce errors such as those shown in Table 4.

3.3 Denoising

Once noised data has been generated, *denoising* simply involves using a neural sequence transduction model to backtranslate the noised text to the original clean text. For denoising, during decoding we apply length normalization as well as a coverage penalty to the scoring function $s(h)$ (Wu et al., 2016). The final scoring function also incorporates a 5-gram language model trained on a subset of Common Crawl, estimated with Kneser-Ney smoothing using KenLM (Heafield, 2011). We incorporate the language model during final reranking by modifying the score for a completed hypothesis $s(h)$ to be

$$s_{\text{LM}}(h) = s(h) + \lambda \log p_{\text{LM}}(h)$$

where λ is a hyperparameter and $p_{\text{LM}}(h)$ is given by the language model.

4 Experiments

To determine the effectiveness of the described noising schemes, we synthesize additional data using each and evaluate the performance of models trainined using the additional data on two benchmarks.

Datasets For training our sequence transduction models, we combine the publicly available English Lang-8 dataset, a parallel corpus collected from a language learner forum, with training data from the CoNLL 2014 challenge (Mizumoto et al., 2011; Ng et al., 2014). We refer to this as the "base" dataset. Junczys-Dowmunt and Grundkiewicz (2016) additionally scraped 3.3M pairs of sentences from Lang-8. Although this expanded dataset, which we call the "expanded" dataset, is not typically used when comparing performance on grammar correction benchmarks, we use it instead to compare performance when training on additional synthesized data versus nonsynthesized data. For clean text to be noised, we use the LDC New York Times corpus for 2007, which yields roughly 1 million sentences. A summary of the data used for training is given in Table 1.

We use the CoNLL 2013 evaluation set as our development set in all cases (Ng et al., 2013). Our test sets are the CoNLL 2014 evaluation set and the JFLEG test set (Ng et al., 2014; Napoles et al., 2017). Because CoNLL 2013 only has a single set of gold annotations while CoNLL 2014 has two, performance metrics tend to be significantly higher on CoNLL 2014. We report precision, recall, and $F_{0.5}$ score, which is standard for the task, as precision is valued over recall. On JFLEG, we report results with the GLEU metric (similar to BLEU) developed for the dataset.

Training and decoding details All models are trained using stochastic gradient descent with annealing based on validation perplexity on a small held-out subset of the Lang-8 corpus. We apply both dropout and weight decay regularization. We observed that performance tended to saturate after 30 epochs. Decoding is done with a beam size of 8; in early experiments, we did not observe significant gains with larger beam sizes (Koehn and Knowles, 2017).

4.1 CoNLL

Results for the CoNLL 2013 (dev) and 2014 (test) datasets but with and without language model reranking are given in Table 2. In general, adding noised data helps, while simply adding clean data leads the model to be too conservative. Overall, we find that the random noising scheme yields the most significant gain of 4.5 F-score. Surprisingly, we find that augmenting the base dataset with synthesized data generated with random noising yields nearly the same performance when compared to using only nonsynthesized examples. To determine whether this might be due to overfitting, we reduced the dropout rate when training on the "expanded" dataset, but did not observe better results.

The random noising scheme achieves the best performance, while the top noising scheme matches the best performance on the development set but not the test set. We believe this is due to a mismatch between the CoNLL 2013 dev and 2014

Method	Dev (no LM)			Dev			Test		
	P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$
none	50.7	10.5	28.7	48.4	17.2	35.5	52.7	27.5	44.5
clean	56.1	9.4	28.1	47.5	16.9	34.8	52.3	27.5	44.3
token	49.7	11.9	30.4	47.7	18.7	36.4	51.4	30.3	45.1
reverse	53.1	13.0	32.8	50.5	19.1	38.0	54.7	29.6	46.8
rank	51.3	12.3	31.4	51.0	18.3	37.6	54.3	29.3	46.4
top	49.1	17.4	36.0	47.7	23.9	**39.8**	50.9	34.7	46.6
random	50.0	17.9	**36.8**	48.9	23.0	**39.9**	54.2	35.4	**49.0**
expanded	64.4	11.2	33.0	54.9	20.0	40.7	57.2	32.0	**49.4**
Yuan and Briscoe (2016)	—	—	—	—	—	—	—	—	39.9
Ji et al. (2017)	—	—	28.6	—	—	33.5	—	—	45.2
Junczys-Dowmunt et al. (2016)	—	—	—	—	—	—	61.3	28.0	49.5
Chollampatt and Ng (2018)	—	—	—	—	—	—	65.5	33.1	**54.8**

Table 2: Results on CoNLL 2013 (Dev) and CoNLL 2014 (Test) sets. All results use the "base" parallel corpus of 1.3M sentence pairs along with additional synthesized data (totaling 2.3M sentence pairs) except for "expanded", which uses 3.3M nonsynthesized sentence pairs (and no synthesized data).

tets sets. Since the 2013 dev set has only a single annotator, methods are encouraged to target higher recall, such that the top noising scheme was optimized for precision over recall. To check this, we ran decoding on CoNLL 2014 using the best dev settings with no language model, and found that the top noising scheme yielded an $F_{0.5}$-score of 45.2, behind only random (47.1) and ahead of token (42.0) and reverse (43.9) noising. Overall, we find the data synthesis method we describe to yield large gains in recall.

For completeness, we also compare to other state-of-the-art systems, such as the phrase-based machine translation system by Junczys-Dowmunt and Grundkiewicz (2016), who performed parameter tuning with sparse and dense features by cross-validation on the CoNLL 2014 training set. Chollampatt and Ng (2018) achieve even higher state-of-the-art results using the neural machine translation model of Gehring et al. (2017) along with improvements to the reranking procedure.

4.2 JFLEG

Recently, Napoles et al. (2017) introduced the JFLEG dataset, intended to evaluate the fluency of grammar correction systems rather than simply the precision and recall of edits. The evaluation metric proposed is GLEU, a variant of BLEU score. Most results for this task were reported with hyperparameter settings from the CoNLL task; hence we report results with the best settings on our CoNLL 2013 dev set. Results are shown in Table 3[1]. Token noising performs surprisingly well; we suspect this is because a significant portion of errors in the JFLEG dataset are spelling errors, as demonstrated from strong gains in performance by using a spelling checker reported by Chollampatt and Ng (2018).

5 Discussion

Our experiments illustrate that synthesized parallel data can yield large gains on the grammar correction task. However, what factors make for an effective data synthesis technique? We consider the properties of the noising scheme and the corresponding data that lead to better performance.

5.1 Realism and Human Evaluation

First, we manually compare each of the different noising methods to evaluate how "realistic" the errors introduced are. This is reminiscent of the generative adversarial network setting (Goodfellow et al., 2014), where the generator seeks to produce samples that fool the discriminator. Here the discriminator is a human evaluator who, given the clean sentence Y, tries to determine which of two sentences X and $\tilde{Y}$ is the true noisy sentence, and which is the synthesized sentence. To be clear,

[1]Comparisons taken from https://github.com/keisks/jfleg

Scheme	P	R	$F_{0.5}$	GLEU
none	68.9	44.2	62.0	53.9
clean	69.2	42.8	61.6	54.1
token	69.2	47.6	**63.5**	55.9
reverse	69.1	42.1	61.3	53.8
rank	68.3	43.3	61.2	54.4
top	67.3	48.2	62.4	55.5
random	69.1	48.5	**63.7**	**56.6**
expanded	72.7	45.9	**65.1**	**56.2**
Sakaguchi et al. (2017)†				54.0
Ji et al. (2017)				53.4
Yuan and Briscoe (2016)				52.1
Junczys-Dowmunt et al. (2016)				51.5
Chollampatt and Ng (2018)				**57.5**

Table 3: Results on the JFLEG test set (we use best hyperparameter settings from CoNLL dev set). GLEU is a variant of BLEU developed for this task; higher is better (Napoles et al., 2017). †Tuned to JFLEG dev set.

we do not train with a discriminator—the beam search noising procedures we proposed alone are intended to yield convincing errors.

For each noising scheme, we took 100 (X, Y) pairs from the development set (500 randomly chosen pairs combined), then generated $\tilde{Y}$ from Y. We then shuffled the examples and the order of X and $\tilde{Y}$ such that the identity of X and $\tilde{Y}$ *as well as the noising scheme* used to generate $\tilde{Y}$ were unknown[2]. Given Y, the task for human evaluators is to predict whether X or $\tilde{Y}$ was the synthesized example. For every example, we had two separate evaluators label the sentence they thought was synthesized. We chose to do this labeling task ourselves (blind to system) since we were familiar with the noising schemes used to generate examples, which should reduce the number of misclassifications. Results are shown in Figure 4, and examples of the evaluation task are provided in Table 4.

5.2 Noise Frequency and Diversity

Comparing the performance using different noising methods on the CoNLL 2014 dataset to the human evaluation in the previous section, we see that generating errors which *match* the real distribution tends to result in higher performance, as seen by the poor performance of token noising relative

[2]Hence the human labelers cannot favor a particular scheme unless it can be distinguished from $\tilde{Y}$.

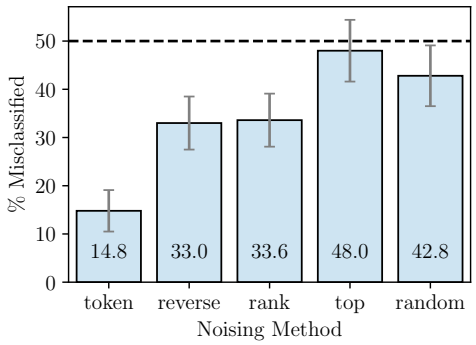


Figure 4: Percentage of time human evaluators misclassified synthesized noisy sentence $\tilde{Y}$ (vs. X) when using each noising scheme, along with 95% confidence intervals. The best we can expect any scheme to do is 50%.

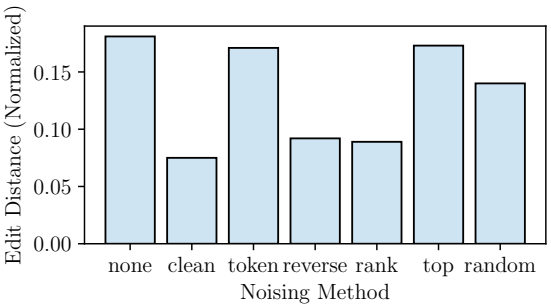


Figure 5: Mean edit distance between sentence pairs in X and Y *after* augmentation with noised sentences. *none* contains no synthesized examples while *clean* refers to the baseline of simply appending clean examples (source = target).

to the other methods. Injecting the appropriate *amount* of noise is important as well, as seen by improved performance when using beam search noising to increase diversity of outputs, and no performance gain when simply adding clean text.

We observe that token noising, despite matching the frequency of errors, fails to generate realistic errors (Figure 4). On the other hand, reverse noising yields significantly more convincing errors, but the edit distance between synthesized examples is significantly lower than in real data (Figure 5). A combination of sufficient amounts of noise and rich, diverse errors appears to lead to better model performance.

5.3 Error Type Distribution Mismatch

Mismatches in the distribution of error types can often severely impact the performance of data synthesis techniques for grammar correction (Felice, 2016). For example, only synthesizing noun number articles or preposition errors based on rules

	Sentence	1 or 2
clean	Day after day , I get up at 8 o‘clock .	
1	I got up at 8 o‘clock day after day .	
2	Day after day , I get up 8 o‘clock in the week .	
clean	Thanks Giving Day in Korea is coming soon .	
1	In Korea , it ’s coming soon , thanks Giving day .	
2	Thanks Giving Day in korea is coming soon .	
clean	After I practiced , I could play the song perfectly .	
1	After the results , I could accomplish without a fault .	
2	When I tried that , I could play the song perfectly .	
clean	Currently , I ’m studying to take the TOEIC exam for my future career .	
1	I am studying to take TOEIC exam for career of my future .	
2	Currently , I will have take TOEIC exam for future career .	
clean	There is one child who is 15 years old and a mother who is around 50 .	
1	There are one child who is 15 years old and mother is around 50 .	
2	It has one child , 15 years old and the mother who is around 50 years old .	
clean	But at the beginning , I suffered from a horrible pain in my jaw .	
1	But at the first time , I suffer from a horrible pain on my jaw .	
2	But at the beginning , I suffered from a horrible pain in my jaw joint .	

Table 4: Examples of nonsynthesized and synthesized sentences from validation set. Which example (1 or 2) was synthesized? Answers: 1, 1, 2, 1, 2, 1

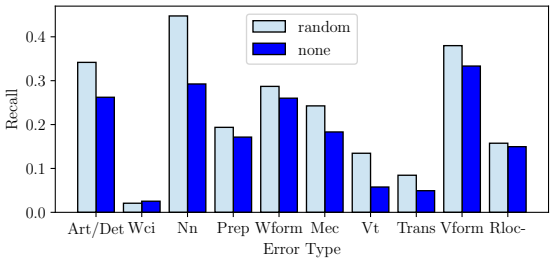


Figure 6: Recall vs. error type for the ten most frequent error types in our dev set. Noising improves recall uniformly across error types (See Ng et al. (2014) for a description of error types).

may improve the performance for those two error types, but may hurt overall performance. In contrast, the approaches we consider, with the exception of token noising, are fully data-driven, and hence we would expect gains across all different error types. We observe this is the case for random noising, as shown in Figure 6.

5.4 Data Sparsity and Domain Adaptation

Domain adaptation can yield significant differences in performance for dissimilar domains (such as those of the datasets used in our experiments) (Daumé III, 2009). The Lang-8, CoNLL, and JFLEG datasets contain online forum data and essay data from English learners. The n-gram language model is estimated using Common Crawl data from the web. The clean data which we noise is collected from a news corpus. Yet each dataset yields significant gains. This suggests that at current levels of system performance, data sparsity remains the key data issue, more so than domain adaptation.

It is also possible that LDC New York Times data is better matched to the CoNLL essay data than the Lang-8 forum data, and this in part accounts for the large gains we observe from training on synthesized data.

6 Conclusion

In this work, we address one of the key issues for developing translation-based grammar correction systems: the need for a large corpus of parallel data. We propose synthesizing parallel data by noising clean text, where instead of applying noise based on finite context windows, we instead train a reverse model and apply noise during the beam search procedure to synthesize noisy examples that human evaluators were nearly unable to distinguish from real examples. Our experiments suggest that the proposed data synthesis technique can yields almost as strong results as when training with additional nonsynthesized data. Hence, we hope that parallel data becomes less of a bottleneck, and more emphasis can be placed on developing better models that can capture the longer dependencies and structure in the text.

Acknowledgments

We thank the anonymous reviewers for their helpful feedback, as well as Steven Tan for comments on an early draft.

References

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2017. Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349* .

Chris Brockett, William B Dolan, and Michael Gamon. 2006. Correcting esl errors using phrasal smt techniques. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 249–256.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

Shamil Chollampatt and Hwee Tou Ng. 2018. A multilayer convolutional encoder-decoder neural network for grammatical error correction. *arXiv preprint arXiv:1801.08831* .

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*. pages 3061–3069.

Hal Daumé III. 2009. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815* .

Mariano Felice. 2016. Artificial error generation for translation-based grammatical error correction. Technical report, University of Cambridge, Computer Laboratory.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122* .

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

Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, pages 187–197.

Jianshu Ji, Qinlong Wang, Kristina Toutanova, Yongen Gong, Steven Truong, and Jianfeng Gao. 2017. A nested attention neural hybrid model for grammatical error correction. *arXiv preprint arXiv:1707.02026* .

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Phrase-based machine translation is state-of-the-art for automatic grammatical error correction. *arXiv preprint arXiv:1605.06353* .

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099* .

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. *arXiv preprint arXiv:1706.03872* .

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *arXiv preprint arXiv:1506.07285* .

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043* .

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055* .

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. A simple, fast diverse decoding algorithm for neural generation. *arXiv preprint arXiv:1611.08562* .

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *arXiv preprint arXiv:1611.01368* .

Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2011. Mining revision log of language learning sns for automated japanese error correction of second language learners. In *International Joint Conference on Natural Language Processing (IJCNLP)*.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. Jfleg: A fluency corpus and benchmark for grammatical error correction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, Valencia, Spain,

pages 229–234. http://www.aclweb.org/anthology/E17-2037.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The conll-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*. pages 1–14.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task (CoNLL-2013 Shared Task)*.

Marek Rei, Mariano Felice, Zheng Yuan, and Ted Briscoe. 2017. Artificial error generation with machine translation and syntactic patterns. *arXiv preprint arXiv:1707.05236* .

Keisuke Sakaguchi, Matt Post, and Benjamin Van Durme. 2017. Grammatical error correction with neural reinforcement learning. *arXiv preprint arXiv:1707.00299* .

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709* .

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2015. Building end-to-end dialogue systems using generative hierarchical neural network models. *arXiv preprint arXiv:1507.04808* .

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714* .

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

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research* 11(Dec):3371–3408.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* .

Ziang Xie, Sida I Wang, Jiwei Li, Daniel Lévy, Aiming Nie, Dan Jurafsky, and Andrew Y Ng. 2017. Data noising as smoothing in neural network language models. *arXiv preprint arXiv:1703.02573* .

Zheng Yuan and Ted Briscoe. 2016. Grammatical error correction using neural machine translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 380–386.

Self-Training for Jointly Learning to Ask and Answer Questions

Mrinmaya Sachan **Eric P. Xing**
School of Computer Science
Carnegie Mellon University
{mrinmays, epxing}@cs.cmu.edu

Abstract

Building curious machines that can answer as well as ask questions is an important challenge for AI. The two tasks of question answering and question generation are usually tackled separately in the NLP literature. At the same time, both require significant amounts of supervised data which is hard to obtain in many domains. To alleviate these issues, we propose a self-training method for jointly learning to ask as well as answer questions, leveraging unlabeled text along with labeled question answer pairs for learning. We evaluate our approach on four benchmark datasets: *SQUAD*, *MS MARCO*, *WikiQA* and *TrecQA*, and show significant improvements over a number of established baselines on both question answering and question generation tasks. We also achieved new state-of-the-art results on two competitive answer sentence selection tasks: *WikiQA* and *TrecQA*.

1 Introduction

Question Answering (QA) is a well-studied problem in NLP which focuses on answering questions using some structured or unstructured sources of knowledge. Alongside question answering, there has also been some work on generating questions (QG) (Heilman, 2011; Du et al., 2017; Tang et al., 2017) which focuses on generating questions based on given sources of knowledge.

QA and QG are closely related[1] tasks. However, NLP literature views the two as entirely separate tasks. In this paper, we explore this relationship between the two tasks by jointly learning to generate as well as answer questions. An improved ability to generate as well as answer questions will help us build *curious* machines that can interact with humans in a better manner. Joint modeling of QA and QG is useful as the two can be used in conjunction to generate novel questions from free text and then answers for the generated questions. We use this idea to perform self-training (Nigam and Ghani, 2000) and leverage free text to augment the training of QA and QG models.

QA and QG models are typically trained on question answer pairs which are expensive to obtain in many domains. However, it is cheaper to obtain large quantities of free text. Our self-training procedure leverages unlabeled text to boost the quality of our QA and QG models. This is achieved by a careful data augmentation procedure which uses pre-trained QA and QG models to generate additional labeled question answer pairs. This additional data is then used to retrain our QA and QG models and the procedure is repeated.

This addition of synthetic labeled data needs to be performed carefully. During self-training, typically the most *confident* samples are added to the training set (Zhu, 2005) in each iteration. We use the performance of our QA and QG models as a proxy for estimating the *confidence* value of the questions. We describe a suite of heuristics inspired from curriculum learning (Bengio et al., 2009) to select the questions to be generated and added to the training set at each epoch. Curriculum learning is inspired from the incremental nature of human learning and orders training samples on the *easiness* scale so that *easy* samples can be introduced to the learning algorithm first and *harder* samples can be introduced successively. We show that introducing questions in increasing order of *hardness* leads to improvements over a baseline that introduces questions randomly.

We use a seq2seq model with soft attention (Sutskever et al., 2014; Bahdanau et al., 2014) for QG and a neural model inspired from *Attentive Reader* (Hermann et al., 2015; Chen et al., 2016) for QA. However, these can be any QA

[1] We can think of QA and QG as inverse of each other.

Proceedings of NAACL-HLT 2018, pages 629–640
New Orleans, Louisiana, June 1 - 6, 2018.

and QG models. We evaluate our approach on four datasets: *SQUAD*, *MS MARCO*, *WikiQA* and *TrecQA*. We use a corpus of English Wikipedia as unlabeled text. Our experiments show that the self-training approach leads to significant improvements over a number of established approaches in QA and QG on these benchmarks. On the two answer sentence selection QA tasks: (*WikiQA* and *TrecQA*), we obtain state-of-the-art.

2 Problem Setup

In this work, we focus on the task of machine comprehension where the goal is to answer a question $\mathbf{q}$ based on a passage $\mathbf{p}$. We model this as an answer sentence selection task i.e., given the set of sentences in the passage $\mathbf{p}$, the task is to select the sentence $\mathbf{s} \in \mathbf{p}$ that contains the answer a. Treating QA as an answer sentence selection task is quite common in literature (e.g. see Yu et al., 2014). We model QG as the task of transforming a sentence in the passage into a question. Previous work in QG (Heilman and Smith, 2009) transforms text sentences into questions via some set of manually engineered rules. However, we take an end-to-end neural approach.

Let $\mathcal{D}^0$ be a labeled dataset of (passage, question, answer) triples where the answer is given by selecting a sentence in the passage. We also assume access to unlabeled text $\mathcal{T}$ which will be used to augment the training of the two models.

3 The Question Answering Model

Since we model QA as the task of selecting an answer sentence from the passage, we treat each sentence in the corresponding passage as a candidate answer for every input question.

We employ a neural network model inspired from the *Attentive Reader* framework proposed in Hermann et al. (2015); Chen et al. (2016). We map all words in the vocabulary to corresponding d dimensional vector representations via an embedding matrix $E \in \mathbb{R}^{d \times V}$. Thus, the input passage $\mathbf{p}$ can be denoted by the word sequence $\{p_1, p_2, \ldots p_{|\mathbf{p}|}\}$ and the question $\mathbf{q}$ can similarly be denoted by the word sequence $\{q_1, q_2, \ldots q_{|\mathbf{q}|}\}$ where each token $p_i \in \mathbb{R}^d$ and $q_i \in \mathbb{R}^d$.

We use a bi-directional LSTM (Graves et al., 2005) with dropout regularization as in Zaremba et al. (2014) to encode contextual embeddings of each word in the passage:

$$\vec{\mathbf{h}}_t = \mathit{LSTM}_1\left(p_t, \vec{\mathbf{h}}_{t-1}\right), \overleftarrow{\mathbf{h}}_t = \mathit{LSTM}_2\left(p_t, \overleftarrow{\mathbf{h}}_{t+1}\right)$$

The final contextual embeddings $\mathbf{h}_t$ are given by concatenation of the forward and backward pass embeddings: $\mathbf{h}_t = [\vec{\mathbf{h}}_t; \overleftarrow{\mathbf{h}}_t]$. Similarly, we use another bi-directional LSTM and encode contextual embeddings of each word in the question.

Then, we use attention mechanism (Bahdanau et al., 2014) to compute the alignment distribution a based on the relevance among passage words and the question: $a_i = \mathit{softmax}\left(\mathbf{q}^T \mathbf{W} \mathbf{h}_i\right)$. The output vector o is a weighted combination of all contextual embeddings: $\mathbf{o} = \sum_i a_i \mathbf{h}_i$. Finally, the correct answer a^* among the set of candidate answers $\mathcal{A}$ is given by: $a^* = \arg\max_{a \in \mathcal{A}} \mathbf{w}^T \mathbf{o}$.

We learn the model by maximizing the log-likelihood of correct answers. Given the training set $\{\mathbf{p}^{(i)}, \mathbf{q}^{(i)}, \mathbf{a}^{(i)}\}_{i=1}^N$, the log-likelihood is:

$$\mathcal{L}_{QA} = \sum_{i=1}^{N} \mathit{logP}\left(\mathbf{a}^{(i)} | \mathbf{p}^{(i)}, \mathbf{p}^{(i)}; \theta\right)$$

Here, θ represents all the model parameters to be estimated.

4 The Question Generation Model

We use a seq2seq model (Sutskever et al., 2014) with soft attention (Bahdanau et al., 2014) as our QG model. The model transduces an input sequence $\mathbf{x}$ to an input sequence $\mathbf{y}$. Here, the input sequence is a sentence in the passage and the output sequence is a generated question. Let $\mathbf{x} = \{x_1, x_2, \ldots, x_{|\mathbf{x}|}\}$, $\mathbf{y} = \{y_1, y_2, \ldots, y_{|\mathbf{y}|}\}$ and $\mathcal{Y}$ be the space of all possible output questions. Thus, we can represent the QG task as finding $\hat{\mathbf{y}} \in \mathcal{Y}$ such that: $\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x})$.

Here, $P(\mathbf{y}|\mathbf{x})$ is the conditional probability of a question sequence $\mathbf{y}$ given input sequence $\mathbf{x}$.

Decoder: Following Sutskever et al. (2014), the conditional factorizes over token level predictions:

$$P(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{|\mathbf{y}|} P(y_t | \mathbf{y}_{<t}, \mathbf{x})$$

Here, $\mathbf{y}_{<t}$ represents the subsequence of words generated prior to the time step t. For the decoder, we again follow Sutskever et al. (2014):

$$P(y_t | \mathbf{y}_{<t}, \mathbf{x}) = \mathit{softmax}\left(\mathbf{W} \mathit{tanh}\left(\mathbf{W}_t [\mathbf{h}_t^{(d)}; \mathbf{c}_t]\right)\right)$$

Here, $\mathbf{h}_t^{(d)}$ is the decoder RNN state at time step t, and $\mathbf{c}_t$ is the attention based encoding of the input sequence $\mathbf{x}$ at decoding time step t (described later). Also $\mathbf{W}$ and $\mathbf{W}_t$ are model parameters to be learned. We use an LSTM with dropout (Zaremba et al., 2014) as the decoder RNN. The LSTM generates the new decoder state $\mathbf{h}_t^{(d)}$ given the representation of previously generated word y_{t1} obtained using a look-up dictionary, and the previous decoder state $\mathbf{h}_{t-1}^{(d)}$.

Encoder: We use a bi-directional LSTM (Graves et al., 2005) with attention mechanism as our sentence encoder. We use two LSTM's: one that makes a forward pass in the sequence and another that makes a backward pass as in the QA model described earlier. We use dropout regularization for LSTMs as in Zaremba et al. (2014) in our implementation. The final context dependent token representation $\mathbf{h}_t^{(e)}$ is the concatenation of the forward and backward pass token representations: $\mathbf{h}_t^{(e)} = [\overrightarrow{\mathbf{h}}_t^{(e)}; \overleftarrow{\mathbf{h}}_t^{(e)}]$. To obtain the final context dependent token representation $\mathbf{c}_j$ at the decoding time step j, we take a weighted average over token representations: $\mathbf{c}_j^{(d)} = \sum_{i=1}^{|\mathbf{x}|} a_{ij}\mathbf{h}_i^{(e)}$. Following Bahdanau et al. (2014), the attention weights a_{ij} are calculated by bilinear scoring followed by softmax normalization:

$$\mathbf{a}_{ij} = \frac{\exp\left(\mathbf{h}_j^{(e)^T}\mathbf{W}\,\mathbf{h}_i^{(d)}\right)}{\sum_{i'}\exp\left(\mathbf{h}_j^{(e)^T}\mathbf{W}\,\mathbf{h}_{i'}^{(d)}\right)}$$

Learning and Inference: We train the encoder decoder framework by maximizing data log-likelihood on a large training set with respect to all the model parameters θ. Let $\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^N$ be the training set. The log-likelihood can be written as:

$$\begin{aligned}\mathcal{L}_{QG} &= \sum_{i=1}^{N} logP\left(\mathbf{y}^{(i)}|\mathbf{x}^{(i)};\theta\right) \\ &= \sum_{i=1}^{N}\sum_{j=1}^{|\mathbf{y}^{(i)}|} logP\left(y_j^{(i)}|\mathbf{x}^{(i)}, \mathbf{y}_{<j}^{(i)};\theta\right)\end{aligned}$$

We use beam search for inference. As in previous works, we introduce a <UNK> token to model rare words during decoding. These <UNK> tokens are finally replaced by the token in the input sentence with the highest attention score.

5 Self-training Framework for Joint Training of QA and QG models

In our self-training framework, we are given unlabeled text in addition to the labeled passages, question and answer pairs. Self-training (Yarowsky, 1995; Riloff et al., 2003), also known as self-teaching, is one of the earliest techniques for using unlabeled data along with labeled data to improve learning. During self-training, the learner keeps on labeling unlabeled examples and retraining itself on an enlarged labeled training set. We extend self-training to jointly learn two models (namely, QA and QG) iteratively. The QA and QG models are first trained on the labeled corpus. Then, the QG model is used to create more questions from the unlabeled text corpus and the QA model is used to answer these newly created questions. These new questions (carefully selected by an oracle – details later) and the original labelled data is then used to (stochastically) update these two models. This procedure can be repeated as long as both the two models continue to improve.

Algorithm 1: Self-training QA and QG.

$\theta_{\mathbf{qa}}^{(\mathbf{0})} \leftarrow$ Train initial QA model.
$\theta_{\mathbf{qg}}^{(\mathbf{0})} \leftarrow$ Train initial QG model.
Init: i = 0
while *performance on dev set rises* **do**
  $\mathbf{CQ_i} \leftarrow$ Set of candidate questions generated using our QG model $\theta_{\mathbf{qg}}^{(\mathbf{i})}$ from the unlabeled text $\mathcal{T}$ which are not in $\mathcal{D}$.
  $\mathbf{Q_i} \leftarrow k \times m^i$ questions drawn from $\mathbf{CQ_i}$ using our question selector oracle $\mathcal{QS}$.
  $\mathbf{A_i} \leftarrow$ Set of answers to questions $\mathbf{Q_i}$ obtained using our QA model $\theta_{\mathbf{qa}}^{(\mathbf{i})}$.
  Let $\mathcal{D}^i$ be the set of chosen questions $\mathbf{Q_i}$ and answers $\mathbf{A_i}$.
  Subsample $\mathcal{S}_1 \subset \mathcal{D}^i$ of size k_1 and $\mathcal{S}_2 \subset \mathcal{D}^0$ of size k_2. Let $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$
  $\theta_{\mathbf{qa}}^{(\mathbf{i+1})} \leftarrow$ Update QA model on $\mathcal{S}$.
  $\theta_{\mathbf{qg}}^{(\mathbf{i+1})} \leftarrow$ Update QG model on $\mathcal{S}$.
  i++
end

Algorithm 1 describes the procedure in detail. In each succesive iteration, we allow the addition of more questions than that introduced in the previous iteration by a multiplicative factor. This sheme adds fewer questions initially when the QA and QG models are weak and more questions thereafter when the two models have (hopefully) improved. We found that this scheme works better in practice than addiing a fixed number of questions in each iteration. The two models are

updated on a subsample of the newly generated datapoints and original unlabelled data.

Self-training has been seldom used in NLP. Most prominently, it has been used for WSD (Yarowsky, 1995), noun learning (Riloff et al., 2003) and AMR parsing and generation (Konstas et al., 2017). However, it has not been explored in this way for QA and QG.

5.1 The Question Selection Oracle

A key challenge in self-training is selecting which unlabeled data sample to label (iwhich generated questions to add to the training set). The self-training process may erroneously generate some bad or incorrect questions which can sidetrack the learning process. Thus, we implement a question selection oracle which determines which questions to add among the potentially very large set of questions generated by the QG model in each iteration.

Traditional wisdom in self-training (Yarowsky, 1995; Riloff et al., 2003) advises selecting a subset of questions on which the models have the highest *confidence*. We experiment with this idea, proposing multiple self-training oracles which introduce questions in the order of how *confident* the QA and QG models are on the new potential question:

- **QG:** The QG oracle introduces the question in the order of how confident the QG model is on generating the question. This is calculated by a number of heuristics (described later).
- **QA:** The QA oracle introduces the question in the order of how confident the QA model is on answering the question. This too is calculated by some heuristics (described later).
- **QA+QG:** The QA+QG oracle introduces a question when both QA and QG models are confident about the question. The oracle computes the minimum confidence of the QA and QG models for a question and introduces questions which have the the highest minimum confidence score.

Our question selection heurisitcs are based on the ideas of *curriculum learning* and *diversity*:

1. *Curriculum learning* (Bengio et al., 2009; Sachan and Xing, 2016a) requires ordering questions on the easiness scale, so that easy questions can be introduced to the learning algorithm first and harder questions can be introduced successively. The main challenge in learning the curriculum is that it requires the identification of easy and hard questions. In our setting, such a ranking of easy and hard questions is difficult to obtain. A human judgement of 'easiness' of a question might not correlate with what is easy for our algorithms in its feature and hypothesis space. We explore various heuristics that define a measure of easiness and learn the ordering by selecting questions using this measure.
2. A number of cognitive scientists (Cantor, 1946) argue that alongside curriculum learning, it is important to introduce diverse (even if sometimes hard) samples. Inspired by this, we introduce a measure of *diversity* and show that we can achieve further improvements by coupling the curriculum learning heuristics with a measure for diversity.

Curriculum Learning: Studies in cognitive science (Skinner, 1958; Peterson, 2004; Krueger and Dayan, 2009) have shown that humans learn much better when the training examples are not randomly presented but organized in increasing order of difficulty. In the machine learning community, this idea was introduced with the nomenclature of *curriculum learning* (Bengio et al., 2009), where a curriculum is designed by ranking samples based on manually curated difficulty measures. A manifestation of this idea is self-paced learning (SPL) (Kumar et al., 2010; Jiang et al., 2014, 2015) which selects samples based on the local loss term of the sample. We extend this idea and explore the following heuristics for our various oracles:

1) Greedy Optimal (GO): The simplest greedy heuristic is to pick a question q which has the minimum expected effect on the QA and QG models. The expected effect on adding q can be written as:

$$\sum_{a \in \mathcal{A}} p(a^* = a) \mathbb{E}[\mathcal{L}_{QA/QG}]$$

Here, $\mathcal{L}_{QA/QG}$ is $\mathcal{L}_{QA}$, $\mathcal{L}_{QG}$ or $min\left(\mathcal{L}_{QA}, \mathcal{L}_{QG}\right)$ depending on which oracle we are using. $p(a^* = a)$ can be estimated by computing the scores of each of the answer candidates for q and normalizing them. $\mathbb{E}[\mathcal{L}_{QA/QG}]$ can be estimated by retraining the model(s) after adding this question.

2) Change in Objective (CiO): Choose question q that causes the smallest increase in $\mathcal{L}_{QA/QG}$. If

there are multiple questions with the smallest increase in objective, pick one of them randomly.

3) Mini-max (M^2): Choose question q that minimizes the expected risk when including the question with the answer candidate a that yields the maximum error.

$$\hat{q} = \arg\min_{q} \max_{a \in \mathcal{A}} \mathcal{L}_{QA/QG}$$

4) Expected Change in Objective (ECiO): In this greedy heuristic, we pick a question q which has the minimum expected effect on the model. The expected effect can be written as:

$$\sum_{a} p(a^* = a) \times \mathbb{E}\left[L_{QA/QG}\right]$$

Here, $p(a^* = a)$ can again be achieved by computing the scores of each of the answer candidates for q and normalizing them and $\mathbb{E}\left[L_{QA/QG}\right]$ can be estimated by evaluating the model.

5) Change in Objective-Expected Change in Objective (CiO - ECiO): We pick a question q which has the minimum value of the difference between the change in objective and the expected change in objective described above. Intuitively, the difference represents how much the model is surprised to see this new question.

Time Complexity: GO and CiO require updating the model, M^2 and ECiO require performing inference on candiate questions, and CiO - ECiO requires retraining as well as inference. Thus, M^2 and ECiO are computationally most efficient.

Ensembling: We introduce an ensembling strategy that combines the heuristics into an ensemble. We tried two ensembling strategies. The first strategy computes the average score over all the heuristics for all potential (top-K in beam) questions and picks questions with the highest average. The second strategy uses minimum instead of the average. Minimum works better than average in practice and we use it in our experiments. The use of minimum is inspired by agreement-based learning (Liang et al., 2008), a well-known extension of self-training which uses multiple views of the data (described using different feature sets or models) and adds new unlabeled samples to the training set when multiple models agree on the label.

Diversity: The strategy of introducing easy questions first and then gradually introducing harder questions is intuitive as it allows the learner to improve gradually. Yet, it has one key deficiency. With curriculum learning, by focusing on easy questions first, our learning algorithm is usually not exposed to a diverse set of questions. This is particularly a problem for deep-learning approaches that learn representations during the process of learning. Hence, when a harder question arrives, it can be difficult for the learner to adjust to the new question as the current representation may not be appropriate for the new level of question difficulty. We tackle this by introducing an explore and exploit (**E&E**) strategy. *E&E* ensures that while we still select easy questions first, we also want to make our selection as diverse as possible. We define a measure for diversity as the angle between the question vectors: $\angle \mathbf{q}_i, \mathbf{q}_j = \mathit{Cosine}^{-1}\left(\frac{|\mathbf{q}_i \mathbf{q}_j|}{||\mathbf{q}_i|| ||\mathbf{q}_j||}\right)$. *E&E* picks the question which optimizes a convex combination (tuned on the dev set) of the curriculum learning objective and sum of angles between the candidate questions and the questions in the training set.

6 Experiments

Implementation Details: We perform the same preprocessing on all the text. We lower-case all the text. We use NLTK for word tokenization. For training our neural networks, we only keep the most frequent 50k words (including entity and placeholder markers), and map all other words to a special <UNK> token. We choose word embedding size d = 100, and use the 100-dimensional pretrained GloVe word embeddings (Pennington et al., 2014) for initialization. We set k, m, k_1 and k_2 (hyperparameters for self-training) by grid search on a held-out development set.

Datasets: We report our results on four datasets: *SQUAD* (Rajpurkar et al., 2016), *MS MARCO* (Nguyen et al., 2016), *WikiQA* (Yang et al., 2015) and *TrecQA* (Wang et al., 2007). *SQUAD* is a cloze-style reading comprehension dataset with questions posed by crowd workers on a set of Wikipedia articles, where the answer to each question is a segment of text from the corresponding reading passage. *MS MARCO* contains questions which are real anonymized queries issued through *Bing* or *Cortana* and the documents are related web pages which may or help answer the question. *WikiQA* is also a datset of queries taken from Bing query logs. Based on user clicks, each query is associated with a Wikipedia page. The summary paragraph of the page is taken as candidate answer sentences, with labels on whether the sentence is a correct answer to the question provided by crowd

	SQUAD			MS MARCO			WikiQA			TrecQA		
	Train	Dev	Test	Train	Dev	Test	Train	Dev	Test	Train	Dev	Test
#Questions	82,326	4,806	5,241	87,341	5,273	5,279	1,040	140	293	1,229	82	100
#Question-Answer Pairs	676,193	39,510	42,850	440,573	26,442	26,604	20,360	2,733	6,165	53,417	1,148	1,517

Table 1: Statistics of the four datasets used in evaluating our QA and QG models.

workers. Finally, *TrecQA* is a QA answer sentence selection dataset from the TREC QA track.

While *WikiQA* and *TrecQA* are directly answer sentence selection tasks, the other two are not. Hence, we treat the *SQUAD* and *MS MARCO* tasks as the answer sentence selection task assuming a one to one correspondence between answer sentences and annotated correct answer spans. Note that only a very small proportion of answers ($<$ 0.2% in training set) span two or more sentences. Since *SQUAD* and *MS MARCO* have a hidden test set, we only use the training and development sets for our evaluation purposes and we further split the provided development set into a dev and test set. This is also the data analysis setting used in previous works (Du et al., 2017; Tang et al., 2017). In fact, we use the same setting as in Tang et al. (2017) for comparison. The statistics of the four datasets and the respective train, dev and test splits are given in Table 1. For *WikiQA* and *TrecQA* datasets, we use the standard data splits. We use a large randomly subsampled corpus of English Wikipedia and use the first paragraph of each document as unlabeled text for self-training.

Evaluation Metrics: Following Tang et al. (2017), we evaluate our QA system with three standard evaluation metrics: *Mean Average Precision* (MAP), *Mean Reciprocal Rank* (MRR) and *Precision@1* (P@1). For QG, we follow Du et al. (2017) and use automatic evaluation metrics from MT and summarization: *BLEU-4* (Papineni et al., 2002), *METEOR* (Denkowski and Lavie, 2014) and $Rouge_L$ (Lin, 2004) to measure the overlap between generated and ground truth questions.

Baselines: For *SQUAD* and *MS MARCO* datasets, we use four QA baselines that have been used in previous works (Tang et al., 2017). The first two baselines, *WordCnt* and *NormWordCnt*, have been taken from Yang et al. (2015) and Yin et al. (2015), and are based on simple word overlap which have been shown to be strong baselines. These compute word co-occurrence between a question sentence and the candidate answer sentence. While *WordCnt* uses unnormalized word co-occurrence, *NormWordCnt* uses normalized word co-occurrence. The third and fourth baselines are *CDSSM* (Shen et al., 2014) and *ABCNN* (Yin et al., 2015) which use a neural network approach to model semantic relatedness of sentence pairs. For the *WikiQA* and *TrecQA* dataset, we report results of various existing state-of-the-art approaches on the two datasets[2].

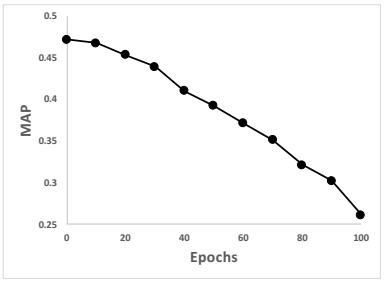


Figure 1: MAP for our best self-trained QA model (with 10,000 Wikipedia paragraphs) without any curriculum learning (i.e. candidate questions are added randomly) vs epochs.

For QG, we compare our model against the following four baselines used in previous work (Du et al., 2017). The first baseline is a simple *IR* baselines taken from Rush et al. (2015) which generates questions by memorizing them from the training set and uses edit distance (Levenshtein, 1966) to calculate distance between a question and the input sentence. The second baseline is a MT system – *MOSES* (Koehn et al., 2007) which models question generation as a translation task where raw sentences are treated as source texts and questions are treated as target texts. The third baseline, *DirectIn*, uses the longest sub-sentence of the input sentence (using a set of simple sentence splitters) as the question. The fourth baseline, *H&S* is a rule-based overgenerate-and-rank system proposed by Heilman and Smith (2010).

The Question Selection Oracle: The first question we wish to answer is: *Is careful question selection even necessary*? To answer this, we plot MAP scores for our best QA model (QA+QG, Ensemble+E&E) when we do not have a curriculum learning based oracle (i.e. an oracle which picks questions to be added to the dataset randomly) in Figure 1 as a function of epochs. We observe that

[2] https://aclweb.org/aclwiki/Question_Answering_(State_of_the_art)

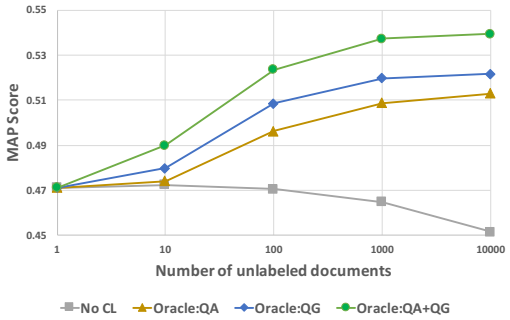


Figure 2: MAP for the best models for the three oracles: QA, QG and QA+QG. Also on the same plot, MAP when we have no curriculum learning.

	SQUAD			MS MARCO		
	MAP	MRR	P@1	MAP	MRR	P@1
WordCnt	0.396	0.401	0.179	0.809	0.817	0.689
NormWordCnt	0.422	0.429	0.203	0.871	0.879	0.796
CDSSM	0.443	0.449	0.228	0.798	0.804	0.672
ABCNN	0.469	0.477	0.263	0.869	0.875	0.784
Tang et al. (2017)	0.484	0.491	0.275	0.864	0.872	0.781
Ens+E&E(0)	0.471	0.478	0.263	0.858	0.865	0.774
Ens+E&E(100)	0.524	0.493	0.273	0.881	0.890	0.799
Ens+E&E(1000)	0.537	0.502	0.284	0.885	0.895	**0.801**
M^2	0.489	0.490	0.268	0.860	0.872	0.785
ECiO	0.498	0.494	0.273	0.877	0.886	0.793
GO	0.506	0.495	0.274	0.879	0.889	0.793
CiO	0.511	0.498	0.277	0.879	0.890	0.795
CiO-ECiO	0.517	0.500	0.280	0.881	0.892	0.798
Ensemble	0.539	0.504	0.284	0.886	0.895	0.800
Ens+E&E(10000)	**0.539**	**0.507**	**0.289**	**0.889**	**0.896**	0.801

Table 2: Performance of our models and QA baselines on *SQUAD* and *MS MARCO* datasets. Shaded part of the table shows results of various question selection heuristics when 10000 Wiki paragraphs are used as unlabeled data.

the MAP score degrades instead of improving with time. This supports our claim that we need to augment the training set by a more careful procedure.

We also plot MAP scores for our best QA model (Ensemble+E&E) when we use various question selection oracles as a function of the amount of unlabeled data in Figure 2. We can observe that when we do not have a curriculum learning based oracle, the MAP score degrades by having more and more unlabeled data. We also observe that the QA+QG oracle performs better than QA and QG which confirms that the best oracle is one that selects questions in increasing degree of hardness in terms of both question answering and question generation. This holds for all the experimental settings. Thus we only show results for the QA+QG strategies in our future experiments.

Evaluating Question Answering: First, we evaluate our models on the question answering task. *Ensemble+E&E(K)* is the variant where we perform self-training using K Wikipedia paragraphs. Hence, *Ensemble+E&E(0)* is the variant of our model without any self-training. We vary K to see the impact of the size of unlabeled Wikipedia paragraphs on the self-training model.

	MAP	MRR
CNN (Yang et al., 2015)	0.665	0.652
APCNN (Santos et al., 2016)	0.696	0.689
NASM (Miao et al., 2016)	0.707	0.689
ABCNN (Yin et al., 2015)	0.702	0.692
KVMN (Miller et al., 2016)	0.707	0.727
Wang et al. (2016b)	0.706	0.723
Wang et al. (2016a)	0.734	0.742
Wang and Jiang (2016)	0.743	0.755
Tang et al. (2017)	0.700	0.684
Ensemble+E&E(0)	0.691	0.675
Ensemble+E&E(100)	0.718	0.719
Ensemble+E&E(1000)	0.734	0.733
M^2	0.719	0.704
ECiO	0.721	0.708
GO	0.725	0.710
CiO	0.727	0.719
CiO-ECiO	0.734	0.724
Ensemble	0.743	0.743
Ensemble+E&E(10000)	**0.754**	**0.753**

Table 3: Performance of our models and the QA baselines on the *WikiQA* dataset. Shaded part of the table shows the effect of various question selection heuristics when 10000 Wikipedia paragraphs are used as unlabeled data. Our model achieves the state-of-the-art.

	MAP	MRR
He and Lin (2016)	0.758	0.822
He et al. (2015)	0.762	0.830
Tay et al. (2017)	0.770	0.825
Rao et al. (2016)	0.780	0.834
Ensemble+E&E(0)	0.742	0.813
Ensemble+E&E(100)	0.776	0.831
Ensemble+E&E(1000)	0.783	0.836
M^2	0.759	0.816
ECiO	0.762	0.822
GO	0.759	0.823
CiO	0.762	0.826
CiO-ECiO	0.767	0.830
Ensemble	0.789	0.843
Ensemble+E&E(10000)	**0.798**	**0.854**

Table 4: Performance of our models and the QA baselines on the *TrecQA* dataset. Shaded part of the table shows the effect of various question selection heuristics when 10000 Wikipedia paragraphs are used as unlabeled data. Our model achieves the state-of-the-art.

Table 2 shows the results of the QA evaluations on the *SQUAD* and *MS MARCO* datasets. We can observe that our QA model has competetive or better performance over all the baselines on both datasets in terms of all the three evaluation metrics. When we incorporate ensembling or diversity, we see a further improvement in the result.

Tables 3 and 4 show results of QA evaluations on the *WikiQA* and *TrecQA* datasets, respectively. We can again observe that our QA model is competitive to all the baselines. When we introduce ensembling and diversity while jointly learning the QA and QG models, we see incremental improvements. In both these answer sentence selection tasks, our approach achieves new state-of-the-art.

	SQUAD			MS MARCO			WikiQA			TrecQA		
	B	M	R	B	M	R	B	M	R	B	M	R
IR	1.07	7.77	20.85	0.81	5.42	15.78	0.93	6.89	19.98	0.83	5.73	16.34
MOSES	0.31	10.49	17.88	0.27	9.74	15.82	0.32	10.26	17.27	0.29	9.86	17.02
DirectIn	11.25	14.91	22.51	10.82	13.35	20.38	10.94	14.18	22.01	9.59	12.21	19.76
H&S	11.23	16.00	31.03	10.16	15.07	30.00	10.35	15.30	30.72	9.19	12.72	23.38
Tang et al. (2017)	5.03	-	-	9.31	-	-	3.15	-	-	-	-	-
Du et al. (2017)	12.28	16.62	39.75	-	-	-	-	-	-	-	-	-
Ens.+E&E(0)	12.31	16.67	39.78	11.14	15.60	37.26	11.38	16.08	38.42	10.96	14.25	27.27
Ens.+E&E(100)	14.14	18.70	42.46	13.25	17.10	40.28	13.10	17.00	40.93	11.63	15.05	29.07
Ens.+E&E(1000)	14.27	18.78	42.93	13.61	17.87	**41.23**	13.22	18.34	42.72	12.24	15.93	30.26
M^2	12.46	16.95	40.27	11.56	15.93	38.32	11.83	16.84	39.26	11.52	16.42	28.92
ECiO	12.79	17.40	40.92	12.11	16.32	38.86	12.14	17.04	39.82	11.67	16.59	29.12
GO	13.12	17.73	41.24	12.75	16.66	39.47	12.56	17.62	40.31	11.61	16.52	29.10
CiO	13.59	17.94	41.57	13.00	16.83	40.02	12.88	18.13	40.97	11.97	16.68	29.89
CiO-ECiO	13.97	18.18	41.90	13.41	17.16	40.65	13.22	18.34	41.28	12.24	16.65	29.63
Ensemble	**14.37**	18.57	42.73	13.56	17.40	40.92	14.26	18.91	43.26	13.32	16.76	30.12
Ens.+E&E(10000)	14.28	**18.79**	**42.97**	**13.74**	**17.89**	41.07	**15.26**	**19.45**	**44.77**	**14.87**	**16.88**	**31.91**

Table 5: Performance (B: BLEU4, M: METEOR, and R: ROUGE) of our model variants and various QG baselines on *SQUAD*, *MS MARCO* and *WikiQA* datasets. The shaded part of the table shows the effect of various question selection heuristics when 10000 Wikipedia paragraphs are used as unlabeled data. The performance numbers for Tang et al. (2017) and Du et al. (2017) were not reported for all the settings.

Evaluating Question Generation: Table 5 shows the results for QG on the four datasets on each of the three evaluation metrics on all the four datasets. We can observe that the QG model described in our paper performs much better than all the baselines. We again observe that self-training while jointly training the QA and QG models leads to even better performance. These results show that self-training and leveraging the relationship between QA and QG is very useful for boosting the performance of the QA and QG models, while additionally only using cheap unlabeled data.

Human Evaluations: We asked two people not involved with this research to evaluate 1000 (randomly selected) questions generated by our best QG model and our best performing baseline (Du et al., 2017) on SQUAD for fluency and correctness on a scale of 1 to 5. The raters were also shown the passage sentence used to generate the question. The raters were blind to which system produced which question. The Pearson correlation between the raters' judgments was $r = 0.89$ for fluency and $r = 0.78$ for correctness. In our analyses, we used the averages of the two raters' judgments. The evaluation showed that our system generates questions that are more fluent and correct than those by the baseline. The mean fluency rating of our best system was 4.15 compared to 3.35 for the baseline, a difference which is statistically significant (t-test, $p < 0.001$).

Evaluating the Question Selection Oracle: As discussed earlier, the choice of which subset of questions to add to our labeled dataset while self-training is important. To evaluate the various heuristics proposed in our paper, we show the effect of the question selection oracle on the final QA and QG performance in Tables 2, 3, 4 and 5. These comparisions are shown in the shaded grey portions of the tables for self-training with 10,000 Wikipedia paragraphs as unlabeled data.

We can observe that all the proposed heuristics (and ensembling and diversity strategies) lead to improvements in the final performance of both QA and QG. The heuristics arranged in increasing order of performance are: M^2, *ECiO*, *GO*, *CiO* and *CiO-ECiO*. While the choice of which heuristic to pick seems to make a lesser impact on the final performance, we do see a much more significant performance gain by *ensembling* to combine the various heuristics and using *E&E* to incorporate diversity. The incorporatation of diversity is important because the neural network models which learnt latent represention of data usually find it hard to adjust to new level of difficulty of questions as the current representation may not be appropriate for the new level of difficulty.

Low data scenarios: A key advantage of our self-training approach is that it can leverage unlabeled text, and thus requires less labeled data. To test this, we plot MAP for our best self-training model and various QA baselines as we vary the proportion of labeled training set in Figure 3. However, we keep the unlabeled text fixed (10K Wikipedia paragraphs). We observe that all the baselines significantly drop in performance as we reduce the proportion of labeled training set. However, the drop happens at a much slower rate for our self-trained model. Thus, we can conclude that our approach requires less labeled data as compared to the baselines.

Does more unlabeled text always help?: Another important question is: *Does more unlabeled*

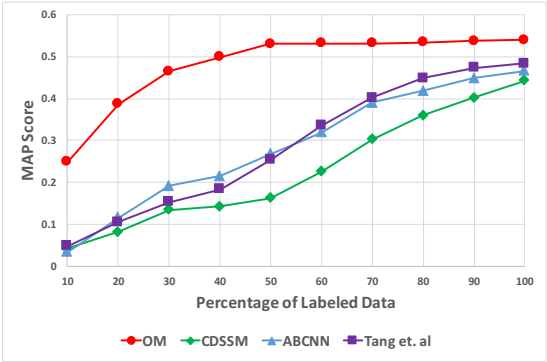


Figure 3: MAP for the best self-training model and QA baselines as we vary the proportion of labeled training set but keep the unlabeled text fixed (10K Wikipedia paragraphs).

text always improve our models? Will the performance improve if we add more and more unsupervised data during self-training. According to our results in Tables 2, 3, 4 and 5, the answer is "probably yes". As we can observe from these tables, the performance of the QA and QG models improves as we increase K, the size of the unsupervised data during training of the various *Ensemble+E&E(K)* models. Having said that, we do see a tapering effect on the performance results, so it is clear that the performance will be capped by some upper-bound and we will need better ways of modeling language and meaning to make progress.

7 Related Work

Our work proposes an approach for joint modeling QA and QG. While QA has recieved a lot of attention from the research community with large scale community evaluations such as *NTCIR*, *TREC*, *CLEF* spurring progress, the focus on QG is much more recent. Recently, there has been a renewed interest in reading comprehensions (also known as *machine comprehension* – a nomenclature popularized by Richardson et al. (2013)). Various approaches (Sachan et al., 2015; Wang et al., 2015; Sachan and Xing, 2016b; Sachan et al., 2016; Narasimhan and Barzilay, 2015) have been proposed for solving this task. After the release of large benchmarks such as *SQUAD*, *MS MARCO* and *WikiQA*, there has been a surge in interest on using neural network or deep-learning models for QA (Yin et al., 2015; Seo et al., 2016; Shen et al., 2016; Chen et al., 2017; Liu et al., 2017; Hu et al., 2017). In our work, we deal with the answer sentence selection task and adapt the *Attentive Reader* framework proposed in Hermann et al. (2015); Chen et al. (2016) as our base model. While, all these models were trained on question answer pairs, we propose a self-training solution to additionally leverage unsupervised text.

Similarly, there have been works on QG. Traditionally, rule based approaches with post-processing (Woo et al., 2016; Heilman and Smith, 2009, 2010) were the norm in QG. However, recent papers build on neural network approaches such as seq2seq (Du et al., 2017; Tang et al., 2017; Zhou et al., 2017), CNNs and RNNs (Duan et al., 2017) for QG. We also choose the seq2seq paradigm in our work. However, we leverage unsupervised text in contrast to these models.

Finally, some very recent works have concurrently recognized the relationship between QA and QG and have proposed joint training (Tang et al., 2017; Wang et al., 2017) for the two. Our work differs from these as we additionally propose self-training to leverage unlabeled data to improve the two models. Self-training has seldom been used in NLP. Most prominently, they have been used for word sense disambiguation (Yarowsky, 1995), noun learning (Riloff et al., 2003) and recently, AMR parsing and generation (Konstas et al., 2017). However, it has not been explored in this way for QA and QG.

An important decision in the workings of our self-training algorithm was the question selection using curriclum learning. While curriculum learning has seldom been used in NLP, we draw some ideas for curriculum learning from Sachan and Xing (2016a) who conduct a case study of curriculum learning for question answering. However, their work focuses only on QA and not QG.

8 Conclusion

We described self-training algorithms for jointly learning to answer and ask questions while leveraging unlabeled data. We experimented with neural models for question answering and question generation and various careful strategies for question filtering based on curriculum learning and diversity promotion. This led to improved performance for both question answering and question generation on multiple datasets and new state-of-the-art results on *WikiQA* and *TrecQA* datasets.

Acknowledgment

We acknowledge the CMLH fellowship to MS and ONR grant N000141712463 for funding support.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*. ACM, pages 41–48.

Nathaniel Freeman Cantor. 1946. *Dynamics of learning*. Foster and Stewart publishing corporation, Buffalo, NY.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. *arXiv preprint arXiv:1606.02858* .

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. *CoRR* abs/1704.00051.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the ninth workshop on statistical machine translation*. pages 376–380.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. *arXiv preprint arXiv:1705.00106* .

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. pages 866–874.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2005. Bidirectional lstm networks for improved phoneme classification and recognition. *Artificial Neural Networks: Formal Models and Their Applications–ICANN 2005* pages 753–753.

Hua He, Kevin Gimpel, and Jimmy J. Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. pages 1576–1586.

Hua He and Jimmy J. Lin. 2016. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*. pages 937–948.

Michael Heilman. 2011. *Automatic factual question generation from text*. Ph.D. thesis, Carnegie Mellon University.

Michael Heilman and Noah A Smith. 2009. Question generation via overgenerating transformations and ranking. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA LANGUAGE TECHNOLOGIES INST.

Michael Heilman and Noah A Smith. 2010. Good question! statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 609–617.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*. pages 1693–1701.

Minghao Hu, Yuxing Peng, and Xipeng Qiu. 2017. Mnemonic reader for machine comprehension. *CoRR* abs/1705.02798.

Lu Jiang, Deyu Meng, Teruko Mitamura, and Alexander G Hauptmann. 2014. Easy samples first: Self-paced reranking for zero-example multimedia search. In *Proceedings of the ACM International Conference on Multimedia*. ACM, pages 547–556.

Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander G Hauptmann. 2015. Self-paced curriculum learning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '07, pages 177–180.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural amr: Sequence-to-sequence models for parsing and generation. *arXiv preprint arXiv:1704.08381* .

Kai A Krueger and Peter Dayan. 2009. Flexible shaping: How learning in small steps helps. *Cognition* 110(3):380–394.

M Pawan Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*. pages 1189–1197.

VI Levenshtein. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady* 10:707.

Percy S Liang, Dan Klein, and Michael I Jordan. 2008. Agreement-based learning. In *Advances in Neural Information Processing Systems*. pages 913–920.

Chin-Yew Lin. 2004. Rouge: a package for automatic evaluation of summaries. In *Workshop on Text Summarization Branches Out, Post-Conference Workshop of ACL, Barcelona, Spain*.

Rui Liu, Junjie Hu, Wei Wei, Zi Yang, and Eric Nyberg. 2017. Structural embedding of syntactic trees for machine comprehension. *CoRR* abs/1703.00572.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *International Conference on Machine Learning*. pages 1727–1736.

Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *CoRR* abs/1606.03126.

Karthik Narasimhan and Regina Barzilay. 2015. Machine comprehension with discourse relations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*. pages 1253–1262.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268* .

Kamal Nigam and Rayid Ghani. 2000. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the ninth international conference on Information and knowledge management*. ACM, pages 86–93.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. ACL '02, pages 311–318.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pages 1532–1543.

Gail B Peterson. 2004. A day of great illumination: Bf skinner's discovery of shaping. *Journal of the Experimental Analysis of Behavior* 82(3):317–328.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. *CoRR* abs/1606.05250. http://arxiv.org/abs/1606.05250.

Jinfeng Rao, Hua He, and Jimmy Lin. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, New York, NY, USA, CIKM '16, pages 1913–1916. https://doi.org/10.1145/2983323.2983872.

Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 193–203. http://www.aclweb.org/anthology/D13-1020.

Ellen Riloff, Janyce Wiebe, and Theresa Wilson. 2003. Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*. Association for Computational Linguistics, pages 25–32.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685* .

Mrinmaya Sachan, Avinava Dubey, Eric P Xing, and Matthew Richardson. 2015. Learning answer-entailing structures for machine comprehension. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Mrinmaya Sachan, Kumar Dubey, and Eric P. Xing. 2016. Science question answering using instructional materials. In *Proceedings of ACL*.

Mrinmaya Sachan and Eric P. Xing. 2016a. Easy questions first? A case study on curriculum learning for question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Mrinmaya Sachan and Eric P. Xing. 2016b. Machine comprehension using rich semantic representations. In *Proceedings of ACL*.

Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *arXiv preprint arXiv:1602.03609* .

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603* .

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, pages 101–110.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2016. Reasonet: Learning to stop reading in machine comprehension. *CoRR* abs/1609.05284. http://arxiv.org/abs/1609.05284.

Burrhus F Skinner. 1958. Reinforcement today. *American Psychologist* 13(3):94.

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

Duyu Tang, Nan Duan, Tao Qin, and Ming Zhou. 2017. Question answering and question generation as dual tasks. *arXiv preprint arXiv:1706.02027* .

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2017. Enabling efficient question answer retrieval via hyperbolic neural networks. *CoRR* abs/1707.07847. http://arxiv.org/abs/1707.07847.

Bingning Wang, Kang Liu, and Jun Zhao. 2016a. Inner attention based recurrent neural networks for answer selection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2015. Machine comprehension with syntax, frames, and semantics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. volume 2, pages 700–706.

Mengqiu Wang, Noah A Smith, and Teruko Mitamura. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. In *EMNLP-CoNLL*. volume 7, pages 22–32.

Shuohang Wang and Jing Jiang. 2016. A compare-aggregate model for matching text sequences. *arXiv preprint arXiv:1611.01747* .

Tong Wang, Xingdi Yuan, and Adam Trischler. 2017. A joint model for question answering and question generation. *CoRR* abs/1706.01450.

Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. 2016b. Sentence similarity learning by lexical decomposition and composition. *CoRR* abs/1602.07019.

Simon Woo, Zuyao Li, and Jelena Mirkovic. 2016. Good automatic authentication question generation. In *Proceedings of the 9th International Natural Language Generation conference*. pages 203–206.

Yi Yang, Scott Wen-tau Yih, and Chris Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of EMNLP*.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 189–196.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2015. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193* .

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632* .

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329* .

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural question generation from text: A preliminary study. *CoRR* abs/1704.01792. http://arxiv.org/abs/1704.01792.

Xiaojin Zhu. 2005. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison.

The Web as a Knowledge-base for Answering Complex Questions

Alon Talmor
Tel-Aviv University
alontalmor@mail.tau.ac.il

Jonathan Berant
Tel-Aviv University
joberant@cs.tau.ac.il

Abstract

Answering complex questions is a time-consuming activity for humans that requires reasoning and integration of information. Recent work on reading comprehension made headway in answering simple questions, but tackling complex questions is still an ongoing research challenge. Conversely, semantic parsers have been successful at handling compositionality, but only when the information resides in a target knowledge-base. In this paper, we present a novel framework for answering broad and complex questions, assuming answering simple questions is possible using a search engine and a reading comprehension model. We propose to decompose complex questions into a sequence of simple questions, and compute the final answer from the sequence of answers. To illustrate the viability of our approach, we create a new dataset of complex questions, COMPLEXWEBQUESTIONS, and present a model that decomposes questions and interacts with the web to compute an answer. We empirically demonstrate that question decomposition improves performance from 20.8 precision@1 to 27.5 precision@1 on this new dataset.

1 Introduction

Humans often want to answer complex questions that require reasoning over multiple pieces of evidence, e.g., *"From what country is the winner of the Australian Open women's singles 2008?"*. Answering such questions in broad domains can be quite onerous for humans, because it requires searching and integrating information from multiple sources.

Recently, interest in question answering (QA) has surged in the context of reading comprehension (RC), where an answer is sought for a question given one or more documents (Hermann et al., 2015; Joshi et al., 2017; Rajpurkar et al., 2016).

Figure 1: Given a complex questions q, we decompose the question to a sequence of simple questions $q_1, q_2, \ldots$, use a search engine and a QA model to answer the simple questions, from which we compute the final answer a.

Neural models trained over large datasets led to great progress in RC, nearing human-level performance (Wang et al., 2017). However, analysis of models revealed (Jia and Liang, 2017; Chen et al., 2016) that they mostly excel at matching questions to local contexts, but struggle with questions that require reasoning. Moreover, RC assumes documents with the information relevant for the answer are available – but when questions are complex, even retrieving the documents can be difficult.

Conversely, work on QA through semantic parsing has focused primarily on compositionality: questions are translated to compositional programs that encode a sequence of actions for finding the answer in a knowledge-base (KB) (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Artzi and Zettlemoyer, 2013; Krishnamurthy and Mitchell, 2012; Kwiatkowski et al., 2013; Liang et al., 2011). However, this reliance on a manually-curated KB has limited the coverage and applicability of semantic parsers.

In this paper we present a framework for QA that is *broad*, i.e., it does not assume information is in a KB or in retrieved documents, and *compositional*, i.e., to compute an answer we must perform some computation or reasoning. Our thesis is that answering simple questions can be achieved

Proceedings of NAACL-HLT 2018, pages 641–651
New Orleans, Louisiana, June 1 - 6, 2018.

by combining a search engine with a RC model. Thus, answering complex questions can be addressed by decomposing the question into a sequence of simple questions, and computing the answer from the corresponding answers. Figure 1 illustrates this idea. Our model decomposes the question in the figure into a sequence of simple questions, each is submitted to a search engine, and then an answer is extracted from the search result. Once all answers are gathered, a final answer can be computed using symbolic operations such as union and intersection.

To evaluate our framework we need a dataset of complex questions that calls for reasoning over multiple pieces of information. Because an adequate dataset is missing, we created COMPLEXWEBQUESTIONS, a new dataset for complex questions that builds on WEBQUESTIONSSP, a dataset that includes pairs of simple questions and their corresponding SPARQL query. We take SPARQL queries from WEBQUESTIONSSP and automatically create more complex queries that include phenomena such as function composition, conjunctions, superlatives and comparatives. Then, we use Amazon Mechanical Turk (AMT) to generate natural language questions, and obtain a dataset of 34,689 question-answer pairs (and also SPARQL queries that our model ignores). Data analysis shows that examples are diverse and that AMT workers perform substantial paraphrasing of the original machine-generated question.

We propose a model for answering complex questions through question decomposition. Our model uses a sequence-to-sequence architecture (Sutskever et al., 2014) to map utterances to short programs that indicate how to decompose the question and compose the retrieved answers. To obtain supervision for our model, we perform a noisy alignment from machine-generated questions to natural language questions and automatically generate noisy supervision for training.[1]

We evaluate our model on COMPLEXWEBQUESTIONSand find that question decomposition substantially improves precision@1 from 20.8 to 27.5. We find that humans are able to reach 63.0 precision@1 under a limited time budget, leaving ample room for improvement in future work.

To summarize, our main contributions are:

[1]We differ training from question-answer pairs for future work.

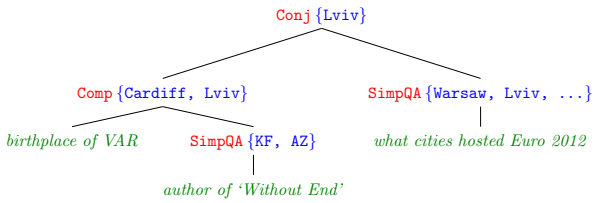


Figure 2: A computation tree for *"What city is the birthplace of the author of 'Without end', and hosted Euro 2012?"*. The leaves are strings, and inner nodes are functions (red) applied to their children to produce answers (blue).

1. A framework for answering complex questions through question decomposition.
2. A sequence-to-sequence model for question decomposition that substantially improves performance.
3. A dataset of 34,689 examples of complex and broad questions, along with answers, web snippets, and SPARQL queries.

Our dataset, COMPLEXWEBQUESTIONS, can be downloaded from http://nlp.cs.tau.ac.il/compwebq and our codebase can be downloaded from https://github.com/alontalmor/WebAsKB.

2 Problem Formulation

Our goal is to learn a model that given a question q and a black box QA model for answering simple questions, SIMPQA$(\cdot)$, produces a *computation tree* t (defined below) that decomposes the question and computes the answer. The model is trained from a set of N question-computation tree pairs $\{q^i, t^i\}_{i=1}^N$ or question-answer pairs $\{q^i, a^i\}_{i=1}^N$.

A computation tree is a tree where leaves are labeled with strings, and inner nodes are labeled with functions. The arguments of a function are its children sub-trees. To compute an answer, or *denotation*, from a tree, we recursively apply the function at the root to its children. More formally, given a tree rooted at node t, labeled by the function f, that has children $c_1(t), \ldots, c_k(t)$, the denotation $[\![t]\!] = f([\![c_1(t)]\!], \ldots, [\![c_k(t)]\!])$ is an arbitrary function applied to the denotations of the root's children. Denotations are computed recursively and the denotation of a string at the leaf is the string itself, i.e., $[\![l]\!] = l$. This is closely related to "semantic functions" in semantic parsing (Berant and Liang, 2015), except that we do not in-

teract with a KB, but rather compute directly over the breadth of the web through a search engine.

Figure 2 provides an example computation tree for our running example. Notice that words at the leaves are not necessarily in the original question, e.g., *"city"* is paraphrased to *"cities"*. More broadly, our framework allows paraphrasing questions in any way that is helpful for the function SIMPQA$(\cdot)$. Paraphrasing for better interaction with a QA model has been recently suggested by Buck et al. (2017) and Nogueira and Cho (2016).

We defined the function SIMPQA$(\cdot)$ for answering simple questions, but in fact it comprises two components in this work. First, the question is submitted to a search engine that retrieves a list of web snippets. Next, a RC model extracts the answer from the snippets. While it is possible to train the RC model jointly with question decomposition, in this work we pre-train it separately, and later treat it as a black box.

The expressivity of our QA model is determined by the functions used, which we turn to next.

3 Formal Language

Functions in our formal language take arguments and return values that can be strings (when decomposing or re-phrasing the question), sets of strings, or sets of numbers. Our set of functions includes:

1. SIMPQA$(\cdot)$: Model for answering simple questions, which takes a string argument and returns a set of strings or numbers as answer.
2. COMP$(\cdot,\cdot)$: This function takes a string containing one unique variable VAR, and a set of answers. E.g., in Figure 2 the first argument is *"birthplace of VAR"*, and the second argument is "{KEN FOLLETT, ADAM ZAGAJEWSKI}". The function replaces the variable with each answer string representation and returns their union. Formally, COMP$(q, \mathcal{A}) = \cup_{a \in \mathcal{A}}$SIMPQA$(q/a)$, where q/a denotes the string produced when replacing VAR in q with a. This is similar to function composition in CCG (Steedman, 2000), or a join operation in λ-DCS (Liang, 2013), where the string is a function applied to previously-computed values.
3. CONJ$(\cdot,\cdot)$: takes two sets and returns their intersection. Other set operations can be defined analogously. As syntactic sugar, we allow CONJ$(\cdot)$ to take strings as input, which means that we run SIMPQA$(\cdot)$ to obtain a set and then perform intersection. The root node in Figure 2 illustrates an application of CONJ.
4. ADD$(\cdot,\cdot)$: takes two singleton sets of numbers and returns a set with their addition. Similar functions can be defined analogously. While we support mathematical operations, they were not required in our dataset.

Other logical operations In semantic parsing superlative and comparative questions like *"What is the highest European mountain?"* or *"What European mountains are higher than Mont Blanc?"* are answered by joining the set of European mountains with their elevation. While we could add such functions to the formal language, answering such questions from the web is cumbersome: we would have to extract a list of entities and a numerical value for each. Instead, we handle such constructions using SIMPQA directly, assuming they are mentioned verbatim on some web document.

Similarly, negation questions (*"What countries are not in the OECD?"*) are difficult to handle when working against a search engine only, as this is an open world setup and we do not hold a closed set of countries over which we can perform set subtraction.

In future work, we plan to interface with tables (Pasupat and Liang, 2015) and KBs (Zhong et al., 2017). This will allow us to perform set operations over well-defined sets, and handle in a compositional manner superlatives and comparatives.

4 Dataset

Evaluating our framework requires a dataset of broad and complex questions that examine the importance of question decomposition. While many QA datasets have been developed recently (Yang et al., 2015; Rajpurkar et al., 2016; Hewlett et al., 2016; Nguyen et al., 2016; Onishi et al., 2016; Hill et al., 2015; Welbl et al., 2017), they lack a focus on the importance of question decomposition.

Most RC datasets contain simple questions that can be answered from a short input document. Recently, TRIVIAQA (Joshi et al., 2017) presented a larger portion of complex questions, but still most do not require reasoning. Moreover, the focus of TRIVIAQA is on answer extraction from documents that are given. We, conversely, highlight question decomposition for finding the relevant documents. Put differently, RC is complementary to question decomposition and can be used as part of the implementation of SIMPQA. In Sec-

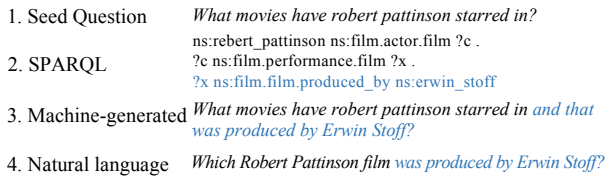


Figure 3: Overview of data collection process.

tion 6 we demonstrate that question decomposition is useful for two different RC approaches.

4.1 Dataset collection

To generate complex questions we use the dataset WEBQUESTIONSSP (Yih et al., 2016), which contains 4,737 questions paired with SPARQL queries for Freebase (Bollacker et al., 2008). Questions are broad but simple. Thus, we sample question-query pairs, automatically create more complex SPARQL queries, generate automatically questions that are understandable to AMT workers, and then have them paraphrase those into natural language (similar to Wang et al. (2015)). We compute answers by executing complex SPARQL queries against Freebase, and obtain broad and complex questions. Figure 3 provides an example for this procedure, and we elaborate next.

Generating SPARQL queries Given a SPARQL query r, we create four types of more complex queries: conjunctions, superlatives, comparatives, and compositions. Table 1 gives the exact rules for generation. For conjunctions, superlatives, and comparatives, we identify queries in WEBQUESTIONSSP whose denotation is a set $\mathcal{A}, |\mathcal{A}| \geq 2$, and generate a new query r' whose denotation is a strict subset $\mathcal{A}', \mathcal{A}' \subset \mathcal{A}, \mathcal{A}' \neq \phi$. For conjunctions this is done by traversing the KB and looking for SPARQL triplets that can be added and will yield a valid set $\mathcal{A}'$. For comparatives and superlatives we find a numerical property common to all $a \in \mathcal{A}$, and add a triplet and restrictor to r accordingly. For compositions, we find an entity e in r, and replace e with a variable y and add to r a triplet such that the denotation of that triplet is $\{e\}$.

Machine-generated (MG) questions To have AMT workers paraphrase SPARQL queries into natural language, we need to present them in an understandable form. Therefore, we automatically generate a question they can paraphrase. When we generate new SPARQL queries, new predicates are added to the query (Table 1). We manually annotated 687 templates mapping KB predicates to text for different compositionality types (with 462 unique KB predicates), and use those templates to modify the original WebQuestionsSP question according to the meaning of the generated SPARQL query. E.g., the template for $?x$ `ns:book.author.works_written obj` is *"the author who wrote OBJ"*. For brevity, we provide the details in the supplementary material.

Question Rephrasing We used AMT workers to paraphrase MG questions into natural language (NL). Each question was paraphrased by one AMT worker and validated by 1-2 other workers. To generate diversity, workers got a bonus if the edit distance of a paraphrase was high compared to the MG question. A total of 200 workers were involved, and 34,689 examples were produced with an average cost of 0.11$ per question. Table 1 gives an example for each compositionality type.

A drawback of our method for generating data is that because queries are generated automatically the question distribution is artificial from a semantic perspective. Still, developing models that are capable of reasoning is an important direction for natural language understanding and COMPLEXWEBQUESTIONS provides an opportunity to develop and evaluate such models.

To summarize, each of our examples contains a question, an answer, a SPARQL query (that our models ignore), and all web snippets harvested by our model when attempting to answer the question. This renders COMPLEXWEBQUESTIONS useful for both the RC and semantic parsing communities.

4.2 Dataset analysis

COMPLEXWEBQUESTIONS builds on the WEBQUESTIONS (Berant et al., 2013). Questions in WEBQUESTIONS are usually about properties of entities (*"What is the capital of France?"*), often with some filter for the semantic type of the answer (*"Which director"*, *"What city"*). WEBQUESTIONS also contains questions that refer to events with multiple entities (*"Who did Brad Pitt play in Troy?"*). COMPLEXWEBQUESTIONS contains all these semantic phenomena, but we add four compositionality types by generating composition questions (45% of the times), conjunctions (45%), superlatives (5%) and comparatives (5%).

Composit.	Complex SPARQL query r'	Example (natural language)
CONJ.	r. $?x$ pred_1 obj. **or** r. $?x$ pred_1 $?c$. $?c$ pred_2 obj.	*"What films star Taylor Lautner and have costume designs by Nina Proctor?"*
SUPER.	r. $?x$ pred_1 $?n$.ORDER BY DESC($?n$) LIMIT 1	*"Which school that Sir Ernest Rutherford attended has the latest founding date?"*
COMPAR.	r. $?x$ pred_1 $?n$. FILTER $?n < V$	*"Which of the countries bordering Mexico have an army size of less than 1050?"*
COMP.	$r[e/y]$. $?y$ pred_1 obj.	*"Where is the end of the river that originates in Shannon Pot?"*

Table 1: Rules for generating a complex query r' from a query r ('.' in SPARQL corresponds to logical and). The query r returns the variable $?x$, and contains an entity e. We denote by $r[e/y]$ the replacement of the entity e with a variable $?y$. pred_1 and pred_2 are any KB predicates, obj is any KB entity, V is a numerical value, and $?c$ is a variable of a CVT type in Freebase which refers to events. The last column provides an example for a NL question for each type.

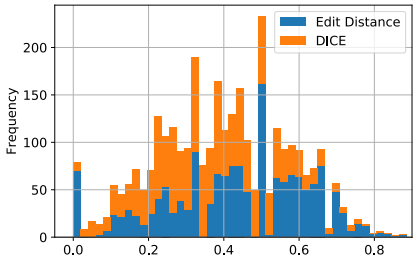


Figure 4: MG and NL questions similarity with normalized edit-distance, and the DICE coefficient (bars are stacked).

Paraphrasing To generate rich paraphrases, we gave a bonus to workers that substantially modified MG questions. To check whether this worked, we measured surface similarity between MG and NL questions, and examined the similarity. Using normalized edit-distance and the DICE coefficient, we found that NL questions are different from MG questions and that the similarity distribution has wide support (Figure 4).

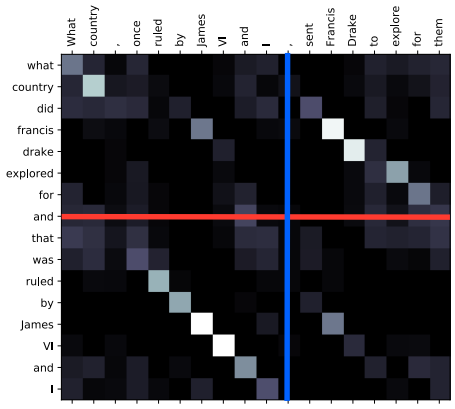


Figure 5: Heat map for similarity matrix between a MG and NL question. The red line indicates a known MG split point. The blue line is the approximated NL split point.

We created a heuristic for approximating the amount of word re-ordering performed by AMT workers. For every question, we constructed a matrix A, where A_{ij} is the similarity between token i in the MG question and token j in the NL question. Similarity is 1 if lemmas match, or cosine similarity according to GloVe embeddings (Pennington et al., 2014), when above a threshold, and 0 otherwise. The matrix A allows us to estimate whether parts of the MG question were re-ordered when paraphrased to NL (details in supplementary material). We find that in 44.7% of the conjunction questions and 13.2% of the composition questions, word re-ordering happened, illustrating that substantial changes to the MG question have been made. Figure 5 illustrates the matrix A for a pair of questions with re-ordering.

Qualitative analysis We randomly sampled 100 examples from the development set and manually identified prevalent phenomena in the data. We present these types in Table 2 along with their frequency. In 18% of the examples a conjunct in the MG question becomes a modifier of a wh-word in the NL question (WH-MODIFIER). In 22% substantial word re-ordering of the MG questions occurred, and in 42% a minor word re-ordering occurred (*"number of building floors is 50"* paraphrased as *"has 50 floors"*). AMT workers used a synonym in 54% of the examples, they omitted words in 27% of the examples and they added new lexical material in 29%.

To obtain intuition for operations that will be useful in our model, we analyzed the 100 examples for the types of operations that should be applied to the NL question during question decomposition. We found that splitting the NL question is insufficient, and that in 53% of the cases a word in the NL question needs to be copied to multiple questions after decomposition (row 3 in Table 3).

Type	MG question	NL question	%
WH-MODIFIER	*what movies does leo howard play in and that is 113.0 minutes long?*	*Which Leo Howard movie lasts 113 minutes?*	18%
MAJOR REORD.	*Where did the actor that played in the film Hancock 2 go to high school?*	*What high school did the actor go to who was in the movie Hancock 2?*	22%
MINOR REORD.	*what to do and see in vienna austria and the number of building floors is 50?*	*What building in Vienna, Austria has 50 floors?*	42%
SYNONYM	*where does the body of water under Kineshma Bridge start?*	*Where does the body of water under Kineshma Bridge originate?*	54%
SKIP WORD	*what movies did miley cyrus play in and involves organization Cirkus?*	*What movie featured Miley Cyrus and involved Cirkus?*	27%
ADD WORD	*what to do if you have one day in bangkok and the place is an amusement park that opened earliest?*	*Which amusement park, that happens to be the one that opened earliest, should you visit if you have only one day to spend in Bangkok?*	29%

Table 2: Examples and frequency of prevalent phenomena in the NL questions for a manually analyzed subset (see text).

Moreover, words that did not appear in the MG question need to be added in 39% of the cases, and words need to be deleted in 28% of the examples.

5 Model and Learning

We would like to develop a model that translates questions into arbitrary computation trees with arbitrary text at the tree leaves. However, this requires training from denotations using methods such as maximum marginal likelihood or reinforcement learning (Guu et al., 2017) that are difficult to optimize. Moreover, such approaches involve issuing large amounts of queries to a search engine at training time, incurring high costs and slowing down training.

Instead, we develop a simple approach in this paper. We consider a subset of all possible computation trees that allows us to automatically generate noisy full supervision. In what follows, we describe the subset of computation trees considered and their representation, a method for automatically generating noisy supervision, and a pointer network model for decoding.

Representation We represent computation trees as a sequence of tokens, and consider trees with at most one compositional operation. We denote a sequence of question tokens $q_{i:j} = (q_i, \ldots, q_j)$, and the decoded sequence by z. We consider the following token sequences (see Table 3):

1. `SimpQA`: The function SIMPQA is applied to the question q without paraphrasing. In prefix notation this is the tree $\text{SIMPQA}(q)$.
2. `Comp` i j: This sequence of tokens corresponds to the following computation tree: $\text{COMP}(q_{1:i-1} \circ \texttt{VAR} \circ q_{j+1:|q|}, \text{SIMPQA}(q_{i:j}))$, where $\circ$ is the concatenation operator. This is used for questions where a substring is answered by SIMPQA and the answers replace a variable before computing a final answer.
3. `Conj` i j: This sequence of tokens corresponds to the computation tree $\text{CONJ}(\text{SIMPQA}(q_{0:i-1}), \text{SIMPQA}(q_j \circ q_{i:|q|}))$. The idea is that conjunction can be answered by splitting the question in a single point, where one token is copied to the second part as well (*"film"* in Table 3). If nothing needs to be copied, then $j = -1$.

This representation supports one compositional operation, and a single copying operation is allowed without any re-phrasing. In future work, we plan to develop a more general representation, which will require training from denotations.

Supervision Training from denotations is difficult as it involves querying a search engine frequently, which is expensive. Therefore, we take advantage of the the original SPARQL queries and MG questions to generate noisy programs for composition and conjunction questions. Note that these noisy programs are only used as supervision to avoid the costly process of manual annotation, but the model itself does not assume SPARQL queries in any way.

We generate noisy programs from SPARQL queries in the following manner: First, we automatically identify composition and conjunction questions. Because we generated the MG question, we can exactly identify the split points (i, j in composition questions and i in conjunction questions) in the MG question. Then, we use a rule-based algorithm that takes the alignment matrix A (Section 4), and approximates the split points in the NL question and the index j to copy in conjunction questions. The red line in Figure 5 corresponds to the known split point in the MG question, and the blue one is the approximated split point in the NL question. The details of this rule-

Program	Question	Split
`SimpQA`	*"What building in Vienna, Austria has 50 floors"*	-
`Comp 5 9`	*"Where is the birthplace of the writer of Standup Shakespeare"*	*"Where is the birthplace of VAR"* *"the writer of Standup Shakespeare"*
`Conj 5 1`	*"What film featured Taylor Swift and was directed by Deborah Aquila"*	*"What film featured Taylor Swift"* *"film and was directed by Deborah Aquila"*

Table 3: Examples for the types of computation trees that can be decoded by our model.

based algorithm are in the supplementary material.

Thus, we obtain noisy supervision for all composition and conjunction questions and can train a model that translates questions q to representations $z = z_1\ z_2\ z_3$, where $z_1 \in \{\texttt{Comp}, \texttt{Conj}\}$ and z_2, z_3 are integer indices.

Pointer network The representation z points to indices in the input, and thus pointer networks (Vinyals et al., 2015) are a sensible choice. Because we also need to decode the tokens COMP and CONJ, we use "augmented pointer networks", (Zhong et al., 2017): For every question q, an augmented question $\hat{q}$ is created by appending the tokens "COMP CONJ" to q. This allows us to decode the representation z with one pointer network that at each decoding step points to one token in the augmented question. We encode $\hat{q}$ with a one-layer GRU (Cho et al., 2014), and decode z with a one-layer GRU with attention as in Jia and Liang (2016). The only difference is that we decode tokens from the augmented question $\hat{q}$ rather than from a fixed vocabulary.

We train the model with token-level cross-entropy loss, minimizing $\sum_j \log p_\theta(z_j|x, z_{1:j-1})$. Parameters θ include the GRU encoder and decoder, and embeddings for unknown tokens (that are not in pre-trained GloVe embeddings (Pennington et al., 2014)).

The trained model decodes COMP and CONJ representations, but sometimes using SIMPQA(q) without decomposition is better. To handle such cases we do the following: We assume that we always have access to a score for every answer, provided by the final invocation of SIMPQA (in CONJ questions this score is the maximum of the scores given by SIMPQA for the two conjuncts), and use the following rule to decide if to use the decoded representation z or SIMPQA(q). Given the scores for answers given by z and the scores given by SIMPQA(q), we return the single answer that has the highest score. The intuition is that the confidence provided by the scores of SIMPQA is correlated with answer correctness. In future work we will train directly from denotations and will handle all logical functions in a uniform manner.

6 Experiments

In this section, we aim to examine whether question decomposition can empirically improve performance of QA models over complex questions.

Experimental setup We used 80% of the examples in COMPLEXWEBQUESTIONS for training, 10% for development, and 10% for test, training the pointer network on 24,708 composition and conjunction examples. The hidden state dimension of the pointer network is 512, and we used Adagrad (Duchi et al., 2010) combined with L_2 regularization and a dropout rate of 0.25. We initialize 50-dimensional word embeddings using GloVe and learn embeddings for missing words.

Simple QA model As our SIMPQA function, we download the web-based QA model of Talmor et al. (2017). This model sends the question to Google's search engine and extracts a distribution over answers from the top-100 web snippets using manually-engineered features. We re-train the model on our data with one new feature: for every question q and candidate answer mention in a snippet, we run RASOR, a RC model by lee et al. (2016), and add the output logit score as a feature. We found that combining the web-facing model of Talmor et al. (2017) and RASOR, resulted in improved performance.

Evaluation For evaluation, we measure precision@1 (p@1), i.e., whether the highest scoring answer returned string-matches one of the correct answers (while answers are sets, 70% of the questions have a single answer, and the average size of the answer set is 2.3).

We evaluate the following models and oracles:

1. SIMPQA: running SIMPQA on the entire question, i.e., without decomposition.
2. SPLITQA: Our main model that answers complex questions by decomposition.
3. SPLITQAORACLE: An *oracle* model that chooses whether to perform question decom-

System	Dev.	Test
SIMPQA	20.4	20.8
SPLITQA	29.0	27.5
SPLITQAORACLE	34.0	33.7
RCQA	18.7	18.6
SPLITRCQA	21.5	22.0
GOOGLEBOX	2.5	-
HUMAN	63.0	-

Table 4: precision@1 results on the development set and test set for COMPLEXWEBQUESTIONS.

position or use SIMPQA in hindsight based on what performs better.

4. RCQA: This is identical to SIMPQA, except that we replace the RC model from Talmor et al. (2017) with the the RC model DOCQA (Clark and Gardner, 2017), whose performance is comparable to state-of-the-art on TRIVIAQA.
5. SPLITRCQA: This is identical to SPLITQA, except that we replace the RC model from Talmor et al. (2017) with DOCQA.
6. GOOGLEBOX: We sample 100 random development set questions and check whether Google returns a box that contains one of the correct answers.
7. HUMAN: We sample 100 random development set questions and manually answer the questions with Google's search engine, including all available information. We limit the amount of time allowed for answering to 4 minutes.

Table 4 presents the results on the development and test sets. SIMPQA, which does not decompose questions obtained 20.8 p@1, while by performing question decomposition we substantially improve performance to 27.5 p@1. An upper bound with perfect knowledge on when to decompose is given by SPLITQAORACLE at 33.7 p@1.

RCQA obtained lower performance SIMPQA, as it was trained on data from a different distribution. More importantly SPLITRCQA outperforms RCQA by 3.4 points, illustrating that this RC model also benefits from question decomposition, despite the fact that it was not created with question decomposition in mind. This shows the importance of question decomposition for retrieving documents from which an RC model can extract answers. GOOGLEBOX finds a correct answer in 2.5% of the cases, showing that complex questions are challenging for search engines.

To conclude, we demonstrated that question decomposition substantially improves performance on answering complex questions using two independent RC models.

Analysis We estimate human performance (HUMAN) at 63.0 p@1. We find that answering complex questions takes roughly 1.3 minutes on average. For questions we were unable to answer, we found that in 27% the answer was correct but exact string match with the gold answers failed; in 23.1% the time required to compute the answer was beyond our capabilities; for 15.4% we could not find an answer on the web; 11.5% were of ambiguous nature; 11.5% involved paraphrasing errors of AMT workers; and an additional 11.5% did not contain a correct gold answer.

SPLITQA decides if to decompose questions or not based on the confidence of SIMPQA. In 61% of the questions the model chooses to decompose the question, and in the rest it sends the question as-is to the search engine. If one of the strategies (decomposition vs. no decomposition) works, our model chooses that right one in 86% of the cases. Moreover, in 71% of these answerable questions, only one strategy yields a correct answer.

We evaluate the ability of the pointer network to mimic our labeling heuristic on the development set. We find that the model outputs the exact correct output sequence 60.9% of the time, and allowing errors of one word to the left and right (this often does not change the final output) accuracy is at 77.1%. Token-level accuracy is 83.0% and allowing one-word errors 89.7%. This shows that SPLITQA learned to identify decomposition points in the questions. We also observed that often SPLITQA produced decomposition points that are better than the heuristic, e.g., for *"What is the place of birth for the lyricist of Roman Holiday"*, SPLITQA produced *"the lyricist of Roman Holiday"*, but the heuristic produced *"the place of birth for the lyricist of Roman Holiday"*. Additional examples of SPLITQA question decompositions are provided in Table 5.

ComplexQuestions To further examine the ability of web-based QA models, we run an experiment against COMPLEXQUESTIONS (Bao et al., 2016), a small dataset of question-answer pairs designed for semantic parsing against Freebase.

We ran SIMPQA on this dataset (Table 6) and obtained 38.6 F_1 (the official metric), slightly lower than COMPQ, the best system, which op-

Question	Split-1	Split-2
"Find the actress who played Hailey Rogers, what label is she signed to"	*"the actress who played Hailey Rogers"*	*"Find VAR, what label is she signed to"*
"What are the colors of the sports team whose arena stadium is the AT&T Stadium"	*"the sports team whose arena stadium is the AT&T Stadium"*	*"What are the colors of VAR"*
"What amusement park is located in Madrid Spain and includes the stunt fall ride"	*"What amusement park is located in Madrid Spain and"*	*"park includes the stunt fall ride"*
"Which university whose mascot is The Trojan did Derek Fisher attend"	*"Which university whose mascot is The Trojan did"*	*"university Derek Fisher attend"*

Table 5: Examples for question decompositions from SPLITQA.

System	Dev. F_1	Test F_1
SIMPQA	40.7	38.6
SPLITQARULE	43.1	39.7
SPLITQARULE++	46.9	-
COMPQ	-	**40.9**

Table 6: F_1 results for COMPLEXQUESTIONS.

erates directly against Freebase. [2] By analyzing the training data, we found that we can decompose COMP questions with a rule that splits the question when the words *"when"* or *"during"* appear, e.g., *"Who was vice president when JFK was president?"*.[3] We decomposed questions with this rule and obtained 39.7 F_1 (SPLITQARULE). Analyzing the development set errors, we found that occasionally SPLITQARULE returns a correct answer that fails to string-match with the gold answer. By manually fixing these cases, our development set F_1 reaches 46.9 (SPLITQARULE++). Note that COMPQ does not suffer from any string matching issue, as it operates directly against the Freebase KB and thus is guaranteed to output the answer in the correct form. This short experiment shows that a web-based QA model can rival a semantic parser that works against a KB, and that simple question decomposition is beneficial and leads to results comparable to state-of-the-art.

7 Related work

This work is related to a body of work in semantic parsing and RC, in particular to datasets that focus on complex questions such as TRIVIAQA (Joshi et al., 2017), WIKIHOP (Welbl et al., 2017) and RACE (Lai et al., 2017). Our distinction is in proposing a framework for complex QA that focuses on question decomposition.

Our work is related to Chen et al. (2017) and Watanabe et al. (2017), who combined retrieval and answer extraction on a large set of documents. We work against the entire web, and propose question decomposition for finding information.

This work is also closely related to Dunn et al. (2017) and Buck et al. (2017): we start with questions directly and do not assume documents are given. Buck et al. (2017) also learn to phrase questions given a black box QA model, but while they focus on paraphrasing, we address decomposition.

Another important related research direction is Iyyer et al. (2016), who answered complex questions by decomposing them. However, they used crowdsourcing to obtain direct supervision for the gold decomposition, while we do not assume such supervision. Moreover, they work against web tables, while we interact with a search engine against the entire web.

8 Conclusion

In this paper we propose a new framework for answering complex questions that is based on question decomposition and interaction with the web. We develop a model under this framework and demonstrate it improves complex QA performance on two datasets and using two RC models. We also release a new dataset, COMPLEXWEBQUESTIONS, including questions, SPARQL programs, answers, and web snippets harvested by our model. We believe this dataset will serve the QA and semantic parsing communities, drive research on compositionality, and push the community to work on holistic solutions for QA.

In future work, we plan to train our model directly from weak supervision, i.e., denotations, and to extract information not only from the web, but also from structured information sources such as web tables and KBs.

Acknowledgements

We thank Jonatahn Herzig, Ni Lao, and the anonymous reviewers for their constructive feedback. This work was supported by the Samsung runway project and the Israel Science Foundation, grant 942/16.

[2] By adding the output logit from RASOR, we improved test F_1 from 32.6, as reported by Talmor et al. (2017), to 38.6.

[3] The data is too small to train our decomposition model.

References

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

J. Bao, N. Duan, Z. Yan, M. Zhou, and T. Zhao. 2016. Constraint-based question answering with knowledge graph. In *International Conference on Computational Linguistics (COLING)*.

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

J. Berant and P. Liang. 2015. Imitation learning of agenda-based semantic parsers. *Transactions of the Association for Computational Linguistics (TACL)* 3:545–558.

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

C. Buck, J. Bulian, M. Ciaramita, A. Gesmundo, N. Houlsby, W. Gajewski, and W. Wang. 2017. Ask the right questions: Active question reformulation with reinforcement learning. *arXiv preprint arXiv:1705.07830* .

D. Chen, J. Bolton, and C. D. Manning. 2016. A thorough examination of the CNN / Daily Mail reading comprehension task. In *Association for Computational Linguistics (ACL)*.

D. Chen, A. Fisch, J. Weston, and A. Bordes. 2017. Reading Wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051* .

K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* .

C. Clark and M. Gardner. 2017. Simple and effective multi-paragraph reading comprehension. *arXiv preprint arXiv:1710.10723* .

J. Duchi, E. Hazan, and Y. Singer. 2010. Adaptive subgradient methods for online learning and stochastic optimization. In *Conference on Learning Theory (COLT)*.

M. Dunn, , L. Sagun, M. Higgins, U. Guney, V. Cirik, and K. Cho. 2017. SearchQA: A new Q&A dataset augmented with context from a search engine. *arXiv* .

K. Guu, P. Pasupat, E. Z. Liu, and P. Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Association for Computational Linguistics (ACL)*.

K. M. Hermann, T. Koisk, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*.

D. Hewlett, A. Lacoste, L. Jones, I. Polosukhin, A. Fandrianto, J. Han, M. Kelcey, and D. Berthelot. 2016. Wikireading: A novel large-scale language understanding task over Wikipedia. In *Association for Computational Linguistics (ACL)*.

F. Hill, A. Bordes, S. Chopra, and J. Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. In *International Conference on Learning Representations (ICLR)*.

M. Iyyer, W. Yih, and M. Chang. 2016. Answering complicated question intents expressed in decomposed question sequences. *CoRR* 0.

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

R. Jia and P. Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Empirical Methods in Natural Language Processing (EMNLP)*.

M. Joshi, E. Choi, D. Weld, and L. Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Association for Computational Linguistics (ACL)*.

J. Krishnamurthy and T. Mitchell. 2012. Weakly supervised training of semantic parsers. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*. pages 754–765.

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

G. Lai, Q. Xie, H. Liu, Y. Yang, and E. Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683* .

K. lee, M. Lewis, and L. Zettlemoyer. 2016. Global neural CCG parsing with optimality guarantees. In *Empirical Methods in Natural Language Processing (EMNLP)*.

P. Liang. 2013. Lambda dependency-based compositional semantics. *arXiv preprint arXiv:1309.4408* .

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

T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *Workshop on Cognitive Computing at NIPS*.

R. Nogueira and K. Cho. 2016. End-to-end goal-driven web navigation. In *Advances in Neural Information Processing Systems (NIPS)*.

T. Onishi, H. Wang, M. Bansal, K. Gimpel, and D. McAllester. 2016. Who did what: A large-scale person-centered cloze dataset. In *Empirical Methods in Natural Language Processing (EMNLP)*.

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

J. Pennington, R. Socher, and C. D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*.

M. Steedman. 2000. *The Syntactic Process*. MIT Press.

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

A. Talmor, M. Geva, and J. Berant. 2017. Evaluating semantic parsing against a simple web-based question answering model. In **SEM*.

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

W. Wang, N. Yang, F. Wei, B. Chang, and M. Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Association for Computational Linguistics (ACL)*.

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

Y. Watanabe, B. Dhingra, and R. Salakhutdinov. 2017. Question answering from unstructured text by retrieval and comprehension. *arXiv preprint arXiv:1703.08885* .

J. Welbl, P. Stenetorp, and S. Riedel. 2017. Constructing datasets for multi-hop reading comprehension across documents. *arXiv preprint arXiv:1710.06481* .

Y. Yang, W. Yih, and C. Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 2013–2018.

W. Yih, M. Richardson, C. Meek, M. Chang, and J. Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Association for Computational Linguistics (ACL)*.

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

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

V. Zhong, C. Xiong, and R. Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103* .

A Meaning-based Statistical English Math Word Problem Solver

Chao-Chun Liang, Yu-Shiang Wong, Yi-Chung Lin and Keh-Yih Su
Institute of Information Science, Academia Sinica, Taiwan
{ccliang, yushiangwtw, lyc, kysu}@iis.sinica.edu.tw

Abstract

We introduce *MeSys*, a *meaning-based* approach, for solving English math word problems (MWPs) via *understanding and reasoning* in this paper. It first analyzes the text, transforms both body and question parts into their corresponding logic forms, and then performs inference on them. The associated context of each quantity is represented with proposed *role-tags* (e.g., *nsubj*, *verb*, etc.), which provides the flexibility for annotating an extracted *math quantity* with its associated context information (i.e., the *physical meaning* of this quantity). Statistical models are proposed to select the operator and operands. A noisy dataset is designed to assess if a solver solves MWPs mainly via understanding or mechanical pattern matching. Experimental results show that our approach outperforms existing systems on both benchmark datasets and the noisy dataset, which demonstrates that the proposed approach understands the meaning of each quantity in the text more.

1 Introduction

The *math word problem* (MWP) (see Figure 1) is frequently chosen to study natural language understanding and simulate human problem solving (Bakman, 2007; Hosseini et al., 2014; Liang et al., 2016) for the following reasons: (1) the answer to the MWP cannot be simply extracted by performing keyword/pattern matching. It thus shows the merit of understanding and inference.

Math Word Problem
Mike takes 88 minutes to walk to school. If he rides a bicycle to school, it would save him 64 minutes. How much time did Mike save?
Solution
88 – 64 = 22

Figure 1: An example of math word problem.

(2) An MWP usually possesses less complicated syntax and requires less amount of domain knowledge, so the researchers can focus on the task of understanding and reasoning. (3) The body part of MWP that provides the given information for solving the problem consists of only a few sentences. The understanding and reasoning procedures thus could be more efficiently checked. (4) The MWP solver has its own applications such as *Computer Math Tutor* (for students in primary school) and *Helper for Math in Daily Life* (for adults who are not good in solving mathematics related real problems).

According to the approaches used to identify entities, quantities, and to select operations and operands, previous MWP solvers can be classified into: (1) Rule-based approaches (Mukherjee and Garain, 2008[1]; Hosseini et al., 2014), which make all related decisions based on a set of rules; (2) Purely statistics-based approaches (Kushman et al., 2014; Roy et al., 2015; Zhou et al., 2015; Upadhyay et al., 2016), in which all related decisions are done via a statistical classifier; (3) DNN-based approaches (Ling et al., 2017; Wang et al., 2017), which map the given text into the corresponding math operation/equation via a DNN; and (4) Mixed approaches, which identify entities and quantities with rules, yet, decide operands and operations via statistical/DNN classifiers. This category can be further divided into two subtypes: (a) Without understanding (Roy and Roth, 2015; Koncel-Kedziorski et al., 2015; Huang et al., 2017; Shrivastava et al., 2017), which does not check the entity-attribute consistency between each quantity and the target of the given question; and (b) With understanding (Lin et al., 2015; Mitra and Baral, 2016; Roy and Roth, 2017), which also checks the entity-attribute consistency while solving the problem.

[1] It is a survey paper which reviews most of the rule-based approaches before 2008.

Proceedings of NAACL-HLT 2018, pages 652–662
New Orleans, Louisiana, June 1 - 6, 2018.

However, a widely covered rule-set is difficult to construct for the rule-based approach. Also, it is awkward in resolving ambiguity problem. In contrast, the performance of purely statistics-based approaches deteriorates significantly when the MWP includes either irrelevant information or information gaps (Hosseini et al., 2014), as it is solved without first understanding the meaning.

For the category (4a), since the physical meaning is only implicitly utilized and the result is not generated via inference, it would be difficult to explain how the answer is obtained in a human comprehensible way. Therefore, the categories (2), (3) and (4a) belong to the less favored direct translation approach[2] (Pape, 2004).

In contrast, the approaches of (4b) can avoid the problems mentioned above. However, among them, Mitra and Baral (2016) merely handled *Addition* and *Subtraction*. Only the *meaning-based framework* proposed by Lin et al. (2015) can handle general MWPs via understanding and reasoning. Therefore, it is possible to explain how the answer is obtained in a human comprehensible way (Huang et al., 2015). However, although their design looks promising, only a few Chinese MWPs had been tested and performance was not evaluated. Accordingly, it is hard to make a fair comparison between their approach and other state-of-the-art methods. In addition, in their prototype system, the desired operands of arithmetic operations are identified with predefined lexico-syntactic patterns and ad-hoc rules. Reusing the patterns/rules designed for Chinese in another language is thus difficult even if it is possible.

In this paper, we adopt the framework proposed by Lin et al. (2015) to solve English MWPs (for its potential in solving difficult/complex MWPs and providing more human comprehensible explanations). Additionally, we make the following improvements: (1) A new statistical model is proposed to select operands for arithmetic operations, and its model parameters can be automatically learnt via weakly supervised learning (Artzi and Zettlemoyer, 2013). (2) A new informative and robust feature-set is proposed to select the desired arithmetic operation. (3) We show the proposed approach significantly outperforms other existing systems on the common benchmark datasets reported in the literature. (4) A noisy dataset with more irrelevant quantities in MWPs is created and released. It could be used to check if an approach really understands what a given MWP looks for. (5) An experiment is conducted to compare various approaches on this new dataset. The superior performance of our system demonstrates that the proposed meaning-based approach has good potential in handling difficult/complex MWPs.

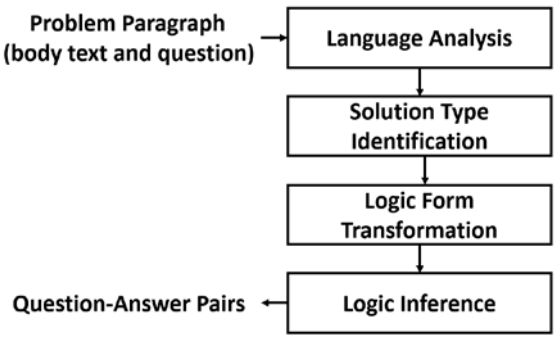


Figure 2: The diagram of *MeSys* framework

2 System Description

The adopted meaning-based framework (Lin et al., 2015) is a pipeline with following four stages (see Figure 2): (1) *Language Analysis*, (2) *Solution Type Identification*, (3) *Logic Form Transformation* and (4) *Logic Inference*. We use the Stanford CoreNLP suite (Manning et al., 2014) as the language analysis module. The other three modules are briefly described below. Last, we adopt the *weakly supervised learning* (Artzi and Zettlemoyer, 2013; Kushman et al., 2014) to automatically learn the model parameters without manually annotating each MWP with the adopted solution type and selected operands benchmark.

2.1 Solution Type Identification (STI)

After language analysis, each MWP is assigned with a specific solution type (such as *Addition*, *Multiplication*, etc.) which indicates the stereotype math operation pattern that should be adopted to solve this problem. We classify the English MWPs released by Hosseini et al. (2014) and Roy and Roth (2015) into 6 different types: *Addition*, *Subtraction*, *Multiplication*, *Division*, *Sum* and *TVQ-F*[3]. An SVM (Chang and Lin, 2011) is used to identify the solution type with 26 features. Most of them are derived from some important properties associated with each quantity.

[2] According to (Pape, 2004), the meaning-based approach of solving MWPs achieves the best performance among various behaviors adopted by middle school children.

[3] *TVQ-F* means to get the final state of a Time-Variant-Quantity that involves both Addition and Subtraction.

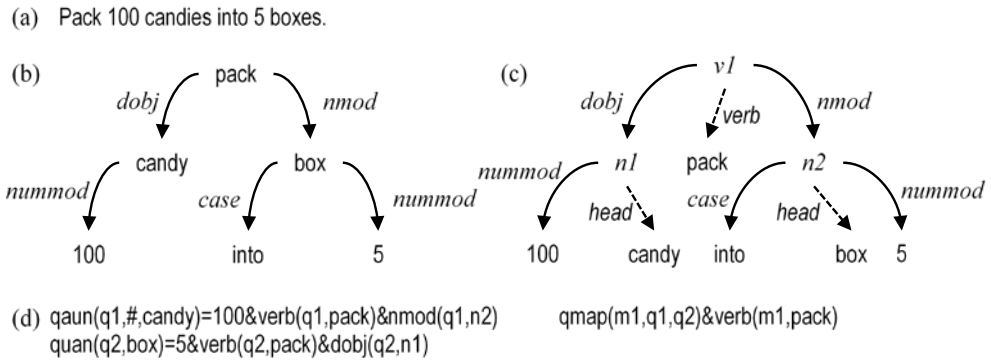


Figure 3: An example of logic form transformation

In addition to the properties *Entity*[4] and *Verb* (Hosseini et al., 2014) associated with the *quantity*, we also introduce a new property *Time* which encodes the tense and aspect of a verb into an integer to specify a point in the timeline. We assign 2, 4, and 6 to the tenses *Past*, *Present* and *Future*, respectively, and then adjust it with the aspect-values -1, 0 and 1 for *Perfect*, *Simple*, and *Progressive*, respectively.

Another property *Anchor* is associated with the *unknown quantity* asked in the question sentence. If the subject of the question sentence is a noun phrase (e.g., "*how many apples does **John** have?*"), *Anchor* is the *subject* (i.e., *John*). If the subject is an expletive nominal (e.g. "*how many apples are there in the box?*"), then *Anchor* is the associated nominal modifier *nmod* (i.e., "*box*"). Otherwise, *Anchor* is set to "*Unknown*".

Inspired by (Hosseini et al., 2014), we transform *Verb* to *Verb-Class* (*VC*) which is *positive*, *negative* or *stative*. A verb is *positive/negative* if it increases/decreases the associated quantity of the subject. For example, in the sentence "*Tom borrowed 3 dollars from Mike*", the verb is *positive* because the money of subject "*Tom*" increases.

However, a *positive* verb does not always imply the *Addition* operation. If the question is "*How much money does Mike have now?*" for the above body sentence, *the operation* should be *Subtraction*. Two new properties *Anchor-Role* (*AR*) and *Action* (*A*) are thus proposed: AR_i indicates the role that *Anchor* associated with q_i, and is set to *nsubj*/*obj*/*nmod*/ϕ. A_i is determined by following rules: (1) A_i=*positive if* (VC_i, AR_i) is either (*positive*, *nsubj*) or (*negative*, *obj*/*nmod*). (2) A_i=*negative if* (VC_i, AR_i) is either (*negative*, *nsubj*) or (*positive*, *obj*/*nmod*). (3) Otherwise, $A_i=VC_i$.

To rule out the noisy quantities introduced by irrelevant information, we further associate each known quantity with the property *Relevance* (*R*) according to the unknown quantity asked in the question sentence. Let q_i denote the *i*-th known quantity, E_i denote the entity of q_i, X_i denote the property *X* of q_i, q_U denote the unknown quantity asked, and X_U denote the property *X* of q_U. R_i is specified with following rules: (1) $R_i=2$ (Directly-Related) if either {*Anchor is Unknown* & E_i *entails* E_U} or {*Anchor is not Unknown* & $AR_i \neq \phi$ & E_i entails E_U} (2) $R_i=1$ (Indirectly-Related) if there is a q_j which maps[5] to q_i and $R_j=2$ (i.e., q_j is Directly-Related). (3) $R_i=0$ (Unrelated) otherwise.

The solution type is identified by an SVM based on 26 binary features. Let the symbols **p**, **n**, **s**, *A*, *E*, *R*, *T*, *V*, S_B, S_Q and w_Q stand for *positive*, *negative*, *stative*, *Action*, *Entity*, *Relevance*, *Time*, *Verb*, "*a body sentence*", "*the question sentence*" and "*a word in question sentence*" respectively. Also, let $I(x)$ be the indicator function to check if x is true. The 26 features are briefly described as follows:

(1) VC_U=**p**; (2) $\exists R_i$=2 s.t. A_i=**p**; (3) $\exists R_i$=2 s.t. A_i=**n**;
(4) $\exists R_i$=2 s.t. A_i=**s**; (5) $\sum_i I(R_i=2) > 2$;
(6) $\sum_i I(R_i=2 \,\&\, A_i \in \{\mathbf{p}, \mathbf{n}\}) = 2$;
(7) $\exists R_i$=2 s.t. A_i=**p** & $T_U<T_i$;
(8) $\exists R_i$=2 s.t. A_i=**n** & $T_U<T_i$;
(9) $\exists R_i$=2 s.t. A_i=**s** & T_i=max T_j;
(10) $\exists R_i$=2 s.t. A_i=**s** & $T_i<T_U$;
(11) $T_U \geq \max T_i$; (12) $T_U \leq \min T_i$;
(13) $\forall R_i$=2, V_i are the same; (14) $\forall R_i$=2 s.t. $T_i=T_U$;
(15) $\forall R_i$=2, T_i are the same;
(16) $\exists R_i$=2, $\exists R_j$=1 s.t. q_i maps to q_j & $q_i > q_j$;

[4] In our works, the term "*Entity*" also includes the unit of the quantity (e.g., "*cup of coffee*").

[5] That is, q_i is linked to a directly-related quantity q_j under an expression such as "*2 pencils weigh 30 grams*".

(a) A sandwich is priced at $0.75. A pudding is priced at $0.25. Tim bought 2 sandwiches and 4 puddings. Mary bought 2 puddings. How much money should Tim pay?

(b) …price(sandwich,0.75)&price(pudding,0.25)… quan(q1,#,sandwich)=2&verb(q1,buy)&nsubj(q1,Tim)… quan(q2,#,pudding)=4&verb(q2,buy)&nsubj(q2,Tim)… quan(q3,#,pudding)=2&verb(q3,buy)&nsubj(q3,Mary)… ASK Sum(quan(?q,dollar,#),verb(?q,pay)&nsubj(?q,Tim))

(c) quan(?q,?u,?o)&verb(?q,buy)&nsubj(?q,?a)&price(?o,?p) → quan($q,dollar,#)=quan(?q,?u,?o)×?p & verb($q,pay) & nsubj($q,?a)

(d) quan(q4,dollar,#)=1.5&verb(q4,pay)&nsubj(q4,Tim)… quan(q5,dollar,#)=1&verb(q5,pay)&nsubj(q5,Tim)… quan(q6,dollar,#)=0.5&verb(q6,pay)&nsubj(q6,Mary)

Figure 4: A logic inference example

(17) $\exists R_i$=2, $\exists R_j$=1 s.t. q_i maps to q_j & q_i is associated with a word "each/every/per/a/an";
(18) $\exists R_i$=2, $\exists R_j$=1 s.t. q_i maps to q_j & q_j is associated with a word "each/every/per/a/an";
(19) $\exists q_i, q_j, q_k$ s.t. $R_i = R_j = R_k$ =2 & $V_i = V_j = V_k$;
(20) $\exists w_Q \in$ {*total, in all, altogether, sum*};
(21) $\exists w_Q \in$ {*more, than}* or $\exists w_Q$ s.t. w_Q-POS=RBR;
(22) $\exists w_Q$ ="*left*"; (23). $\exists q_i$ appears in S_Q;
(24) "the rest ∨ E_U" appears in S_B (∨ for any verb);
(25) "each NN" appears in S_Q (NN for any noun);
(26) *Anchor*$_U$ is *Unknown*/*nmod* & VC_U = **s**.

2.2 Logic Form Transformation (LFT)

The results of language analysis are transformed into a logic form, which is expressed with the *first-order logic* (FOL) formalism (Russell and Norvig, 2009). Figure 3 shows how to transform the sentence (a) "*Pack 100 candies into 5 boxes.*" into the corresponding logic form (d). First, the dependency tree (b) is transformed into the semantic representation tree (c) adopted by Lin et al., (2015). Afterwards, according to the procedure proposed in (Lin et al., 2015), the domain-dependent logic expressions are generated in (d).

The domain-dependent logic expressions are related to crucial generic math facts, such as *quantities* and *relations* between quantities. The FOL function *quan(quan$_{id}$, unit[6],entity)=number* is for describing the quantity fact. The first argument denotes its unique *identifier*. The other arguments and the function value describe its meaning. Another FOL predicate *qmap(map$_{id}$, quan$_{id1}$, quan$_{id2}$)* (denotes the mapping from *quan$_{id1}$* to *quan$_{id2}$*) is for describing a relation between two quantity facts, where the first argument is a unique identifier to represent this relation.

The role-tags (e.g., *verb*, *dobj*, etc.) associated with *quan$_{id}$* and *map$_{id}$* denote entity attributes (i.e., the physical meaning of the quantity), are created to help the logic inference module find the solution. For example, *quan(q_2,#,box) = 5 & verb(q_2,pack) &…* means that q_2 is the quantity of boxes being packed. With those role-tags, the system can select the operands more reliably, and the inference engine can also derive new quantities to solve complex MWPs which require multi-step arithmetic operations (see section 2.3).

The question in the MWP is also transformed into an FOL-like utility function according to the solution type to ask the logic inference module to find out the answer. For example, the utility function instance *Division(quan(q_1, #, candy), quan(q_2, #, box))* asks the inference module to divide "*100 candies*" by "*5 boxes*". Since associated operands must be specified before calling those utility functions, a statistical model (see section 2.4) is used to identify the appropriate quantities.

2.3 Logic Inference

The logic inference module adopts the inference engine from (Lin et al., 2015). Figure 4 shows how it uses inference rules to derive new facts from the initial facts directly provided from the description. The MWP (a) provides some facts (b) generated from the LFT module. An inference rule (c)[7], which implements the common sense that people must pay money to buy something, is unified with the given facts (b) and derives new facts (d). The facts associated with *q6* can be interpreted as "*Mary paid 0.5 dollar for two puddings*".

The inference engine (IE) also provides 5 utility functions, including *Addition*, *Subtraction*, *Multiplication* and *Division*, and *Sum*. The first four utilities all return a value by performing the named math operation on its two input arguments. On the other hand, *Sum(function,condition)* returns the sum of the values of FOL *function instances* which can be unified with the first argument (i.e., *function*) and satisfy the second argument (i.e., *condition*). For example, according to

[6] This second argument denotes the associated unit used to count the entity. It is set to "#" if the unit of the entity is not specified.

[7] In the inference rule, $q is a meta symbol to ask the inference engine to generate a unique identifier for the newly derived quantity fact.

(a) Tim bought 2 roses and 3 lilies. Mary bought 4 roses and 5 lilies. How many flowers did Tim buy?

(b) quan(q1,#,rose)=2&verb(q1,buy)&nsubj(q1,Tim)…
quan(q2,#,lily)=3&verb(q2,buy)&nsubj(q2,Tim)…
quan(q3,#,rose)=4&verb(q3,buy)&nsubj(q3,Mary)…
quan(q4,#,lily)=5&verb(q4,buy)&nsubj(q4,Mary)…
quan(q_Q,#,flower)&verb(q_Q,buy)&nsubj(q_Q,Tim)…

Figure 5: An example for operand selection

the last line in Figure 4(b), three newly derived quantity facts *q4*, *q5* and *q6* (in 4(d)) can be unified with the first argument *quan(?q,dollar,#)* in 4(c), but only *q4* and *q5* satisfy the second argument *verb(?q,pay)&nsubj(?q,Tim)*. As a result, the answer 2.5 is returned by taking sum on the values of the quantity facts *quan(q4,dollar,#)* and *quan(q5,dollar,#)*.

2.4 Probabilistic Operand Selection

The most error-prone part in the LFT module is instantiating the utility function of math operation especially if many irrelevant quantity facts appear in the given MWP. Figure 5 shows the LFT module needs to select two quantity facts (among 4) for *Addition*. Please note that the *question quantity* q_Q, transformed from "*how many flowers*", is not a candidate for operand selection.

Lin et al., (2015) used predefined lexico-syntactic patterns and ad-hoc rules to instantiate utility functions. However, their rule-based approach fails when the MWP involves more quantities. Therefore, we propose a statistical model to select operands for the utility functions *Addition*, *Subtraction*, *Multiplication* and *Division*. The operand selection procedure can be regarded as finding the most likely configuration (o_1^n, r), where $o_1^n = o_1, \cdots, o_n$ is a sequence of random indicators which denote if the corresponding quantity will be selected as an operand, and r is a tri-state variable to represent the relation between the values of two operands (i.e., $r = -1, 0$ or 1 ; which denote that the first operand is less than, equal to, or greater than the second operand, respectively). Given a solution type s, the MWP logic expressions $\mathbb{L}$ and the n quantities $q_1^n = q_1, \cdots, q_n$ in $\mathbb{L}$. The procedure is formulated as:

$$P(r, o_1^n | q_1^n, \mathbb{L}, s) \approx P(r|s) \times P(o_1^n | q_1^n, \mathbb{L}, s), \quad (1)$$

$P(r|s)$ simply refers to *Relative Frequency* (as it has only a few parameters and we have enough training samples). $P(o_1^n | q_1^n, \mathbb{L}, s)$ is further derived as:

$$P(o_1^n | q_1^n, \mathbb{L}, s) \approx \prod_{i=1}^{n} P(o_i | q_i, \mathbb{L}, s) \approx \prod_{i=1}^{n} P\big(o_i \big| \Phi(q_i, \mathbb{L}, s)\big), \quad (2)$$

where $\Phi(\cdot)$ is a feature extraction function to map q_i and its context into a feature vector. Here, the probabilistic factor $P\big(o_i \big| \Phi(q_i, \mathbb{L}, s)\big)$ is obtained via an SVM classifier (Chang and Lin, 2011).

$\Phi(\cdot)$ extracts total 25 features (specified as follows, and 24 of them are binary) for q_i. The following 11 of them are independent on the question in the MWP:

1. Four features to indicate if s is *Addition*, *Subtraction*, *Multiplication* or *Division*.
2. A feature to indicate if q_i is within a *qmap*(…).
3. A feature to indicate if $q_i = 1$.
4. Five features to indicate if $n < 2$, $n = 2$, $n = 3$, $n = 4$ or $n > 4$; where n is the number of quantities in Eq (1).

$\Phi(\cdot)$ also extracts features by matching the logic expressions of q_i with those of *question quantity* $\mathrm{q_Q}$ to check the role-tag consistencies between q_i and $\mathrm{q_Q}$. Another fourteen features are extracted with three indicator functions $I_m(\cdot)$, $I_e(\cdot)$, $I_\exists(\cdot)$ and one tri-state function $T_m(\cdot)$ as follows:

$$\begin{aligned}[& I_m(q_i, \mathrm{q_Q}, \text{entity}), I_e(q_i, \mathrm{q_Q}, \text{entity}),\\ & I_m(q_i, \mathrm{q_Q}, \text{verb}), I_e(q_i, \mathrm{q_Q}, \text{verb}),\\ & I_\exists(\mathrm{q_Q}, \text{nsubj}), T_m(q_i, \mathrm{q_Q}, \text{nsubj}),\\ & I_\exists(\mathrm{q_Q}, \textit{modifier}), I_m(q_i, \mathrm{q_Q}, \textit{modifier}),\\ & I_\exists(\mathrm{q_Q}, \textit{place}), I_m(q_i, \mathrm{q_Q}, \textit{place}),\\ & I_\exists(\mathrm{q_Q}, \textit{temporal}), I_m(q_i, \mathrm{q_Q}, \textit{temporal}),\\ & I_\exists(\mathrm{q_Q}, \textit{xcomp}), I_m(q_i, \mathrm{q_Q}, \textit{xcomp})]\end{aligned}$$

where the indicator functions $I_m(x, y, z)$ checks if the z of x matches the z of y, $I_e(x, y, z)$ checks if the z of x entails the z of y and $I_\exists(y, z)$ checks if the z of y exists. $T_m(q_i, \mathrm{q_Q}, \text{nsubj})$ returns "exact-match" (if *nsubj* of q_i matches *nsubj* of $\mathrm{q_Q}$), "quasi-match" (if *nsubj* of $\mathrm{q_Q}$ does not exist or is a plural pronoun), and "unmatch".

$I_e(\cdot)$ uses the WordNet hypernym and hyponym relationship to judge whether one entity/verb entails another one or not via checking if they are in an inherited hypernym-path in WordNet. The *entity*, *verb* and *nsubj* of a quantity are determined according to the logic expressions. The modifier, place, temporal and *xcomp* of a quantity are extracted from the dependency tree with some lexico-syntactic patterns. For example, the *modifier* and *place* of the quantity in the sentence "*There are 30 red flowers in the garden.*" are "*red*" and "*garden*" respectively. The *temporal*

and *xcomp* of a quantity are extracted according to the dependency relations "*tmod*" (i.e., *temporal modifier*) and "*xcomp*" (i.e., *open clausal complement*), respectively.

3 Datasets for Performance Evaluation

The AI2 dataset provided by Hosseini et al. (2014) and the IL dataset released by Roy and Roth (2015) are adopted to compare our approach with other state-of-the-art methods. The AI2 dataset has 395 MWPs on addition and subtraction, with 121 MWPs containing irrelevant information (Hosseini et al., 2014). It is the most popular one for comparing different approaches. On the other hand, the IL dataset consists of 562 elementary MWPs which can be solved by one of the four arithmetic operations (i.e., +, −, ×, and ÷) without any irrelevant quantity. It is the first publicly available dataset for comparing performances that covers all four arithmetic operations.

However, the difficulty of solving an MWP depends not only on the number of arithmetic operations required, but also on how many irrelevant quantities inside, and even on how the quantities are described. One way to test if a proposed approach solves the MWPs with understanding is to check whether it is robust to those irrelevant quantities. Therefore, it is desirable to have a big enough dataset that contains irrelevant quantities which are created under different situations (e.g., confusing with an irrelevant agent, entity, or modifier, etc.) and allow us to probe the system weakness from different angles. We thus create a new dataset with more irrelevant quantities[8]. But before we do that, we need to know how difficult the task of solving the given MWPs is. Therefore, we first propose a way to measure how easy that a system solves the problem by simply guessing.

3.1 Perplexity-flavor Measure

We propose to adopt the *Perplexity* to measure the task difficulty, which evaluates how likely a solver will get the correct answer by guessing. Every MWP in the datasets can be associated with a solution expression template, such as "$\boxdot + \boxdot$" or "$\boxdot - \boxdot$", where the symbol $\boxdot$ represents a slot to hold a quantity. The solution can be obtained by placing correct quantities at appropriate slots. A random baseline is to solve an MWP by guessing. It first selects a solution expression template according to the prior distribution of the templates and then places quantities into the selected template according to the uniform distribution.

The expected accuracy of the random baseline on solving an MWP is a trivial combination and permutation exercise[9]. For example, the expected accuracy of solving an MWP associated with "$\boxdot + \boxdot$" template is $p_{\boxdot+\boxdot} \times {}_nC_2^{-1}$, where the factor $p_{\boxdot+\boxdot}$ denotes the prior probability of the template "$\boxdot + \boxdot$" and n is the total number of quantities (including irrelevant ones) in the MWP. On the other hand, expected accuracy of solving an MWP associated with "$\boxdot - \boxdot$"[10] template is $p_{\boxdot-\boxdot} \times {}_nP_2^{-1}$. Let A_i denote the expected accuracy of solving the i-th MWP in a dataset. The accuracy of the random baseline on the dataset of size N is then computed as $A = (1/N) \times \sum_{i=1}^{N} A_i$.

The word "*Accuracy*" comprises the opposite sense of the word "*Perplexity*"[11] (i.e., in the sense of how hard a prediction problem is). The lower the *Accuracy* is, the higher the *Perplexity* is. Therefore, we transform the *Accuracy* measure into a *Perplexity-Flavor* measure (PP) via the formula:

$$\mathrm{PP} = 2^{-\log_2 A}$$

For instance, the *Perplexity-Flavor* measures of AI2 and IL datasets are 4.46 and 8.32 respectively.

3.2 Noisy Dataset

Human Math/Science tests have been considered more suitable for judging AI progress than Turing test (Clark and Etzioni, 2016). In our task, solving MWPs is mainly regarded as a test for intelligence (not just for creating a *Math Solver* package). By injecting various irrelevant quantities into original MWPs, a noisy dataset is thus created to assess if a solver solves the MWPs mainly via *understanding* or via *mechanical/statistical pattern matching*. If a system solves an MWP mainly via pattern matching, it would have difficulty in solving a similar MWP augmented from the original one with some irrelevant quantities. Therefore, we first create a noisy dataset by selecting some

[8] The IL dataset does not include any irrelevant information; on the other hand, the AI2 dataset only contains 121 MWPS with irrelevant information (but not systematically created).

[9] Let ${}_nC_k$ denote k-combinations of n and ${}_nP_k$ denote k-permutations of n.

[10] We assume the operands have different values and, therefore, they are not permutable for the *subtraction* operator.

[11] The *Perplexity* of a uniform distribution over *k* discrete events (such as casting a fair *k*-sided dice) is *k*.

(a) Tim has 10 yellow flowers and 12 red flowers. How many flowers does Tim have?

(a.1) Tim has … Mary has 3 yellow flowers. How many …

(a.2) Tim has … Tim also has 3 books. How many …

Figure 6: Examples of noisy sentences

MWPs that can be correctly solved, and then augmenting each of them with an additional noisy sentence which involves an irrelevant quantity. This dataset is created to examine if the solver knows that this newly added quantity is irrelevant.

Figure 6 shows how we inject noise into an MWP (a). (a.1) is created by associating an irrelevant quantity to a new subject (i.e., *Mary*). Here the ellipse symbol "…" denotes unchanged text. (a.2) is obtained by associating an irrelevant quantity to a new entity (i.e., *books*). In addition, we also change modifiers (such as *yellow*, *red*, …) to add new noisy sentence (not shown here). Since the noisy dataset is not designed to assess the lexicon coverage rate of a solver, we reuse the words in the original dataset as much as possible while adding new subjects, entities and modifiers.

136 MWPs that both Illinois Math Solver [12] (Roy and Roth, 2016) and our system can correctly solve are selected from the AI2 and IL datasets. This subset is denoted as OSS (Original Sub-Set). Afterwards, based on the 136 MWPs of OSS, we create a noisy dataset of 396 MWPs by adding irrelevant quantities. This noisy dataset is named as NDS [13]. Table 1 lists the size of MWPs, *Perplexities* (PP), and the average numbers of quantities in each MWP of these two datasets.

4 Experimental Results and Discussion

We compare our approach with (Roy and Roth, 2015) and (Roy and Roth, 2017) because they achieved the state-of-the-art performance on the IL dataset. In the approach of (Roy and Roth, 2015), each quantity in the MWP was associated with a *quantity schema* whose attributes are extracted from the context of the quantity. Based on the attributes, several statistical classifiers were used to select operands and determine the operator. They also reported the performances on the AI2 dataset to compare their approach with those of others (e.g., Kushman et al. (2014), which is a purely statistical approach that aligns the text with various pre-extracted equation templates). Roy and Roth (2017) further introduced the concept of *Unit Dependency Graphs* to reinforce the consistency of physical units among selected operands associated with the same operator.

	OSS	NDS
# MWPs	136	396
Perplexity (PP)	7.42	18.83
#Quantities	2.64	3.62

Table 1: *Perplexity* measures of OSS and NDS

	AI2	IL
Our system (Statistical)	**81.5**	**81.0**
Our system (DNN)	69.8	70.6
(Roy and Roth, 2017)	76.2	74.4
(Roy and Roth, 2015)	78.0	73.9
(Kushman et al., 2014)	64.0	73.7

Table 2: Performances of various approaches

	AI2	IL
STI (Statistical)	83.0	83.1
STI (DNN)	74.5	68.8
LFT	92.1	94.8

Table 3: Performances of *different* STIs and LFT

To compare the performance of the statistical method with the DNN approach, we only implement a Bi-directional RNN-based *Solution Type Identifier* (as our original statistical *Operand Selector* is relatively much better). It consists of a word embedding layer (for both body and question parts), and a bidirectional GRU layer as an encoder. We apply the attention mechanism to scan all hidden state sequence of body by the last hidden state of question to pay more attention to those more important (i.e., more similar between the body and the question) words. Lastly, it outputs the solution type by a softmax function. We train it for 100 epochs, with mini-batch-size = 128 and learning-rate = 0.001; the number of nodes in the hidden layer is 200, and the drop-out rate is 0.7 [14].

We follow the same n-fold cross-validation evaluation setting adopted in (Roy and Roth, 2015) exactly. Therefore, various performances could be directly compared. Table 2 lists the accuracies of different systems in solving the MWPs

[12] We submit MWPs to Illinois Math Solver (https://cogcomp.cs.illinois.edu/page/demo_view/Math) in May and June, 2017.

[13] The noisy dataset can be downloaded from https://github.com /chaochun/nlu-mwp-noise-dataset. It includes 102 Addition, 147 Subtraction, 101 Multiplication and 46 Division MWPs.

[14] Since the dataset is not large enough for splitting a development set, we choose those hyper parameters based on the test set in coarse grain. Therefore, the DNN performance we show here might be a bit optimistic.

of various datasets. The performance of (Roy and Roth, 2017) system is directly delivered by their code[15]. The last two rows are extracted from (Roy and Roth, 2015). The results show that our performances of the statistical approach significantly outperform that of our DNN approach and other systems on every dataset.

The performances of STI and LFT modules are listed in Table 3. As described in section 2, the benchmark for both solution type and the operand selection benchmark are automatically determined by weakly supervised learning. The first and second rows of Table 3 show the solution type accuracies of our statistical and DNN approaches, respectively. The third row shows the operand selection accuracy obtained by given the solution type benchmark. Basically, LFT accuracies are from 92% to 95%, and the system accuracies are dominated by STI. We analyzed errors resulted from our statistical STI on AI2 and IL datasets, respectively. For AI2, major errors come from: (1) failure of ruling out some irrelevant quantities (40%), and (2) making confusion between *TVQ-F* and *Sum* these two solution types (20%) for those cases that only involve addition operation (however, both types would return the same answer). For IL, major errors come from: (1) requiring additional information (35%), and (2) not knowing Part-Whole relation (17%). Table 4 shows a few examples for different STI error types.

The left-half of Table 5 shows the performances on the OSS and NDS datasets. Recall that OSS is created by selecting some MWPs which both Illinois Math Solver (Roy and Roth, 2016) and our system[16] can correctly solve. Therefore, both systems have 100% accuracy in solving the OSS dataset. However, these two systems behave very differently while solving the noisy dataset. The much higher accuracy of our system on the noisy dataset shows that our meaning-based approach understands the meaning of each quantity more. Therefore, it is less confused[17] with the irrelevant quantities.

One MWP in the noisy dataset that confuses *Illinois Math Solver* (IMS) is "*Tom has 9 yellow balloons. Sara has 8 yellow balloons. Bob has 5 yellow flowers. How many yellow balloons do they have in total?*", where the underlined sentence is the added noisy sentence. The solver sums all quantities and gives the wrong answer 22, which reveals that IMS cannot understand that the quantity "*5 yellow flowers*" is irrelevant to the question "*How many yellow balloons?*". On the contrary, our system avoids this mistake.

Error Type	Example
Confusing *TVQ-F* and *Sum* solution type	Sally found 9 seashells, Tom found 7 seashells, and Jessica found 5 seashells on the beach. How many seashells did they find together?
Requiring additional information	A garden has 52 rows and 15 columns of bean plans. How many plants are there in all?
Not knowing Part-Whole relationship	Eric wants to split a collection of peanuts into groups of 8. Eric has 64 peanuts. How many groups will be created?

Table 4: Examples for different STI error types

	Statistical	R&R, 2016		Statistical	DNN
OSS	100	100	OSS′	100	100
NDS	82.1	28.5	NDS′	81.4	75.4

Table 5: Performances on the OSS and NDS

Although the meaning of each quantity is explicitly checked in our LFT module, our system still cannot correctly solve all MWPs in NDS. The error analysis reveals that the top-4 error sources are STI, LFT, CoreNLP and incorrect problem construction (for 27%, 27%, 18%, 18%), which indicates that our STI and LFT still cannot completely prevent the damage caused from the noisy sentences (which implies that more consistency check for quantity meaning should be done). The remaining errors are due to incorrect quantity extraction, lacking common-sense or not knowing entailment relationship between two entities.

A similar experiment is performed to check if the DNN approach will be affected by the noisy information more. We first select 124 MWPs (denoted as OSS′) from OSS that can be correctly solved by both our statistical and DNN approaches and then filter out 350 derived MWPs (denotes as NDS′) from NDS. The right-half of Table 5 shows that the performance of the DNN approach drops more than the statistical approach does in the noisy dataset, which indicates that our statistical approach is less sensitive to the irrelevant quantities and more close to human's approach.

[15] https://github.com/CogComp/arithmetic.

[16] In evaluating the performances on OSS and NDS datasets, our system is trained on the folds 2-5 of the IL dataset.

[17] Since the gap between two different types of approaches is quite big, those 396 examples on OSS and 196 examples on NDS are sufficient to confirm the conclusion.

5 Related Work

To the best of our knowledge, MWP solvers proposed before 2014 all adopted the rule-based approach. Mukherjee and Garain (2008) had given a good survey for all related approaches before 2008. Afterwards, Ma et al. (2010) proposed a MSWPAS system to simulate human arithmetic multi-step addition and subtraction behavior without evaluation. Besides, Liguda and Pfeiffer (2012) proposed a model based on augmented semantic networks, and claimed that it could solve multi-step MWPs and complex equation systems and was more robust to irrelevant information (also no evaluation).

Recently, Hosseini et al. (2014) proposed a *Container-Entity* based approach, which solved the MWP with a sequence of state transition. And Kushman et al. (2014) proposed the first statistical approach, which heuristically extracts some algebraic templates from labeled equations, and then aligns them with the given sentence. Since no semantic analysis is conducted, the performance is quite limited.

In more recent researches (Roy and Roth, 2015; Koncel-Kedziorski et al., 2015; Roy and Roth, 2017), quantities in an MWP were associated with attributes extracted from their contexts. Based on the attributes, several statistical classifiers were used to select operands and determine operators to solve multi-step MWPs. Since the physical meaning of each quantity is not explicitly considered in getting the answer, the reasoning process cannot be explained in a human comprehensible way. Besides, Shi et al. (2015) attacked the *number word problem*, which only deal with numbers, with a semantic parser. Mitra and Baral (2016) mapped MWPs into three types of problems, including Part-Whole, Change and Comparison. Each problem was associated with a generic formula. They used a log-linear model to determine how to instantiate the formula with quantities and solve the only one *Unknown* variable. They achieved the best performance on the AI2 dataset. However, their approach cannot handle *Multiplication* or *Division* related MWPs. Recently, DNN-based approaches (Ling et al, 2017; Wang et al, 2017) have emerged. However, they only attacked algebraic word problems, and required a very large training-set.

Our proposed approach mainly differs from those previous approaches in *combining the statistical framework with logic inference*, and also in *adopting the meaning-based statistical approach for selecting the desired operands*.

6 Conclusion

A *meaning-based* logic form represented with *role-tags* (e.g., *nsubj*, *verb*, etc.) is first proposed to associate the extracted math quantity with its physical meaning, which then can be used to identify the desired operands and filter out irrelevant quantities. Afterwards, a statistical framework is proposed to perform understanding and reasoning based on those logic expressions. We further compare the performance with a typical DNN approach, the results show the proposed approach is still better. We will try to integrate domain concepts into the DNN approach to improve the learning efficiency in the future.

The main contributions of our work are: (1) Adopting a meaning-based approach to solve English math word problems and showing its superiority over other state-of-the-art systems on common datasets. (2) Proposing a statistical model to select operands by explicitly checking the meanings of quantities against the meaning of the question sentence. (3) Designing a noisy dataset to test if a system solves the problems by understanding. (4) Proposing a perplexity-flavor measure to assess the complexity of a dataset.

References

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

Yefim Bakman. 2007. Robust understanding of word problems with extraneous information. http://lanl.arxiv.org/abs/math.GM/0701393.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27:1–27:27. Software available at http://www.csie.ntu.edu.tw/cjlin/libsvm.

Peter Clark and Oren Etzioni. 2016. My computer is an honor student - but how Intelligent is it? Standardized Tests as a Measure of AI. *AI Magazine*, pages 5–12.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, EMNLP 2014, October 25-29, 2014, Do-

ha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL, pages 523–533.

Chien-Tsung Huang, Yi-Chung Lin, and Keh-Yih Su. 2015. Explanation generation for a math word problem solver. *International Journal of Computational Linguistics and Chinese Language processing* (IJCLCLP), 20(2):27–44.

Danqing Huang, Shuming Shi, Chin-Yew Lin and Jian Yin. 2017. Learning fine-grained expressions to solve math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 816–825.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 271–281.

Chao-Chun Liang, Shih-Hong Tsai, Ting-Yun Chang, Yi-Chung Lin and Keh-Yih Su. 2016. A meaning-based English math word problem solver with understanding, Reasoning and Explanation. In *Proceedings of COLING 2016*.

Christian Liguda and Thies Pfeiffer. 2012. Modeling math word problems with augmented semantic networks. In Gosse Bouma, Ashwin Ittoo, Elisabeth Metais, and Hans Wortmann, editors, *Natural Language Processing and Information Systems/17th International Conference on Applications of Natural Language to Information Systems*, volume 7337. Springer.

Yi-Chung Lin, Chao-Chun Liang, Kuang-Yi Hsu, Chien- Tsung Huang, Shen-Yun Miao, Wei-Yun Ma, Lun-Wei Ku, Churn-Jung Liau, and Keh-Yih Su. 2015. Designing a tag-based statistical math word problem solver with reasoning and explanation. *International Journal of Computational Linguistics and Chinese Language processing* (IJCLCLP), 20(2):1–26.

Wang Ling, Dani Yogatama, Chris Dyer and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 158–167.

Yuhui Ma, Ying Zhou, Guangzuo Cui, Yun Ren, and Ronghuai Huang. 2010. Frame-based calculus of soling arithmetic multi-step addition and subtraction word problems. *Education Technology and Computer Science, International Workshop* on, 2:476–479.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL Demonstrations*.

Arindam Mitra and Chitta Baral. 2016. Learning to use formulas to solve simple arithmetic problems. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 2144–2153.

Anirban Mukherjee and Utpal Garain. 2008. A review of methods for automatic understanding of natural language mathematical problems. *Artif. Intell.* Rev., 29(2):93–122.

Stephen J. Pape. 2004. Middle school children's problem-solving behavior: A cognitive analysis from a reading comprehension perspective. *Journal for Research in Mathematics Education*, 35(3):187–219.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752.

Subhro Roy and Dan Roth. 2016. Illinois Math Solver: Math reasoning on the web. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 52–66.

Subhro Roy and Dan Roth. 2017. Unit dependency graph and its application to arithmetic word problem solving. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*.

Subhro Roy, Tim Vieira, and Dan Roth. 2015. Reasoning about quantities in natural language. Transactions of the *Association for Computational Linguistics*, 3:1–13.

Stuart Russell and Peter Norvig. 2009. *Artificial Intelligence*: A modern approach. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition.

Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1132–1142.

Manish Shrivastava and Dipti Misra Sharma. 2017. Deep neural network based system for solving arithmetic word problems. In *Proceedings of the 2017 International Joint Conference on Natural Language Processing*, pages 65–68.

Shyam Upadhyay, Ming-Wei Chang, Kai-Wei Chang, and Wen-tau Yih. 2016. Learning from explicit and implicit supervision jointly for algebra word prob-

lems. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 297–306.

Yan Wang, Xiaojiang Liu and Shuming Shi. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 856–865.

Lipu Zhou, Shuaixiang Dai, and Liwei Chen. 2015. Learn to solve algebra word problems using quadratic programming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 817–822.

Fine-grained Temporal Orientation and its Relationship with Psycho-demographic Correlates

Sabyasachi Kamila[1], **Mohammed Hasanuzzaman**[2], **Asif Ekbal**[1], **Pushpak Bhattacharyya**[1] **and Andy Way**[2]

[1]Department of Computer Science and Engineering,
Indian Institute of Technology Patna, India
[2]ADAPT Centre, School of Computing, Dublin City University, Dublin, Ireland
sabysachi.pcs16@iitp.ac.in

Abstract

Temporal orientation refers to an individual's tendency to connect to the psychological concepts of *past*, *present* or *future*, and it affects personality, motivation, emotion, decision making and stress coping processes. The study of the social media users' psycho-demographic attributes from the perspective of human temporal orientation can be of utmost interest and importance to the business and administrative decision makers as it can provide an extra precious information for them to make informed decisions. In this paper, we propose a very first study to demonstrate the association between the sentiment view of the temporal orientation of the users and their different psycho-demographic attributes by analyzing their tweets. We first create a temporal orientation classifier in a minimally supervised way which classifies each tweet of the users in one of the three temporal categories, namely *past*, *present*, and *future*. A deep Bi-directional Long Short Term Memory (BLSTM) is used for the tweet classification task. Our tweet classifier achieves an accuracy of 78.27% when tested on a manually created test set. We then determine the users' overall temporal orientation based on their tweets on the social media. The sentiment is added to the tweets at the fine-grained level where each temporal tweet is given a sentiment with either of the positive, negative or neutral. Our experiment reveals that depending upon the sentiment view of temporal orientation, a user's attributes vary. We finally measure the correlation between the users' sentiment view of temporal orientation and their different psycho-demographic factors using regression.

1 Introduction

The rapid growth of social media data in recent years has encouraged different studies which only existed at the psychological level (theory or pure logic). Various attributes of users can be analyzed from the texts they write on the social media platform. The studies include age, gender prediction (Marquardt et al., 2014; Sap et al., 2014), psychological well being (Dodds et al., 2011; Choudhury et al., 2013), and a host of other behavioural, psychological and medical phenomena (Kosinski et al., 2013). However, a few works exist which analyze these factors using socio-economic characteristics of the Twitter users.

Time is generally defined by a dimension where the events are ordered from the past through the *present* into the *future* which includes duration and intervals. Major studies on time have been done for event detection (Ihler et al., 2006; Batal et al., 2012; Sakaki et al., 2013) which are mainly of the subjective consent. In contrast, temporal orientation of a user is defined by his/her tendency to emphasize *past*, *present* or *future* (Zimbardo and Boyd, 2015) which gives more objective consent of time. The growth of social media content has enabled us to study this objective consent more precisely.

Past studies have established a consistent link between the temporal orientation and several user characteristics such as age, gender, education, and psychological traits (Webley and Nyhus, 2006; Adams and Nettle, 2009; Schwartz et al., 2013; Zimbardo and Boyd, 2015). However, the sentiment dimension (positive, negative or neutral) of the temporal orientation is merely studied at the empirical level on a large-scale. For example, people who are optimistic are future-oriented and positive at the same time. So, only defining the temporal orientation cannot specify the optimistic people correctly. We need the sentiment dimension as well to find the exact correlation.

In this paper, we first develop a temporal-orientation classifier to classify tweets into *past*, *present*, and *future* and then group over the users

Proceedings of NAACL-HLT 2018, pages 663–674
New Orleans, Louisiana, June 1 - 6, 2018.

to create user-level assessments. We use a Bidirectional Long Short Term Memory (Bi-LSTM) network for tweet temporal classification where tweet vectors are fed to generate the classification model. We propose a hash tag-based *minimally supervised* method with the two-pass filtering to create the *past*, *present* and *future*-oriented tweets for the training of the Bi-LSTM network. We manually examined trending hashtags in Twitter for a specific period of time and selected hashtags which represent *past, present/ongoing*, or *future* events. The English tweets containing one of the selected hashtags are crawled using Twitter streaming API.[1] The tweet temporal orientation classifier is validated on a manually annotated test set. Finally, we use this classifier to automatically classify a large dataset consisting of ≈10 million tweets from 5,191 users mapped to their user-level features.

Besides these three temporal categories (*past*, *present* or *future*), we have considered the positive, negative and neutral sentiments of the tweets for the fine-grained classification. The user-level tweets with a particular temporal orientation is further subdivided into either positive, negative or neutral sentiment. Finally, we evaluated whether the sentiment view of temporal orientation (i.e. past-positive, past-negative, past-neutral, present-positive, present-negative, present-neutral, future-positive, future-negative, and future-neutral) of the users is related to their several psycho-demographic attributes. In this research, we have considered five psycho-demographic attributes, namely age, eduction, relationship, intelligence, and optimism.

Our contributions are summarised as below:

- We introduce the sentiment dimensions in the human temporal orientation to infer the social media users' psycho-demographic attributes on a large-scale.
- We propose a minimally supervised approach to the temporal orientation classification task that leverages large quantities of unlabeled data and requires no hand-annotated training corpora. The empirical evidence shows that the method performs reasonably well.
- We create a gold-standard temporal orientation tweet corpus.
- We define a way to find a novel association between the sentiment view of temporal orientation and the different psycho-demographic factors of the tweet users.

[1] https://dev.twitter.com/streaming/overview. Last accessed on 10-01-2018.

2 Related Background

The temporal study has recently received an increased attention in several application domains of Natural Language Processing (NLP) and Information Retrieval (IR). The introduction of the TempEval task (Verhagen et al., 2009) and the subsequent challenges i.e. TempEval-2 and -3 (Verhagen et al., 2010; UzZaman et al., 2013) in the Semantic Evaluation workshop series have clearly established the importance of time in dealing with the different NLP tasks. Alonso et al. (2011) reviewed the current research trends and presented a number of interesting applications along with the open problems.

The shared task like the NTCIR-11 Temporalia task (Joho et al., 2014) further pushed this idea and proposed to distinguish whether a given query is related to *past*, *recency*, *future* or *atemporal*. It is the first such challenge, which is organized to provide a common platform for designing and analyzing the time-aware information access systems. In parallel, new trends have emerged in the context of the human temporal orientation (Schwartz et al., 2013; Sap et al., 2014; Park et al., 2015; Schwartz et al., 2015; Park et al., 2017). The underlying idea is to understand how the past, present, and future emphasis in the text may affect people's finances, health, and happiness. For that purpose, the temporal classifiers are built to detect the overall temporal dimension of a given sentence. For instance, the following sentence "*can't wait to get a pint tonight*" would be tagged as *future*.

In summary, most of the temporal text processing applications have been mainly relying on the rule-based time taggers, for e.g. HeidelTime (Strötgen and Gertz, 2015) or SUTime (Chang and Manning, 2012) to identify and normalize time mentions in the texts. Although interesting results have been reported (UzZaman et al., 2013), but the coverage is limited to the finite number of rules they implement.

The time perspective and its importance in various social science and psychological studies is well established in literature. It plays a fundamental role in our interpersonal relation influenced by cognitive process (Zimbardo and Boyd, 2015).

This is also useful in forming goals, expectations and imaginations. Time perspective is a fundamental process, which intern, influenced by many user attributes such as age, religion, education etc. In their research, Zimbardo and Boyd (2015) have shown that the negative view of the past is related to depression, anxiety, unhappiness, and low self-esteem but the positive view of the past is related to self-esteem and happiness. The hedonistic view of the present is related to novelty seeking and sensation seeking whereas the fatalistic view of the present is related to aggression, anxiety and depression. The future is related to conscientiousness but negatively correlated with depression and anxiety.

Another research suggests that the satisfaction with life of the older adults depends on their positive views of past (Kazakina, 1999). In their research, Drake et al. (2008) described that the past-positive is positively correlated to happiness. The link between the past-negative and many psychological distress like depression and anxiety has been well established in literature (Cully et al., 2001). A focus on the future is very effective for functioning positively. The future orientation also helps in better health in later life (Kahana et al., 2005). In a research, George (2009) evaluated that subjective well-being, happiness, psychological well-being, positive effects and morale refer to the positive orientation towards life.

Past research has established that the time perspective is an important factor to determine the human emotional intelligence (Stolarski et al., 2011). In our work, we measure the relationship with different level of intelligence and the sentiment view of the temporal orientation by more objective consent of the time perspective, i.e. temporal orientation from the tweets on the social media. In a social science research, Guthrie et al. (2009) have shown that the future time perspective is associated with the current socioeconomic status, and the past-fatalistic time perspective is associated with the both current and childhood socioeconomic status.

Although these kinds of research exist extensively in the psychological study, it is not well explored with the empirical study using more objective consent of the time perspective, i.e the temporal orientation. As per our best knowledge, only a very few studies exist that focus on the temporal orientation where only the coarse-grained classes have been considered (Schwartz et al., 2015; Park et al., 2017). In these researches, many user attributes were correlated with the temporal orientation which include conscientiousness, age, gender, openness, extraversion, agreeableness, neuroticism, satisfaction with life, depression, IQ, number of friends etc. In our work, we incorporated fine-grained temporal orientation and found the correlation with the users' age, education, relationship, intelligence, and optimism. The fine-grained study of the temporal orientation only existed at the theoretical level validated with very limited user dataset. Besides validating these findings in the empirical way for the large number of users, we also discuss some previously unexplored relationships.

3 Methodology

We first create a deep temporal-orientation classifier to capture the temporal orientation (*past*, *present* and *future*) of the users' tweets. Thereafter we further classify the users' tweets at the fine-grained level by associating sentiment, i.e. positive, negative or neutral for each temporal category. We compare our temporal-orientation classifier with an existing state-of-the-art method.

3.1 Temporal Orientation Classification

The temporal orientation of tweets is defined by classifying each tweet T in one of the temporal categories t, where $t \in \{$ *past, present*, or *future*$\}$. Given the following tweet *"Let me change lanes and turn left legally"*, the temporal orientation classifier should predict it as an instance of *future* orientation. At first we create a temporal oriented tweet dataset in a minimally supervised way by exploiting the hashtag information. Deep Bi-LSTM network is then trained on this dataset. We use LSTM networks (Hochreiter and Schmidhuber, 1997) as these are well known for capturing the long-term dependencies within the text.

Many times we fail to capture the temporal orientation of a text using just the tense information or the existing temporal keywords. In particular, the tweet *"Today I have a plan for a meeting at night."* is future-oriented. Here, the temporal keyword *'Today'* has a time sense of *present* whereas the tense of the verb is also *present*. The deep learning networks have been very useful to correctly capture the temporal dimension of these kinds of tweets. Although the basic Artifi-

cial Neural Networks (ANNs) (Schalkoff, 1997) and Convolutional Neural Network (CNN) (LeCun et al., 1995) capture the temporal orientation of many tweets correctly, they fail to properly identify where the validating temporal information in the tweet has a long dependency between them. For example, the tweet *"Working in the same unit today with different staff was much better."* has temporal orientation as *past*. Here, the word which has a temporal sense (*i.e. working, today, was*) are placed at a distance from each other. This motivates us to use the LSTM network.

Bidirectional Long Short Term Memory Networks (Bi-LSTM): LSTMs are a special kind of recurrent neural network (RNN) capable of learning long-term dependencies in the text by effectively handling the vanishing or exploding gradient problem. The Bidirectional LSTMs (Schuster and Paliwal, 1997) train two LSTMs, instead of one, on the input sequence. The first on the input sequence and the second on a reversed copy of the input sequence. It is designed to capture information of the sequential dataset and maintain the contextual features from the past and the future. This can provide an additional context to the network and result in faster and even fuller learning on the problem without keeping the redundant context information.

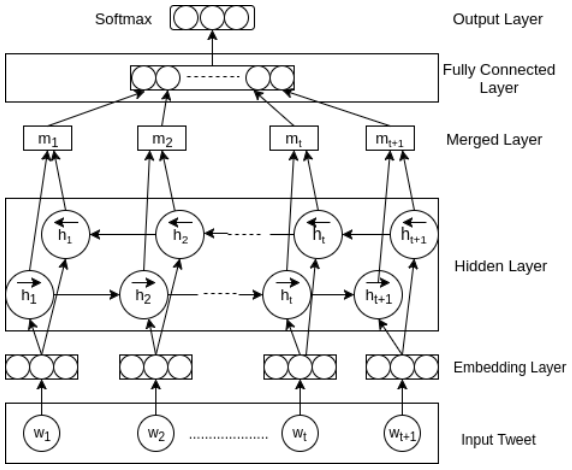


Figure 1: Temporal Orientation learning architecture.

The previous study on the temporal orientation classification based on machine learning includes a supervised classification based on the manually created training set (Schwartz et al., 2015). The multi-class classification was based on a one vs rest approach. But adapting the multiple binary classifiers is not always the best way to deal with a multi-class classification problem. It requires building of three independent classifiers for each temporal category, which consumes more time. Unlike this approach, we incorporate a deep learning-based multi-class classification method for the temporal orientation. The training corpus is generated in a minimally supervised way and fitted to the Bi-LSTM network.

Our experiment uses Bi-LSTM with 200 neurons at the input layer. The loss function we used is *categorical cross-entropy* and the optimizer used is *Root Mean Square Propagation (rmsprop)*. We repeat the training for 100 *number of epochs* with *batch size* set to 128. We also employ dropout (Srivastava et al., 2014) for regularization with a *dropout rate* of 0.2 to prevent over-fitting. All of these attributes are finalized by parameter tuning with the performance obtained on 10-fold cross-validation using the grid search method. Tweet vectors are generated by existing Glove vectors (Pennington et al., 2014) for tweets[2] of 200 dimensions which are trained on 27 billion tweets. We also validate our model on the validation set which was 10% of the training set.

3.2 Sentiment View of Temporal Orientation

We use an existing sentiment classifier available with the NLTK toolkit (Bird, 2006) to classify the *user-level* tweets into positive, negative or neutral.[3] Sentiment is added at the fine-grained level of the temporal orientation. Given the tweets of a user, the sentiment view of temporal orientation of that user is defined by the following equation:

$$orientation_{s,t}(user) = \frac{|tweets_{s/t}(user)|}{|tweets_t(user)|} \quad (1)$$

where, ($t \in$ { *past, present*, or *future*}), and ($s \in$ { *positive, negative*, or *neutral*}), in equation (1). Here, we first classify each user's tweet into the past, present or future temporal category. Then for each temporal category, we find the percentage of each sentiment class (i.e positive, negative or neutral) to obtain the sentiment view of temporal orientation.

We measure the correlation between a user's sentiment view of temporal orientation with their

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

[3] As we are using a sentiment classifier from well known NLTK library, we are not validating it on a manually-tagged test set.

age, education, intelligence, optimism, and relationship using a linear regression (LR) classifier (Neter et al., 1996).

4 Data Sets

For experiments we categorize the datasets into three kinds: training, test and user-level. Training set consists of 27k tweets, whereas the test set is manually annotated with 741 tweets.[4] The user-level tweets consist of ≈10 million tweets from 5,191 users mapped to their user-level features.

4.1 Training Set

Training tweets are collected using the Twitter streaming API.[5] The tweets are collected for the duration of September 2017 and October 2017. We consider day-wise trending topics during this period.[6] We only consider those hashtags which signify a temporal event. Finally, we chose world-wide trending events and collected the tweets based on the hashtags.

The collection of the temporal tweets are based on the following three hypotheses: (a) if a trending topic is of a future event then mostly people would write the futuristic tweets; (b) if a trending topic is about a past incident, then the people would write more about the past but they also write about the present effects of that event; (c) the tweets of trending present event are most critical to handle as besides writing about the present incidents, people always join the links with the past incidents and also give opinion about the future effects.

The task was challenging as the tweets contain a lot of noises and people use various ways to refer to the past, the present and the future. To deal with the pitfalls described in the hypotheses, we filter the tweets using a two-pass filtering method. The method is based on two assumptions (a) every meaningful sentence should contain a verb. (b) mostly past-oriented tweets have tense of the verb as past.

The first assumption is well-established in literature, whereas the second assumption is based on our observation on the tweets and validation against a tense-based classifier. In the first pass of the filtering method, we filter out the tweets which do not contain a verb. The verb part-of-speech tag is determined using the CMU tweet-tagger (Gimpel et al., 2011). In the second pass of the filtering method, we removed the tweets having tense as past from the tweets of the present and future events.

The CMU tweet-tagger does not provide verbs in different sub-categories. For this reason, we also retrieve the Part-of-Speech (PoS) tag information from the Standford PoS-tagger (Manning et al., 2014) for all the tweets to get the sub-categories of verb (i.e. VB, VBD, VBG, VBN, VBP, VBZ). We observed that although Standford PoS-tagger assigned the required verb sub-categories, it also incorrectly tagged some non-verbs as verbs. This is the reason why we considered only those verbs for sub-categorization which were identified (as verbs) by the CMU tweet-tagger.

We varied the training set starting from 3K (equally distributed) to 30K and observed that the accuracy on the gold standard test set did not improve after 27K training instances. Few example tweets with the trending topics are depicted in Table 1.

4.2 Test Set

We evaluate our temporal-orientation classifier on a manually created test set. To get proper assessment on the user-level test set, we randomly selected 800 tweets from the user-level test tweets. Three annotators (post-graduate level students) were asked to tag the tweets in one of the four available classes, namely *past*, *present*, *future* and *other*. The annotation guidelines were as follows:

- Tag a tweet as *past* if it talks about an event which has started as well as ended or the underlaying temporal connotation of the tweet refers to the *past* time.
- Tag a tweet as *present* if it talks about an event which started but not ended yet or the tweet has a *present* temporal connotation,
- Tag a tweet as *future* if it talks about an event which is yet to happen.
- Tag a tweet as *other* in case they found it difficult to get the exact temporal tag for the tweets.

[4] All the developed resources are available at http://www.iitp.ac.in/~ai-nlp-ml/resources.html

[5] https://developer.twitter.com/en/docs

[6] The reason for this selection strategy was the fact that during the passage of time, the future events become present event and the present event becomes past event.

Temporal Orientation	Hashtag	Example Sentence
Past	#Elections2016	did it have influence? of course it did.
	#CPC17	just heard gazza made a guest appearance outside the tory conference.
Present	#HappyHalloween	when you leave for work early but atlanta traffic has other ideas.
	#LHHH	i am trying to figure out who this is
	#WorldTeachersDay	hats off to all the teachers who work hard to not only educate but protect kids everyday.
Future	#U17WC	2017 fifa u17 world cup starts in 3 days
	#2Point0	gonna treat us with 3d visual extravaganza!
	#FutureDecoded	want to get your hands on a new? enter our giveaway at for your chance to win.

Table 1: Example tweets for different temporal orientation categories with trending topics.

We measured the multi-rater kappa agreement (Fleiss, 1971) among the annotators and it was found to have a substantial kappa value of 0.82. The higher kappa value indicates that associating text with temporal dimensions, namely, *past, present, future,* and *other* is relatively straightforward task for humans by using world knowledge than words (Dias et al., 2014). Moreover, our inter-annotator agreement value is in line with the literature.[7] Finally, we select the temporal class of a tweet based on the majority voting among the annotators. The distribution of annotated tweets is as follows:

- *Past*- 375 Tweets
- *Present*- 164 Tweets
- *Future*- 202 Tweets
- *Other*- 59 Tweets

We removed tweets tagged as *Other* and used 741 tweets as the test set.[8]

4.3 User-level Test Set

The user-level tweets consist of ≈10 million tweets from 5,191 users mapped to their user-level psycho-demographic features developed by Preoţiuc-Pietro et al. (2015) are used for this current work. In particular, we use five psycho-demographic attributes such as *age*, *education*, *intelligence*, *optimism*, and *relationship* for our experiment. The users' psycho-demographic features are automatically deduced based on the users' published texts. Preoţiuc-Pietro et al. (2015) used a predictive model to automatically infer user-level features. The method uses various user properties (annotated using crowdsourcing) including age, gender, income, education, relationship status, optimism and life satisfaction as well as all the tweets published by a user to infer user-level features.

5 Experimental Results

We first evaluate our temporal orientation classifier which measures the orientation of each tweet as either of *past*, *present* or *future*. The classifier was trained on the training set and evaluated on the test set. We obtain the highest accuracy of 78.27% over 741 test samples. For comparative evaluation, we can consider a strong baseline system proposed by Schwartz et al. (2015). The baseline system was built following a supervised learning strategy over different features such as ngrams, time expression, PoS tags, tweet length, and temporal class-specific lexicons. The system achieved an accuracy of 71.8% when tested over 500 manually annotated data. The baseline was not reproducible as both the training and test set were manually tagged and the datasets were not available.[9] The baseline model was constructed using the manually annotated data, creation of which involved considerable efforts and expenses. In contrast, we follow a minimally supervised method (does not incur any manual effort) to create our own datasets which are of acceptable quality. We show the results in Table 2.

Orientation	Precision	Recall	F1-measure
Past	81.75	92.0	86.57
Present	79.04	50.61	61.71
Future	71.02	75.24	73.07

Table 2: Precision, Recall and F1-measure of our proposed temporal orientation classification model on manually annotated test data.

[7] Inter-annotator agreement value for the same task in Schwartz et al. (2015) is 0.83.

[8] We only considered *past*, *present* and *future* classes for the reason justified in Schwartz et al. (2015).

[9] We approached the authors of Schwartz et al. (2015) for the data. They did not share the data due to copyright issues. This is the reason for generating our own gold-standard test set.

Results in Table 2 show that the past class is the most correctly classified followed by the future and the present. We observe low recall for the *present* class as many present tweets were mis-classified into either *past* or *present* class. The confusion matrix is shown in Figure 2. The

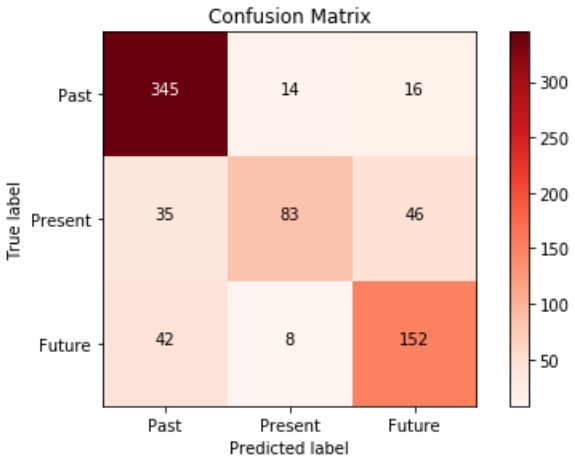


Figure 2: Confusion matrix for the temporal orientation classification.

present class is mis-classified into future when the tweet is of the declarative type. For example, the tweet "*Its not a casserole as theres no binding matrix*" has *present* orientation but our classifier classifies it as of *future* orientation. Another reason could be the fact that the words in the sentence representing *present* temporal orientation are not in the correct form (Its, theres). The *present* classes are mis-classified into the past classes in those cases where mainly the existence of the tense of a verb is *past* but actually the tweet has *present* orientation. For example, the tweet "*For me gloves and mitts made for Cross Country skiing work well for ventilated warmth*" is mis-classified into *past* because of the existence of the word (made) which has tense as *past*. The tweets with *future* orientation are mostly mis-classified into *past* orientation. These kinds of mis-classification is due to either for the presence of past tense or the tweet is a compound sentence which has an independent clause of the past orientation. For example, the tweet "*Hoping to have fun among my friends but wishing I were with you instead*" has a future orientation but it is mis-classified into the past orientation.

We measure the potential limitations of the NLTK sentiment classifier on 100 randomly selected tweets from the test set. The manual observation shows that the classifier generally misclassifies where the sentiment of the tweet is not well understood (example: "Big Trucks parked all over). In some cases, the tweets having conflict sentiment are mis-classified in either of positive or negative class. For example, the tweet "I am very sorry that is a working weekend for me but thanking you very much for the invitation has a conflict sentiment but the classifier classified it into an instance of negative sentiment.

We measure the predictive power of the sentiment view of temporal orientation by performing regression on different psycho-demographic factors. The correlation results between the users' sentiment view of temporal orientation and their psycho-demographic factors using linear regression are presented in Table 3 and Table 4. The performance is measured using a standard metric, namely *Pearson's correlation coefficient r* between the inferred and the target values. All the results in Table 3 and Table 4 are statistically significant when tested against null hypothesis (p_value <0.05).

6 Discussion and Findings

We measure the correlation between the Twitter users' psycho-demographic features and their sentiment view of temporal orientation. In this section, all the discussions and analyses are based on the correlation results over the user test set.

6.1 Demographic Correlates

We select 'age', 'education' and 'relationship' as demographic features for this study. The correlation coefficients between the users' demographic attributes and their sentiment view of temporal orientation are shown in Table 3.

Results in Table 3 demonstrate that any user's past-orientation is significantly correlated (0.4677) with their age. In other words, it suggests that when people age they think more about the past than the present and future. To the best of our knowledge, psychology literature (Nurmi, 2005; Steinberg et al., 2009) has not established the correlation of the past orientation with age. Our finding is consistent with a recent computational study on the human temporal orientation (Schwartz et al., 2015) which shows positive correlation between age and the past orientation. However, we also observed that the users' age has the highest positive correlation (0.5235) with the

User Attribute	Past	Past-Pos	Past-Neg	Past-Neu	Present	Present-Pos	Present-Neg	Present-Neu	Future	Future-Pos	Future-Neg	Future-Neu
Age	**0.4677**	0.3736	-0.0639	-0.3086	0.0802	0.4392	-0.0538	-0.3635	-0.4547	**0.5235**	-0.0186	-0.4590
Education:degree	-0.0577	-0.0281	-0.1340	0.0853	**0.0347**	-0.0402	-0.1588	**0.1013**	0.0340	-0.0393	-0.1470	0.0807
Education:graduate_degree	-0.2214	-0.1837	-0.2136	0.2625	-0.0082	-0.2139	-0.2454	**0.2898**	**0.2004**	-0.2259	-0.2603	0.2817
Education:high_school	**0.1137**	0.0780	0.1748	-0.1488	-0.0264	0.0970	**0.2048**	-0.1702	-0.0878	0.0997	0.1994	-0.1507
Relationship:divorced	-0.3100	-0.2414	-0.2106	0.3139	-0.0299	-0.2946	-0.2425	0.3596	0.2898	-0.3139	-0.2654	**0.3614**
Relationship:in_a_relationship	0.0306	0.0240	**0.0742**	-0.0560	0.0169	0.0208	0.0596	-0.0431	-0.0355	0.0326	0.0664	-0.0496
Relationship:married	-0.0859	-0.0385	-0.1593	0.1075	0.0173	-0.0605	-0.1800	**0.1279**	0.0677	-0.0546	-0.1812	0.1049
Relationship:single	0.1280	0.0822	0.1613	-0.1479	-0.0107	0.1082	0.1936	-0.1755	-0.1082	0.1069	**0.1866**	-0.1531

Table 3: Correlation between users sentiment view of temporal orientation and their different demographic features using LR. Here, pos-positive, neg-negative, neu-neutral.

User Attribute	Past	Past-Pos	Past-Neg	Past-Neu	Present	Present-Pos	Present-Neg	Present-Neu	Future	Future-Pos	Future-Neg	Future-Neu
Intelligence:below_average	-0.0565	-0.0680	0.0724	0.0289	-0.0370	-0.0691	**0.0885**	0.02401	0.0685	-0.0718	0.0792	0.0391
Intelligence:average	**0.1777**	0.1422	0.0996	-0.1725	0.0237	0.1604	0.1105	-0.1868	-0.1694	**0.1697**	0.1277	-0.1904
Intelligence:much_above	-0.2946	-0.2466	-0.2123	0.3198	-0.0333	-0.2928	-0.2451	0.3590	**0.2778**	-0.3122	-0.2736	**0.3625**
Optimism:optimist	0.0696	0.1397	-0.0173	-0.1212	0.0097	0.1486	-0.0144	-0.1245	0.0666	**0.1604**	-0.0023	-0.1417
Optimism:pessimist	0.0536	-0.0670	0.1695	-0.0167	-0.0110	-0.0430	**0.1815**	-0.0375	-0.0420	-0.0415	0.1708	-0.0164
Optimism:neither	-0.0550	-0.1027	0.0014	0.0944	-0.0033	-0.1132	-0.0010	0.0999	0.0504	-0.1224	-0.0103	0.1119

Table 4: Correlation between users sentiment view of temporal orientation and their different psychological features using LR. Here, pos-positive, neg-negative, neu-neutral.

future-positive. It indicates that people become positively future-orientated when they age, though not surprising, yet somewhat novel. The results indicate that only considering the temporal orientation without the sentiment dimensions can be sometimes misleading as we can observe that the negative future-orientation has a negative correlation (-0.4547) with age while the future-positive has a positive correlation with age.

Figure 3 explains how the trends of the sentiment view of temporal orientation varies from age 10 to 60. We observe that for all the temporal classes, the positive sentiment increases rapidly with the increase of age. Most interestingly, for all the temporal orientation people become negative up to the age of 28 and then their negative sentiment steadily reduces. We also observe that human's neutral sentiment rapidly decreases up to the age of 27 and then it decreases steadily.

The second demographic attribute we considered is 'education'. We measure the correlation between the temporal orientation and three different levels of education: *degree*, *graduate_degree*, and *high_school*. In the psychological literature (Horstmanshof and Zimitat, 2007; Richardson et al., 2012), it was reviewed that the students' temporal orientations is a new dimension to approaches in enhancing student engagement in the academics. It was found that the first year students of university were more *future-oriented* rather than present or past oriented. From our results in Table 3, we found that the users who have education of *degree* level are *present-oriented*. But interestingly they neither think positively nor negatively-they express more neutral sentiment. Users with education of *graduate_degree* are found to be *future-oriented*. Here, the fine-grained classification suggests that they also express the *neutral* sentiment. Interestingly, we found that the users with education of *high_school* had *positive* correlation with *past orientation*. However, when we considered the sentiment dimension, we found that it was actually correlated with *present orientation* with *negative sentiment*.

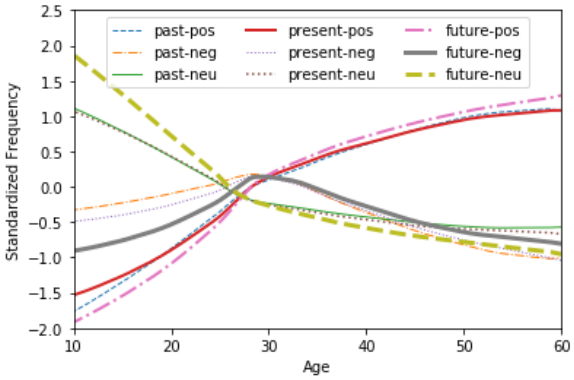


Figure 3: Standardized sentiment view of temporal orientation of the users over their age. Smoothing was done using loess smoothing estimates. Here, pos-positive, neg-negative, neu-neutral.

Our third and final demographic feature 'relationship' is categorized into four types in our current study: *divorced*, *in_a_relationship*, *married*, and *single*. From the results in Table 3, we observe that the users who are *divorced* found to be more future oriented and they seem to express the neu-

tral sentiment. The *in_a_relationship* users seem to be more past-oriented. They also found to be negative minded. Married people are found to be present oriented while expressing the neutral sentiment. The users who are single, are generally futuristic but they are negative about it.

6.2 Psychological Correlates

We chose two psychological factors, intelligence and optimism. The correlation coefficient results are shown in Table 4.

The intelligence level of the users was measured in three sub-categories: *intelligence:below_average*, *intelligence:average* and *intelligence:much_above*. Some novel findings have been observed through our results. We found a modest yet significant positive correlation between intelligence below_average and negative view of the present orientation. It suggests that the users having intelligence below average are present oriented but they seem to have negative view of it. Surprisingly, we found that average intelligent users are past-oriented but considering the sentiment dimension they seem to be more future-positive. However, this should be validated with further investigation. We found that the users who have intelligence much_above are more future orientated. Interestingly, we found a negative correlation with the future-positive. However, we found a positive correlation (0.3614) with the future-neutral which suggests that the users with much above intelligence are futuristic and they express a neutral view.

We chose three categories of optimism: optimistic, pessimistic, and neither for our observation. The result shown in Table 4 suggests that the optimistic people are future oriented. They also seem to have positive sentiment. Although the link between the future orientation and optimism is well established in literature (Lennings, 2000; Busseri et al., 2013), there is no empirical study for a large number of users. We find a relatively higher positive correlation between the pessimist and the present-negative which suggests that the pessimistic people are negative minded and focus more on present. People who are neither optimistic nor pessimistic are found to be future oriented with the neutral sentiment which is also a novel finding.

7 Conclusion and Future Work

This paper presents a first large-scale study to associate the psycho-demographic profile of the Twitter users with their sentiment view of temporal orientation based on language they use in Twitter. We first detect the temporal orientation of the tweets using the Bi-LSTM based temporal orientation classifier. We generated the temporal categories of our training set in a minimally supervised way. We created a benchmark dataset for the evaluation of our temporal orientation classifier. The temporal orientation classifier achieved an accuracy of 78.27% when run on the manually tagged test data. We added the sentiment dimension at the fine-grained level of the temporal orientation. The associations between the users' sentiment view of temporal orientation and their different psycho-demographic attributes (age, education, intelligence, optimism, and relationship) are somewhat novel in the context of the computational social science studies. Whereas the study on the temporal orientation concentrated on a coarse-grained level, we focused on the fine-grained level of temporal orientation which opens more aspects of the social, economic, and psychological research which was not possible previously on a large scale.

Acknowledging the possible limitations of this study including the quality of the sentiment classifier and a low recall of the present temporal orientation, in future, we will consider more sophisticated sentiment classifier for better performance and explore more linguistic insight into consideration to improve the performance of the temporal orientation classifier. We also like to extend our work with the link to more behavioral study and analysis.

8 Acknowledgments

Asif Ekbal acknowledges Young Faculty Research Fellowship (YFRF), supported by Visvesvaraya PhD scheme for Electronics and IT, Ministry of Electronics and Information Technology (MeitY), Government of India, being implemented by Digital India Corporation (formerly Media Lab Asia). Mohammed Hasanuzzaman and Andy Way would like to acknowledge ADAPT Centre for Digital Content Technology, funded under the SFI Research Centres Programme (Grant 13/RC/2106) and is co-funded under the European Regional Development Fund.

References

Jean Adams and Daniel Nettle. 2009. Time Perspective, Personality and Smoking, Body Mass, and Physical Activity: An Empirical Study. *British journal of health psychology* 14(1):83–105.

Omar Alonso, Jannik Strötgen, Ricardo A. Baeza-Yates, and Michael Gertz. 2011. Temporal Information Retrieval: Challenges and Opportunities. In *WWW2011 Workshop on Linked Data on the Web*. Hyderabad, India, pages 1–8.

Iyad Batal, Dmitriy Fradkin, James H. Harrison Jr., Fabian Moerchen, and Milos Hauskrecht. 2012. Mining Recent Temporal Patterns for Event Detection in Multivariate Time Series Data. In *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Beijing, China, pages 280–288.

Steven Bird. 2006. NLTK: The Natural Language Toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*. Sydney, Australia, pages 69–72.

Michael A Busseri, Annette Malinowski, and Becky L Choma. 2013. Are Optimists Oriented Uniquely Toward the Future? Investigating Dispositional Optimism from a Temporally-expanded Perspective. *Journal of Research in Personality* 47(5):533–538.

Angel X. Chang and Christopher D. Manning. 2012. SUTime: A Library for Recognizing and Normalizing Time Expressions. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation*. Istanbul, Turkey, pages 3735–3740.

Munmun De Choudhury, Scott Counts, and Eric Horvitz. 2013. Predicting Postpartum Changes in Emotion and Behavior via Social Media. In *2013 ACM SIGCHI Conference on Human Factors in Computing Systems*. Paris, France, pages 3267–3276.

Jeffrey A Cully, Donna LaVoie, and Jeffrey D Gfeller. 2001. Reminiscence, Personality, and Psychological Functioning in Older Adults. *The Gerontologist* 41(1):89–95.

Gaël Harry Dias, Mohammed Hasanuzzaman, Stéphane Ferrari, and Yann Mathet. 2014. TempoWordNet for Sentence Time Tagging. In *Proceedings of the 23rd International Conference on World Wide Web*. New York, USA, pages 833–838.

Peter Sheridan Dodds, Kameron Decker Harris, Isabel M Kloumann, Catherine A Bliss, and Christopher M Danforth. 2011. Temporal Patterns of Happiness and Information in a Global Social Network: Hedonometrics and Twitter. *PloS one* 6(12):1–26.

Lisa Drake, Elaine Duncan, Fi Sutherland, Clare Abernethy, and Colette Henry. 2008. Time Perspective and Correlates of Wellbeing. *Time & Society* 17(1):47–61.

Joseph L Fleiss. 1971. Measuring Nominal Scale Agreement among Many Raters. *Psychological Bulletin* 76(5):378–382.

Linda K George. 2009. Still Happy after all these Years: Research Frontiers on Subjective Well-being in Later Life. *Journals of Gerontology Series B: Psychological Sciences and Social Sciences* 65(3):331–339.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-Speech Tagging for Twitter: Annotation, Features, and Experiments. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA, pages 42–47.

Lori C Guthrie, Stephen C Butler, and Michael M Ward. 2009. Time Perspective and Socioeconomic Status: A Link to Socioeconomic Disparities in Health? *Social science & medicine* 68(12):2145–2151.

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

Louise Horstmanshof and Craig Zimitat. 2007. Future Time Orientation Predicts Academic Engagement among First-year University Students. *British Journal of Educational Psychology* 77(3):703–718.

Alexander T. Ihler, Jon Hutchins, and Padhraic Smyth. 2006. Adaptive Event Detection with Time-varying Poisson Processes. In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Philadelphia, PA, USA, pages 207–216.

Hideo Joho, Adam Jatowt, Roi Blanco, Hajime Naka, and Shuhei Yamamoto. 2014. Overview of NTCIR-11 Temporal Information Access (Temporalia) Task. In *Proceedings of the 11th NTCIR Conference on Evaluation of Information Access Technologies*. Tokyo, Japan, pages 429–437.

Eva Kahana, Boaz Kahana, and Jianping Zhang. 2005. Motivational Antecedents of Preventive Proactivity in Late Life: Linking Future Orientation and Exercise. *Motivation and emotion* 29(4):438–459.

Elena Kazakina. 1999. *Time Perspective of Older Adults: Relationships to Attachment Style, Psychological Well-being, and Psychological Distress*. Ph.D. thesis, ProQuest Information & Learning.

Michal Kosinski, David Stillwell, and Thore Graepel. 2013. Private Traits and Attributes are Predictable from Digital Records of Human Behavior. *Proceedings of the National Academy of Sciences* 110(15):5802–5805.

Yann LeCun, Yoshua Bengio, et al. 1995. Convolutional Networks for Images, Speech, and Time Series. *The handbook of brain theory and neural networks* 3361(10):276–279.

CJ Lennings. 2000. Optimism, Satisfaction and Time Perspective in the Elderly. *The International Journal of Aging and Human Development* 51(3):167–181.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Baltimore, MD, USA, pages 55–60.

James Marquardt, Golnoosh Farnadi, Gayathri Vasudevan, Marie-Francine Moens, Sergio Davalos, Ankur Teredesai, and Martine De Cock. 2014. Age and gender identification in social media. In *Working Notes for CLEF 2014 Conference*. Sheffield, UK, pages 1129–1136.

John Neter, Michael H Kutner, Christopher J Nachtsheim, and William Wasserman. 1996. *Applied Linear Statistical Models*, volume 4. Irwin, Chicago.

Jari-Erik Nurmi. 2005. Thinking about and Acting upon the Future: Development of Future Orientation across the Life Span. *Understanding behavior in the context of time: Theory, research, and application* pages 31–57.

Gregory Park, H Andrew Schwartz, Johannes C Eichstaedt, Margaret L Kern, Michal Kosinski, David J Stillwell, Lyle H Ungar, and Martin EP Seligman. 2015. Automatic Personality Assessment through Social Media Language. *Journal of personality and social psychology* 108(6):934–952.

Gregory Park, H Andrew Schwartz, Maarten Sap, Margaret L Kern, Evan Weingarten, Johannes C Eichstaedt, Jonah Berger, David J Stillwell, Michal Kosinski, Lyle H Ungar, et al. 2017. Living in the Past, Present, and Future: Measuring Temporal Orientation With Language. *Journal of personality* 85(2):270–280.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pages 1532–1543.

Daniel Preoţiuc-Pietro, Svitlana Volkova, Vasileios Lampos, Yoram Bachrach, and Nikolaos Aletras. 2015. Studying User Income through Language, Behaviour and affect in Social Media. *PloS one* 10(9):1–17.

Michelle Richardson, Charles Abraham, and Rod Bond. 2012. Psychological Correlates of University Students' Academic Performance: a Systematic Review and Meta-analysis. *Psychological bulletin* 138(2):353–387.

Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2013. Tweet Analysis for Real-time Event Detection and Earthquake Reporting System Development. *IEEE Transactions on Knowledge and Data Engineering* 25(4):919–931.

Maarten Sap, Gregory J. Park, Johannes C. Eichstaedt, Margaret L. Kern, David Stillwell, Michal Kosinski, Lyle H. Ungar, and H. Andrew Schwartz. 2014. Developing Age and Gender Predictive Lexica over Social Media. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Doha, Qatar, pages 1146–1151.

Robert J Schalkoff. 1997. *Artificial Neural Networks*, volume 1. McGraw-Hill, New York.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional Recurrent Neural Networks. *IEEE Transactions on Signal Processing* 45(11):2673–2681.

H Andrew Schwartz, Johannes C Eichstaedt, Margaret L Kern, Lukasz Dziurzynski, Stephanie M Ramones, Megha Agrawal, Achal Shah, Michal Kosinski, David Stillwell, Martin EP Seligman, et al. 2013. Personality, Gender, and Age in the Language of Social Media: The Open-vocabulary Approach. *PloS one* 8(9):1–16.

H. Andrew Schwartz, Greg Park, Maarten Sap, Evan Weingarten, Johannes Eichstaedt, Margaret Kern, Jonah Berger, Martin Seligman, and Lyle Ungar. 2015. Extracting Human Temporal Orientation in Facebook Language. In *Proceedings of The 2015 Conference of the North American Chapter of the Association for Computational Linguistics-Human Language Technologies (NAACL)*. Denver, Colorado, USA, pages 409–419.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a Simple Way to Prevent Neural Networks from Overfitting. *Journal of machine learning research* 15(1):1929–1958.

Laurence Steinberg, Sandra Graham, Lia OBrien, Jennifer Woolard, Elizabeth Cauffman, and Marie Banich. 2009. Age Differences in Future Orientation and Delay Discounting. *Child development* 80(1):28–44.

Maciej Stolarski, Joanna Bitner, and Philip G Zimbardo. 2011. Time Perspective, Emotional Intelligence and Discounting of Delayed Awards. *Time & Society* 20(3):346–363.

Jannik Strötgen and Michael Gertz. 2015. A Baseline Temporal Tagger for all Languages. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal, pages 541–547.

Naushad UzZaman, Hector Llorens, Leon Derczynski, James F. Allen, Marc Verhagen, and James Pustejovsky. 2013. SemEval-2013 Task 1: TempEval-3: Evaluating Time Expressions, Events, and

Temporal Relations. In *Proceedings of the 7th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2013*. Atlanta, Georgia, USA, pages 1–9.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Jessica Moszkowicz, and James Pustejovsky. 2009. The Tempeval Challenge: Identifying Temporal Relations in Text. *Language Resources and Evaluation (LRE)* 43(2):161–179.

Marc Verhagen, Roser Saurí, Tommaso Caselli, and James Pustejovsky. 2010. SemEval-2010 Task 13: TempEval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation, SemEval@ACL 2010*. Uppsala, Sweden, pages 57–62.

Paul Webley and Ellen K Nyhus. 2006. Parents Influence on Childrens Future Orientation and saving. *Journal of Economic Psychology* 27(1):140–164.

Philip G Zimbardo and John N Boyd. 2015. Putting Time in Perspective: A Valid, Reliable Individual-differences Metric. *Journal of Personality and Social Psychology* 77(6):1271–1288.

Querying Word Embeddings for Similarity and Relatedness

Fatemeh Torabi Asr
Simon Fraser University
Buranby, BC, Canada
ftorabia@sfu.ca

Robert Zinkov
University of Oxford
Oxford, UK
zinkov@robots.ox.ac.uk

Michael N. Jones
Indiana University
Bloomington, IN, USA
jonesmn@indiana.edu

Abstract

Word embeddings obtained from neural network models such as Word2Vec Skipgram have become popular representations of word meaning and have been evaluated on a variety of word similarity and relatedness norming data. Skipgram generates a set of word and context embeddings, the latter typically discarded after training. We demonstrate the usefulness of context embeddings in predicting asymmetric association between words from a recently published dataset of production norms (Jouravlev and McRae, 2016). Our findings suggest that humans respond with words closer to the cue within the context embedding space (rather than the word embedding space), when asked to generate thematically related words.

1 Introduction

Modern distributional semantic models such as Word2Vec (Mikolov et al., 2013a,b) and GloVe (Pennington et al., 2014) have been evaluated on a variety of word similarity and relatedness datasets. A considerable amount of attention has been paid to what models and, more recently, what parameter settings and input data produce embedding representations that better reflect similarity/relatedness between words, taking human normative judgments as the gold standard (Baroni et al., 2014; Kiela et al., 2015; Levy et al., 2015; Melamud et al., 2016; Sahlgren and Lenci, 2016).

Similarity between two words is often assumed to be a direction-less measure (e.g., *car* and *truck* are similar due to feature overlap), whereas relatedness is inherently directional (e.g., *broom* and *floor* share a functional relationship). In addition, it is well established in human behavioral data that similarity and relatedness judgments are both asymmetric. For example, humans judge *leopard* to be much more similar to *tiger* than *tiger* is to *leopard* (Tversky and Gati, 1982). A concordant asymmetry is seen in relation tasks: in free association data, *baby* is a much more likely response when cued with *stork* than *stork* would be as a response when cued with *baby* (Nelson et al., 1999). The distinction between similarity and relatedness, and the asymmetry of the judgments have typically been ignored in recent evaluations of popular embedding models.

There is ample experimental evidence in the psycholinguistic literature that similarity and relatedness are both well represented in human behavior (see Hutchison (2003), for a review), and are qualitatively distinct representations or processes. In semantic priming paradigms, a target word is processed more efficiently when briefly preceded by a related or similar word (e.g., *honey-bee* or *wasp-bee*) relative to a neutral or unrelated prime (e.g., chair-bee). Facilitation is seen for word pairs that are purely category coordinates (*lawyer-surgeon*) or purely associates (*scalpel-surgeon*), and pairs that share both types of relations (*nurse-surgeon*) tend to see an additive processing benefit that reflects the privilege of both similarity and relatedness, an effect generally referred to as the "associative boost" (Chiarello et al., 1990; Lucas, 2000). Asymmetries are the norm in semantic priming data, leading to the early theoretical prominence of spreading activation models to account for human data.

Free association data provide complimentary evidence of the qualitative distinction between relatedness and similarity in human memory. In a free association task, participants are provided with a cue word and are asked to rapidly respond with a word that comes to mind first. Huge norms of human responses have been collected over the years; for example, (Nelson et al., 1999) early norms contain three-quarters of a million responses to over 5,000 cue words across 6,000 par-

Proceedings of NAACL-HLT 2018, pages 675–684
New Orleans, Louisiana, June 1 - 6, 2018.

ticipants. More recently, (De Deyne et al., 2016) have more than doubled the size of Nelsons norms in multiple languages by gamifying the task[1] . The majority of responses in free association data are based on thematic relatedness rather than similarity per se (De Deyne and Storms, 2008). As with semantic priming, free association norms are dominated by asymmetric relations: While *stork* has a very high probability of eliciting *baby* as a response across participants, cuing with *baby* brings so many competitors to mind that it is extremely unlikely to respond with *stork* (Hutchison, 2003).

The difficulty of accounting for similarity and relatedness with a single vector representation for each word has led to the suggestion that distinct representations, and perhaps even distinct learning models, are needed for optimal performance on these distinct tasks (Mandera et al., 2017). It may be unrealistic to expect a single vector representation to account for qualitatively distinct similarity and relatedness data. Further, asymmetries in human similarity and relatedness tasks have been used as strong evidence against spatial models of semantics such as word embedding models, and in favor of Bayesian models (Griffiths et al., 2007); but see (Jones et al., 2017). The cosine between two word vectors is inherently symmetric: *leopard-tiger* has the same cosine as *tiger-leopard*.

In order to understand how distributional representation of words reflect similarity and relatedness one should study the algorithms. Each cell of a word vector in a count model indicates the first-order association between the target word and a context word, document, or topic. Dimensionality reduction algorithms are applied to obtain denser representations that can demonstrate second-order relatedness/similarity between words (e.g. applying SVD to PMI matrix). Relative to these classic models, predictive distributional models such as Word2Vec are generally more complicated. Decomposition and interpretation of the neural word embeddings is less straightforward because the final vectors incrementally converge from a predict-and-update process based on a local objective function rather than by global counting or a batch abstraction process. Most evaluative studies of predictive distributional semantics have viewed these models as a black box, considering only at the output vectors. For example, the Word2Vec Skipgram architecture has easily taken the lead and become representative of the predictive distributional semantic models, but little attention has been paid to what statistical information is best represented in the two resulting embedding sets. The Skipgram is a feed-forward network with localist input and output layers, and one hidden layer which determines the dimensionality of the final vectors. It is trained on word-context pairs with an objective function trying to minimize the error of predicting context words within a specific window around the center word. At the end of training, two matrices are produced, one representing **word embeddings** and the other representing **context embeddings** for each and every vocabulary word. While word embeddings have been used as the output of Skipgram in many previous studies, little attention has been paid to the context embeddings and the usefulness of these vectors in performing lexical semantic tasks (Levy et al., 2015; Melamud et al., 2015; Aoki et al., 2017).

Recently, Asr and Jones (2017) used an artificial language to evaluate how hyperparameter settings affected the Skipgrams representation of first- vs. second-order statistical sources. In natural languages, paradigmatic and syntagmatic information sources are non-independent, confounding similarity and relatedness judgments. Words that are more similar tend to also share functional, script, or thematic relations (Hutchison, 2003; Lucas, 2000); e.g., *surgeon-nurse*. Asr and Jones artificial language was engineered to disentangle the two sources of statistical information. Following on suggestions by Levy et al. (2015), Asr and Jones found that averaging context vectors with the word vectors (w+c post-processing) produced optimal organization of the semantic space for both paradigmatic and syntagmatic structure. The goal of the current work is to more systematically explore the integration of word and context vectors in similarity and relatedness data; our two core objectives are:

1. To evaluate the Skipgram model on thematic relatedness production norms, which implicitly manifests asymmetric relations between words compared to the typical evaluation on direction-less similarity/relatedness.
2. To explore novel ways of computing relatedness scores by contributing both word and context embeddings produced by Word2Vecs Skipgram architecture.

[1] https://smallworldofwords.org

2 Similarity vs. Relatedness Data

One of the famous datasets on word similarity/relatedness is Wordsim353 (Finkelstein et al., 2001) including 353 English word pairs, and a revised version (Agirre et al., 2009) splitting similar from related word pairs (WrordSim). These data have been repeatedly used in comparative studies on distributional semantic models. Recently, the division between similarity and relatedness judgments has been highlighted in the literature, resulting in development of new datasets with more specific annotation instructions.

Hill et al. (2015) introduced the SimLex-999 dataset (SimLex) for purified evaluation of word similarity by asking the annotators explicitly not to score based on degree of relatedness. For example, the word pair coast-shore received an average similarity score of 9.00 in SimLex and 9.10 in WordSim, while the related word pair clothes-closet was assigned an average score of 1.96 in SimLex and 8.00 in WordSim. More recently, Jouravlev and McRae (2016) collected pure relatedness data through a production experiment. They presented participants with cue words and instructed them to response only with directly related words and not taxonomically similar words. This database (ProNorm) includes responses to 100 object words, providing us with directional relatedness score for 1,169 word pairs.

The important distinction of SimLex and ProNorm datasets compared to other available similarity/relatedness data is the explicit instruction of participants to pay attention to one aspect of word relations and not the other. The ProNorm dataset, also has an advantage of a more natural setup, where associatively related words were generated by participants, rather than being selected by language experts and only rated by the participants. In this paper, we use ProNorm as the main dataset to investigate how word embedings should be used to measure relatedness between two words and how the free recall experiment can be simulated for the model. The SimLex dataset is used to set a baseline for comparison against the similarity measurement task, which is the most common intrinsic benchmark for evaluation of word embeddings. Finally, we use the WordSim dataset to explore whether the observed differences between vector-based measures of similarity and relatedness come out if the benchmark data is collected in implicit setup, where participants did not know they were rating for similarity or relatedness.

3 Word and Context Embeddings

Word embeddings produced by the Skipgram architecture have been used in many previous studies as the word meaning representation and are the main output of the model. In the original implementation of Word2Vec, the context embeddings (weights on the hidden to output layer of the neural network) were discarded after learning was complete. Inspired by Pennington et al. (2014) in the architecture of the GloVe model, Levy et al. (2015) proposed that the final word embeddings in Word2Vec could be obtained from the average of word and context embeddings. They implemented word + context (w+c) as a useful post processing option for the Word2Vec Skipgram algorithm in their published version of the model[2]. The w+c option allows computation of word similarity based upon both first and second-order co-occurrence information. The cosine similarity between two words based on the dot product of their w+c embeddings, which we call the AA measure (*A* standing for the average of word and context embeddings of a word), includes the following terms:

$$cos(a,b) = \frac{W_aW_b + C_aC_b + W_aC_b + C_aW_a}{2\sqrt{W_aC_a + 1}\sqrt{W_bC_b + 1}} \quad (1)$$

While traditional measures, i.e., WW (cosine similarity of the word embeddings), and AA (cosine similarity of the word+context embeddings) are suitable predictors for words similarity, we hypothesize that the asymmetric measures WC (word embedding of the first word and context embedding of the second) and CW (context embedding of the first word and word embedding of the second) should be better indicators of relatedness. This decomposition of similarity measures is especially useful when asymmetric associations between words are being inferred: the asymmetric measures reserve the direction and the type of relation: WC reflects the likelihood of the second word occurring in the context of the first word, and CW reflects the likelihood of the first word occurring in the context of the second word. These two quantities are different, given that the W and C matrices are obtained from two different layers

[2] https://bitbucket.org/omerlevy/hyperwords

of the neural network, one connected to the input layer and the other to the output layer.

4 Experiments

SimLex and ProNorm provide complementary scores on similarity and relatedness between words. In order to demonstrate and examine how word embeddings should be used in asymmetric relatedness measurement, we designed two experiments. In both experiments word and context embeddings were obtained from Skipgram models trained on a tokenized English Wikipedia dump[3]. We slightly modified the original Word2Vec Skipgram implementation by Levy et al. (2015) to save both word and context vectors.

We tested vector spaces with varying dimensionalities (*dim*=100/200/300) and number of context words (*win*=3/6/10), as well as minimum occurrence cutoff (*min*=1/5), negative samples (*neg*=1/5) and iterations (*iter*=1/5). These variations were tested to ensure the observed patterns reported in the experiments, but we report numerical results only for best performing models. In particular, higher dimensional vectors with *dim*=300 produced consistently better alignment with human scoring data. We also found *min*=1, *neg*=5 and *iter*=5 to be the optimal parameter settings across all experiments.

5 Quantitative Evaluation

Our first experiment follows an established evaluation strategy by computing the Spearman correlation coefficient between the set of similarity measures produced by the word embedding model (WW/CC/WC/CW/AA) and the similarity/relatedness scores taken from the SimLex and ProNorm datasets. As ProNorm score of a word pair (w_1, w_2), we simply use the total number of times a response word w_2 was produced by all subjects given w_1 as a cue word. Interested readers are encouraged to see Jouravlev and McRae (2016) for more details on the data collection procedure.

Our hypothesis is that for taxonomic similarity judgment the classic WW measure, i.e., the cosine of the word vectors of w_1 and w_2 would perform best, especially given the fact that in collection of similarity norms the direction between two words was not a factor. For explicit relatedness judgment, on the other hand, we expect one of the asymmetric measures to be the best predictor. WC, which is the cosine between the word embedding of the cue w_1 and the context embedding of the response w_2 tells us how likely we would see w_2 and similar words in the context of w_1. CW reflects the opposite way relatedness, meaning how likely it is to see w_1 and similar words in the context of w_2. Note that these two quantities are different both mathematically and conceptually, because they are obtained from generalization over word occurrences in many different contexts. We hypothesize that WC should be the best predictor for the ProNorm score of (w_1, w_2) given that production in the constrained setup of the ProNorm experiment was guided by thematic relatedness, making it more like a non-syntactic language modeling task: guessing which other words/concepts might appear within the context of the current word.

SimLex and ProNorm collections have almost the same number of word pairs. However, it is important to note that ordering ProNorm word pairs based on their relatedness scores is probably more difficult than ordering the SimLex list of word pairs. This is because in the ProNorm data collection setup, all word pairs were basically generated based on relatedness, whereas in SimLex, experimental items were pre-designed in a way they covered a wide range of closely similar to totally different word pairs. Ordering SimLex should in turn be harder than ordering words in the old WordSim353 similar and related word pair collections, because each of the latter subsets has a much smaller number of items compared to SimLex collection.

In order to demonstrate the difference between the tasks of ordering words based on similarity vs. relatedness in an explicit setup (SimLex and ProNorm) with an implicit, i.e., a mixed setup we include WordSim353 (Agirre et al., 2009) in our experiment. We hypothesize that the patterns of superiority of one vector-based measure to another in ranking word pairs based on their similarity and relatedness should come out even if people were not explicitly instructed to pay attention to a specific aspect.

5.1 Results

Table 1 displays correlation scores between similarity ratings in SimLex and Skipgram similarity measures introduced in the previous section (all

[3] https://sites.google.com/site/rmyeid/projects/polyglot

Measure	300-3	300-6	300-10
WW	**0.44**	**0.42**	**0.41**
CC	0.40	0.40	0.39
WC	0.34	0.36	0.37
CW	0.32	0.36	0.35
AA	0.42	0.41	**0.41**
AllReg	**0.46**	**0.43**	**0.41**

Table 1: Spearman correlation between human similarity judgments (SimLex) and Skipgram measures.

Measure	300-3	300-6	300-10
WW	0.19	0.22	0.23
CC	0.18	0.19	0.20
WC	**0.24**	**0.25**	**0.26**
CW	0.20	0.20	0.20
AA	0.20	0.22	0.22
AllReg	**0.24**	**0.27**	**0.27**

Table 2: Spearman correlation between forward relatedness scores (ProNorm) and Skipgram measures.

significant at $p < 0.001$). Results on models with *dim*=300 and *win*=3/6/10 are reported (see Appendix for supplementary results). The WW measure exhibits consistently a better alignment with the human rating data compared to the all other measure. This suggests that second-order co-occurrence information plays the main role in similarity between two words. In collection of SimLex, subjects were asked explicitly not to rate similarity based on thematic relatedness. It is likely that the human ratings were affected not only by co-occurrence information encoded in word embeddings but also in context embeddings. As we expected, the best predictors of this data are the symmetric similarity measures, and in particular, WW. The last row of the table includes Spearman correlation between human similarity judgment and a linear regression model using all Skipgram measures as predictors. Thus, numbers in this row show an upper bound for Spearman scores of the individual measures (obtained from an optimal weighting of all individual measures).

Table 2 shows the Spearman correlation between ProNorm scores and the Skipgram measures (all significant at $p < 0.001$). As we hypothesized, WC stands out as the best predictor, suggesting that human responses to a cue word (when asked to name related words) are more likely to be found in the vicinity of the cue word within the context embedding space rather than within the word embedding space. The correlation between the ProNorm scores with WC is larger than with WW or AA scores. This indicates the importance of the knowledge encoded in the context embeddings, but specifically the prediction power of the asymmetric similarity measure compared to the symmetric ones. Interestingly, CW is not as good as WC in this task. This reveals the importance of the direction in associative relatedness between words such as baby and stork, which seems to correlate with their vector representations. Finally, the regression model, which applies an optimal weighing on different Skipgram measures finds the best fit, whereas AA which gives equal weights to symmetric and asymmetric measures fails to compete with WC alone. Comparisons between Tables 1 and 2 suggest that, similarity and relatedness are best approximated by symmetric and asymmetric measures, respectively.

We next examined the WordSim353 data to evaluate whether above implications apply also to ratings collected in implicit setup, i.e., where human subjects were not instructed to response based either on taxonomic similarity or associative relatedness. We examine each subset of WordSim353 separately and treat them like similarity and relatedness data. Table 3 shows results on these two collections of word pairs with best parameter setup; i.e., with *dim*=300 and *win*=3 and 6[4]. Similar to our previous experiments on the other datasets, relative ranking of similar word pairs is best predicted with commonly used measure WW alone, which is indicative of second-order co-occurrence similarity. For related word pairs, asymmetric measures WC and CW, which are indicative of first-level co-occurrence come out as better individual predictors compared to WW. However, the balanced combination of all, i.e., the AA measure seems to be the consistent winner across both datasets. This finding suggests that when similarity/relatedness is scored by people as an overall degree of closeness between words and without explicit instruction to focus on one aspect, the most reliable predictor would be a cosine measure that considers both symmetric and asymmetric types of relations between words.

[4]Results for *win*=10 were not as good as in other conditions for this experiment, therefore we only report the very best setups with *win*=3 and 6.

	Similar pairs		Related pairs	
	300-3	300-6	300-3	300-6
WW	0.79	**0.81**	0.62	0.63
CC	0.76	0.77	0.58	0.61
WC	0.76	0.79	**0.65**	**0.69**
CW	0.76	0.78	**0.65**	0.67
AA	**0.80**	**0.81**	**0.65**	0.67
AllReg	**0.80**	**0.82**	**0.69**	**0.70**

Table 3: Spearman correlation between scores from WordSim353 subsets and Skipgram measures.

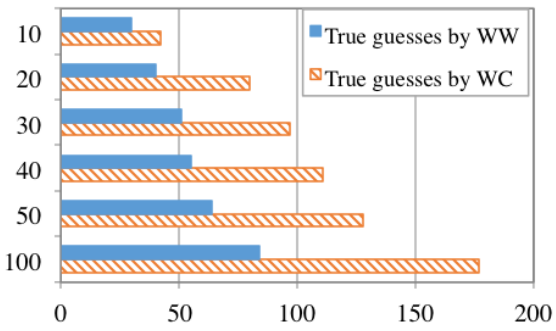


Figure 1: Number of human responses found in word and context embedding spaces near the word embedding of the cue (x-axis) as the search space is increased (y-axis).

6 Qualitative Evaluation

Our first experiment focused on discovering the best vector-based predictor for similarity and relatedness between two words. We found that considering context vectors in calculation of the similarity score produces a superior predictor, specially for relatedness, compared to the traditionally used measure (WW) based only on word vectors. The experiment in this section is a more tangible evaluation of the Word2Vec model in a relatedness task when a cue word is given. The aim is to simulate the production experiment with which the ProNorm data were collected and to evaluate whether using the WC measure will give us more true responses than WW.

For the purpose of this experiment, we use the Skipgram model with *dim*=300 and *win*=10 as these settings produced the best overall performance in the quantitative experiment on ProNorm data. The simulation procedure is as follows: For each cue word w_1 in the ProNorm dataset, each model generates the n most similar words in the vocabulary and we count how many of the human responses were contained in each set. The first model looks up nearest neighbors of w_1 within the word space (thus using WW as the proximity measure) and the second model searches for the nearest neighbors of w_1 within the Context space (thus using WC as the proximity measure). Variable n indicates the total number of guesses a model is allowed to make when responding to a given cue word. In other words, n is the size of the subspace explored around the cue word within each distributional semantic space. Since our previous experiment showed a higher correlation between WC and the relatedness norms, we expect that neighboring words within the context embedding space (in the vicinity of the cues word embedding) should be more populated with related words (i.e., human responses) compared to neighboring words within the word embedding space. Regarding the above procedure, we first extract the word embedding of the cue w_1 and then consider all human responses for that cue, i.e. w_2 of all existing pairs (w_1, w_2) in the dataset, within both the word and context embedding spaces. If, as results of the previous experiment suggest, WC is a better measure of forward relatedness, then a larger portion of human responses should be found in neighboring words within the context space than within the word space surrounding the cue word.

6.1 Results

Our distributional spaces are constructed based on Wikipedia text; therefore, the model vocabulary is very large and noisy. While the top-rank guesses of the model (both measures) are indeed similar/related to the cue words, a lot of them are more frequent in the training corpus genre, i.e. Wikipedia language, than in the simpler language humans (e.g., subjects of the ProNorm study) use when recalling direct relations. For example, in response to the cue word *restaurant* subjects of the ProNorm study generated words such as *plate*, *food*, *menu*, *drink*, and *chef*. In addition to correct guesses, both WW and WC models trained on web corpora generated words such as *bistro*, *eatery*, *hotel*, *grill* and *buffet* as closest words to *restaurant*. Another example would be the cue word *house*, which in the ProNorm experiment triggered *door*, *family*, *bricks*, *bed*, *window*, *roof*, *furniture*, *fireplace*, *chimney*, and *kitchen*. The WW model generated the following words as top candidates, which are in fact taxonomically sim-

ilar, to *house*: *mansion*, *farmhouse*, and *cottage*. WC model generates relatively more thematically related words, some of which are correct guesses (overlapping with human data) and some are not: *barn*, *residence*, *estate*, *dining*, *room*, *stables*, *fireplace*, *family*, and *kitchen*. On average, only one human response per cue can be found in the top 30 model responses. Blue and candy bars in Figure 6 show the total number of correct guesses by the model using WW and WC measures, respectively. This quantity is the total correct guesses for all 100 cues in the ProNorm dataset (x-axis), when the n most similar neighboring words are examined in each space (y-axis). We explored n values between 10 and 100.

Table 4 shows an example of our simulation for the word *car*. As the search space widens up to 100 most similar words in the vicinity of the cue word embedding, more overlap is observed between human responses and model responses. In addition to synonymous words such as *automobile*, the majority of incorrect guesses for the cue word *car* are names of automobile models such as *suv* and *bmw*. The W space around the cue word embedding is more populated with such taxonomically similar words compared to the C space around the cue word. On the other hand, as the results suggest, thematically related words such as *driver* and *steering wheel* can more easily be found within the surrounding C space. This pattern is very consistent across all the cue words in the ProNorm dataset, suggesting that WC is a more valid measure of forward thematic relatedness. This qualitative observation suggests that the differences between the Spearman correlations in Table 2 were meaningful, and vector-based measures of similarity and relatedness, i.e., WW and WC, return different sets of neighboring words to a given cue word.

n		Correct guesses by each measure
20	WW	tires
	WC	tires|driver
50	WW	tires
	WC	tires|driver|driving
100	WW	tires
	WC	tires|driver|driving|steering wheel

Table 4: Human responses for the cue word *car* found in top-n neighboring words within the word and context embedding spaces using WW and WC measures.

7 Related Work

Word embeddings learned from unlabeled text using different models such as Word2Vec and Glove are currently being used for representation of input to deep neural networks that carry out a variety of NLP tasks. Word similarity/relatedness datasets have been the basis for intrinsic evaluation of word embeddings. These datasets provide researchers with insights about how word relations are demonstrated in a distributional space. Previous work has employed WordSim353, SimLex999 and several other established similarity/relatedness datasets for evaluation of word embeddings (Baroni et al., 2014; Kiela et al., 2015; Levy et al., 2015; Melamud et al., 2016; Sahlgren and Lenci, 2016).

A closely related previous study to the current study is the comprehensive evaluation of Word2Vec and three other distributional semantic models by Levy et al. (2015), where they demonstrated that all the models could learn word relations to similar extent if hyper-parameters were carefully tuned. In particular, Levy et al. discussed the effect of averaging word and context vectors on capturing first and second-order similarity. However, the w+c option did not make it to their result tables because it was not selected as one of the generally optimal settings, while mentioned to be useful to test.

Asr and Jones (2017) looked more closely into this optional parameter setting in their study of count-based vs. predictive distributional semantic models (Word2Vec Skipgram vs. PPMI SVD). Using an artificial language framework, they showed that considering the w+c option would extend the range of word-to-word cosine similarity scores, and directly affect the topology of word clusters in the distributional space. However, none of the mentioned works studied the individual terms in the cosine similarity obtained from Word2Vec Skipgram when the w+c option is used, thus they left the question of using these terms for replicating psycholinguistic data on asymmetric association open. Another related line of research in NLP is work on retrofitting of word embeddings using additional lexical resources to reflect specific relations between words more strongly (Faruqui et al., 2015; Kiela et al., 2015). Kiela et al. (2015) looked into the particular case of similarity and relatedness. They pro-

posed using Thesaurus synonymy data and free-association data in training of the word embeddings to obtain vectors suitable for similarity and relatedness, respectively. In contrast to this category of work though, the objective of our research is elaborating the functionality of the word embedding algorithms and how their general-purpose output should be interpreted and queried rather than trying to maximize the performance of the model on a given task by modifying training data or the training mechanism.

Our study adds to the existing body of research by employing word relatedness data collected within a standard psychology experiment and showing how first- vs. second-order information accumulated on the two layers of the popular Skipgram model can be used for different tasks. We showed that the distributional measure for capturing asymmetric relatedness between two words is different from a measure that captures taxonomic similarity even though both types of information are obtained from a unified model trained on a single source of co-occurrence data.

8 Conclusions

Word and context embeddings produced by Word2Vec Skipgram are two different semantic representations of the vocabulary words within the same Euclidean space. We proposed several measures for complementary similarity and relatedness judgments computed based on these embeddings. Asymmetric measures obtained from the inner product of a vector from the word embedding space and a vector from the context embedding space are representative of first-order thematic relations between words.

We examined our proposal using a recently published dataset of production norms (Jouravlev and McRae, 2016) and confirmed when people were explicitly asked to recall thematically related words, their responses were more likely located within the context embedding space in the vicinity of the cues word embedding. In other words, WC, where W is the word embedding of the cue and C is the context embedding of the response, best measures forward thematic relatedness.

We also ran experiments on pure similarity judgment by employing a commonly used dataset of word pairs scored according to taxonomic similarity rather than other types of relations (Hill et al., 2015). Human judgments on word similarity taken from this data were best predicted by a symmetric measure, the classic WW cosine similarity between the word vectors. This suggests that the best measures of taxonomic similarity and thematic relatedness are different in distributional space, even though information involved in both measurements is collected from the same set of co-occurrence features.

Based on the observations made in the paper, we can also argue that the free recall task in the constraint manner where people are asked to name related words (such as in Jouravlev and McRae's study) is similar to the task of predicting context words for the given cue word. This is an important finding for the psycholinguitic research trying to study the mechanisms in lexical production tasks. For NLP research, these findings motivate taking different approaches in problems where thematic relations between words is important for the task, e.g., in assessment of text coherence, question answering, or language generation.

Finally, our experiments elaborated the functionality of the two transformation matrices in Word2Vec architecture. We repeated some of our experiments with GloVe, another popular word embedding model with two final sets of (word/context) vectors. We found similar patterns of relative goodness of measures: WW was consistently better in scoring similarity between two words and WC was better in measuring the thematic relatedness. However, the asymmetry between WC and CW did not come out clearly in these experiments and the overall performance of the GloVe model in the similarity task was much lower than Skipgram. A closer investigation of the GloVe model architecture will be necessary for argumentation about its different results (B Appendix includes results of our preliminary experiments with GloVe). Other vector space models obtained from non-neural architectures can also be examined in this framework. For example, Levy et al. (2015) showed that the w+c option (using the average of word and context embeddings as word vectors) could be simulated in a count-based model that applies SVD to the PMI matrix of word-context co-occurrences. Examining these models on similarity vs. relatedness using our proposed measures will be left for the future.[5]

[5]Code for running all experiments using Word2Vec and GloVe models is available at https://github.com/FTAsr/wordvet

Acknowledgments

We are thankful to our reviewers for their helpful feedback on the initial version of the paper and suggestions for extension of the work. This research was funded by grant R305A140382 from the Institute of Education Sciences, USA.

A Appendix

Supplementary results on SimLex and ProNorm datasets using Skipgram with *dim*=100 & 200 are presented here. Patterns of how WW and CW measures predict similarity and relatedness are consistently repeated in these parameter settings.

Measure	100-3	100-6	100-10
WW	**0.36**	**0.36**	**0.34**
WC	0.31	0.32	0.33
Measure	200-3	200-6	200-10
WW	**0.42**	**0.40**	**0.39**
WC	0.34	0.35	0.35

Table 5: Spearman correlation between similarity scores (SimLex) and Skipgram measures.

Measure	100-3	100-6	100-10
WW	0.18	0.19	0.19
WC	**0.19**	**0.21**	**0.21**
Measure	200-3	200-6	200-10
WW	0.19	0.20	0.21
WC	**0.22**	**0.24**	**0.24**

Table 6: Spearman correlation between similarity scores (ProNorm) and Skipgram measures.

B Supplementary Results using GloVe

Supplementary result on SimLex and ProNorm datasets using GloVe models with *dim*=300 and *win*=3/6/10 are presented in this section. GloVe had a general disadvantage in learning word similarity (SimLex) compared to Skipgram. Patterns of how WW and CW measures predict similarity and relatedness are nevertheless similar across models: WC is much better than WW for relatedness prediction.

Measure	300-3	300-6	300-10
WW	0.25	0.26	0.16
CC	**0.26**	0.25	0.18
WC	0.13	0.17	0.16
CW	0.15	0.17	0.14
AA	**0.26**	**0.27**	**0.20**
AllReg	**0.29**	**0.30**	**0.25**

Table 7: Spearman correlation between similarity scores (SimLex) and GloVe measures.

Measure	300-3	300-6	300-10
WW	0.15	0.16	0.14
CC	0.13	0.17	0.14
WC	**0.22**	**0.21**	0.21
CW	0.19	**0.21**	**0.22**
AA	0.20	**0.21**	0.19
AllReg	**0.22**	**0.21**	**0.24**

Table 8: Spearman correlation between relatedness scores (ProNorm) and GloVe measures.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27.

Tatsuya Aoki, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2017. Distinguishing japanese non-standard usages from standard ones. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2323–2328.

Fatemeh Torabi Asr and Michael Jones. 2017. An artificial language evaluation of distributional semantic models. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 134–142.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 238–247.

Christine Chiarello, Curt Burgess, Lorie Richards, and Alma Pollock. 1990. Semantic and associative priming in the cerebral hemispheres: Some words do, some words don't sometimes, some places. *Brain and language*, 38(1):75–104.

Simon De Deyne, Amy Perfors, and Daniel J Navarro. 2016. Predicting human similarity judgments with distributional models: The value of word associations. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1861–1870.

Simon De Deyne and Gert Storms. 2008. Word associations: Network and semantic properties. *Behavior Research Methods*, 40(1):213–231.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1606–1615.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.

Thomas L Griffiths, Mark Steyvers, and Joshua B Tenenbaum. 2007. Topics in semantic representation. *Psychological review*, 114(2):211.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Keith A Hutchison. 2003. Is semantic priming due to association strength or feature overlap? a microanalytic review. *Psychonomic Bulletin & Review*, 10(4):785–813.

Michael N Jones, Melody Dye, and Brendan T Johns. 2017. Context as an organizing principle of the lexicon. In *Psychology of Learning and Motivation*, volume 67, pages 239–283. Elsevier.

Olessia Jouravlev and Ken McRae. 2016. Thematic relatedness production norms for 100 object concepts. *Behavior research methods*, 48(4):1349–1357.

Douwe Kiela, Felix Hill, and Stephen Clark. 2015. Specializing word embeddings for similarity or relatedness. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2044–2048.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Margery Lucas. 2000. Semantic priming without association: A meta-analytic review. *Psychonomic Bulletin & Review*, 7(4):618–630.

Paweł Mandera, Emmanuel Keuleers, and Marc Brysbaert. 2017. Explaining human performance in psycholinguistic tasks with models of semantic similarity based on prediction and counting: A review and empirical validation. *Journal of Memory and Language*, 92:57–78.

Oren Melamud, Ido Dagan, and Jacob Goldberger. 2015. Modeling word meaning in context with substitute vectors. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 472–482.

Oren Melamud, David McClosky, Siddharth Patwardhan, and Mohit Bansal. 2016. The role of context types and dimensionality in learning word embeddings. *arXiv preprint arXiv:1601.00893*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Douglas L. Nelson, Cathy L. McEvoy, and Thomas A. Schreiber. 1999. The university of south florida word association, rhyme, and fragment norms. retrieved february 5, 2004.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Magnus Sahlgren and Alessandro Lenci. 2016. The effects of data size and frequency range on distributional semantic models. *arXiv preprint arXiv:1609.08293*.

Amos Tversky and Itamar Gati. 1982. Similarity, separability, and the triangle inequality. *Psychological review*, 89(2):123.

Semantic Structural Evaluation for Text Simplification

Elior Sulem, Omri Abend, Ari Rappoport
Department of Computer Science, The Hebrew University of Jerusalem
{eliors|oabend|arir}@cs.huji.ac.il

Abstract

Current measures for evaluating text simplification systems focus on evaluating lexical text aspects, neglecting its structural aspects. In this paper we propose the first measure to address structural aspects of text simplification, called SAMSA. It leverages recent advances in semantic parsing to assess simplification quality by decomposing the input based on its semantic structure and comparing it to the output. SAMSA provides a reference-less automatic evaluation procedure, avoiding the problems that reference-based methods face due to the vast space of valid simplifications for a given sentence. Our human evaluation experiments show both SAMSA's substantial correlation with human judgments, as well as the deficiency of existing reference-based measures in evaluating structural simplification.[1]

1 Introduction

Text simplification (*TS*) addresses the translation of an input sentence into one or more simpler sentences. It is a useful preprocessing step for several NLP tasks, such as machine translation (Chandrasekar et al., 1996; Mishra et al., 2014) and relation extraction (Niklaus et al., 2016), and has also been shown useful in the development of reading aids, e.g., for people with dyslexia (Rello et al., 2013) or non-native speakers (Siddharthan, 2002).

The task has attracted much attention in the past decade (Zhu et al., 2010; Woodsend and Lapata, 2011; Wubben et al., 2012; Siddharthan and Angrosh, 2014; Narayan and Gardent, 2014), but has yet to converge on an evaluation protocol that yields comparable results across different methods and strongly correlates with human judgments. This is in part due to the difficulty to combine the effects of different simplification operations (e.g., deletion, splitting and substitution). Xu et al. (2016) has recently made considerable progress towards that goal, and proposed to tackle it both by using an improved reference-based measure, named SARI, and by increasing the number of references. However, their research focused on lexical, rather than structural simplification, which provides a complementary view of TS quality as this paper will show.

This paper focuses on the evaluation of the structural aspects of the task. We introduce the semantic measure SAMSA (Simplification Automatic evaluation Measure through Semantic Annotation), the first structure-aware measure for TS in general, and the first to use semantic structure in this context in particular. SAMSA stipulates that an optimal split of the input is one where each predicate-argument structure is assigned its own sentence, and measures to what extent this assertion holds for the input-output pair in question, by using semantic structure. SAMSA focuses on the core semantic components of the sentence, and is tolerant towards the deletion of other units.[2]

For example, SAMSA will assign a high score to the output split "John got home. John gave Mary a call." for the input sentence "John got home and gave Mary a call.", as it splits each of its predicate-argument structures to a different sentence. Splits that alter predicate-argument relations such as "John got home and gave. Mary called." are penalized by SAMSA.

SAMSA's use of semantic structures for TS evaluation has several motivations. First, it provides means to measure the extent to which the meaning of the source is preserved in the output. Second, it provides means for measuring whether the input sentence was split to semantic units of

[1] All data and code are available in https://github.com/eliorsulem/SAMSA.

[2] We do not consider other structural operations, such as passive to active transformations (Canning, 2002), that are currently not treated by corpus-based simplification systems.

Proceedings of NAACL-HLT 2018, pages 685–696
New Orleans, Louisiana, June 1 - 6, 2018.

the right granularity. Third, defining a semantic measure that does not require references avoids the difficulties incurred by their non-uniqueness, and the difficulty in collecting high quality references, as reported by Xu et al. (2015) and by Narayan and Gardent (2014) with respect to the Parallel Wikipedia Corpus (PWKP; Zhu et al., 2010). SAMSA is further motivated by its use of semantic annotation only on the source side, which allows to evaluate multiple systems using same source-side annotation, and avoids the need to parse system outputs, which can be garbled.

In this paper we use the UCCA scheme for defining semantic structure (Abend and Rappoport, 2013). UCCA has been shown to be preserved remarkably well across translations (Sulem et al., 2015) and has also been successfully used for machine translation evaluation (Birch et al., 2016) (Section 2). We note, however, that SAMSA can be adapted to work with any semantic scheme that captures predicate-argument relations, such as AMR (Banarescu et al., 2013) or Discourse Representation Structures (Kamp, 1981), as used by Narayan and Gardent (2014).

We experiment with SAMSA both where semantic annotation is carried out manually, and where it is carried out by a parser. See Section 4. We conduct human rating experiments and compare the resulting system rankings with those predicted by SAMSA. We find that SAMSA's rankings obtain high correlations with human rankings, and compare favorably to existing reference-based measures for TS. Moreover, our results show that existing measures, which mainly target lexical simplification, are ill-suited to predict human judgments where structural simplification is involved. Finally, we apply SAMSA to the dataset of the QATS shared task on simplification evaluation (Štajner et al., 2016). We find that SAMSA obtains comparative correlation with human judgments on the task, despite operating in a more restricted setting, as it does not use human ratings as training data and focuses only on structural aspects of simplicity. Section 2 presents previous work. Section 3 discusses UCCA. Section 4 presents SAMSA. Section 5 details the collection of human judgments. Our experimental setup for comparing our human and automatic rankings is given in Section 6, and results are given in Section 7, showing superior results for SAMSA. A discussion on the results is presented in Section 8. Section 9 presents experiments with SAMSA on the QATS evaluation benchmark.

2 Related Work

Evaluation Metrics for Text Simplification. As pointed out by Xu et al. (2016), many of the existing measures for TS evaluation do not generalize across systems, because they fail to capture the combined effects of the different simplification operations. The two main directions pursued are direct human judgments and automatic measures borrowed from machine translation (MT) evaluation. Human judgments generally include grammaticality (or fluency), meaning preservation (or adequacy) and simplicity. Human evaluation is usually carried out with a small number of sentences (18 to 20), randomly selected from the test set (Wubben et al., 2012; Narayan and Gardent, 2014, 2016).

The most commonly used automatic measure for TS is BLEU (Papineni et al., 2002). Using 20 source sentences from the PWKP test corpus with 5 simplified sentences for each of them, Wubben et al. (2012) investigated the correlation of BLEU with human evaluation, reporting positive correlation for simplicity, but no correlation for adequacy. Štajner et al. (2014) explored the correlation with human judgments of six automatic metrics: cosine similarity with a bag-of-words representation, METEOR (Denkowski and Lavie, 2011), TERp (Snover et al., 2009), TINE (Rios et al., 2011) and two sub-components of TINE: T-BLEU (a variant of BLEU which uses lower n-grams when no 4-grams are found) and SRL (based on semantic role labeling). Using 280 pairs of a source sentence and a simplified output with only structural modifications, they found positive correlations for all the metrics except TERp with respect to meaning preservation and positive albeit lower correlations for METEOR, T-BLEU and TINE with respect to grammaticality. Human simplicity judgments were not considered in this experiment. In this paper we collect human judgments for grammaticality, meaning preservation and *structural* simplicity. To our knowledge, this is the first work to target structural simplicity evaluation, and it does so both through elicitation of human judgments and through the definition of SAMSA.

Xu et al. (2016) were the first to propose two evaluation measures tailored for simplification, focusing on lexical simplification. The first metric is FKBLEU, a combination of iBLEU (Sun

and Zhou, 2012), originally proposed for evaluating paraphrase generation by comparing the output both to the reference and to the input, and of the Flesch-Kincaid Index (FK), a measure of the readability of the text (Kincaid et al., 1975). The second one is SARI (System output Against References and against the Input sentence) which compares the n-grams of the system output with those of the input and the human references, separately evaluating the quality of words that are added, deleted and kept by the systems. They found that FKBLEU and even more so SARI correlate better with human simplicity judgments than BLEU. On the other hand, BLEU (with multiple references) outperforms the other metrics on the dimensions of grammaticality and meaning preservation.

As the Parallel Wikipedia Corpus (PWKP), usually used in simplification research, has been shown to contain a large portion of problematic simplifications (Xu et al., 2015; Hwang et al., 2015), Xu et al. (2016) further proposed to use multiple references (instead of a single reference) in the evaluation measures. SAMSA addresses this issue by directly comparing the input and the output of the simplification system, without requiring manually curated references.

Structural Measures for Text-to-text Generation. Other than measuring the number of splits (Narayan and Gardent, 2014, 2016), which only assesses the frequency of this operation and not its quality, no structural measures were previously proposed for the evaluation of structural simplification. The need for such a measure is pressing, given recent interest in structural simplification, e.g., in the Split and Rephrase task (Narayan et al., 2017), which focuses on sentence splitting.

In the task of sentence compression, which is similar to simplification in that they both involve deletion and paraphrasing, Clarke and Lapata (2006) showed that a metric that uses syntactic dependencies better correlates with human evaluation than a metric based on surface sub-strings. Toutanova et al. (2016) found that structure-aware metrics obtain higher correlation with human evaluation over bigram-based metrics, in particular with grammaticality judgments, but that they do not significantly outperform bigram-based metrics on any parameter. Both Clarke and Lapata (2006) and Toutanova et al. (2016) use reference-based metrics that use syntactic structure on both the output and the references. SAMSA on the other hand uses linguistic annotation only on the source side, with semantic structures instead of syntactic ones.

Semantic structures were used in MT evaluation, for example in the MEANT metric (Lo et al., 2012), which compares the output and the reference sentences, both annotated using SRL (Semantic Role Labeling). Lo et al. (2014) proposes the XMEANT variant, which compares the SRL structures of the source and output (without using references). As some frequent constructions like nominal argument structures are not addressed by the SRL annotation, Birch et al. (2016) proposed HUME, a human evaluation metric based on UCCA, using the semantic annotation only on the source side when comparing it to the output. We differ from HUME in proposing an automatic metric, tackling monolingual text simplification, rather than MT.

The UCCA annotation has also been recently used for the evaluation of Grammatical Error Correction (GEC). The USIM metric (Choshen and Abend, 2018) measures the semantic faithfulness of the output to the source by comparing their respective UCCA graphs.

Semantic Structures in Text Simplification. In most of the work investigating the structural operations involved in text simplification, both in rule-based systems (Siddharthan and Angrosh, 2014) and in statistical systems (Zhu et al., 2010; Woodsend and Lapata, 2011), the structures that were considered were syntactic. Narayan and Gardent (2014, 2016) proposed to use semantic structures in the simplification model, in particular in order to avoid splits and deletions which are inconsistent with the semantic structures. SAMSA identifies such incoherent splits, e.g., a split of a phrase describing a single event, and penalizes them.

Glavas and Štajner (2013) presented two simplification systems based on event extraction. One of them, named Event-wise Simplification, transforms each factual event motion into a separate sentence. This approach fits with SAMSA's stipulation, that an optimal structural simplification is one where each (UCCA-) event in the input sentence is assigned a separate output sentence. However, unlike in their model, SAMSA stipulates that not only should multiple events evoked by a verb in the same sentence be avoided in a simplification, but penalizes sentences containing multiple events evoked by a lexical item of any category. For example, the sentence "John's un-

expected kick towards the gate saved the game" which has two events, one evoked by "kick" (a noun) and another by "saving" (a verb) can be converted to "John kicked the ball towards the gate. It saved the game."

3 UCCA's Semantic Structures

In this section we will briefly describe the UCCA scheme, focusing on the concepts of Scenes and Centers which are key in the definition of SAMSA. UCCA (Universal Cognitive Conceptual Annotation; Abend and Rappoport, 2013) is a semantic annotation scheme based on typological (Dixon, 2010b,a, 2012) and cognitive (Langacker, 2008) theories which aims to represent the main semantic phenomena in the text, abstracting away from syntactic detail. UCCA structures are directed acyclic graphs whose nodes (or units) correspond either to the leaves of the graph (including the words of the text) or to several elements jointly viewed as a single entity according to some semantic or cognitive consideration. Unlike AMR, UCCA semantic units are directly anchored in the text (Abend and Rappoport, 2017; Birch et al., 2016), which allows easy inclusion of a word-to-word alignment in the metric model (Section 4).

UCCA Scenes. A Scene, which is the most basic notion of the foundational layer of UCCA considered here, describes a movement, an action or a state which persists in time. Every Scene contains one main relation, which can be either a Process or a State. The Scene may contain one or more Participants, which are interpreted in a broad sense, including locations and destinations. For example, the sentence "He ran into the park" has a single Scene whose Process is "ran". The two Participants are "He" and "into the park".

Scenes can have several roles in the text. First, they can provide additional information about an established entity (Elaborator Scenes) as for example the Scene "who entered the house" in the sentence "The man who entered the house is John". They can also be one of the Participants of another Scene, for example, "he will be late" in the sentence: "He said he will be late". In the other cases, the Scenes are annotated as parallel Scenes (H) which can be linked by a Linker (L): "When$_L$ [he will arrive at home]$_H$, [he will call them]$_H$".

Unit Centers. With regard to units which are not Scenes, the category Center denotes the semantic head of the unit. For example, "dogs" is the center of the expression "big brown dogs" and "box" is the center of "in the box". There could be more than one Center in a non-Scene unit, for example in the case of coordination, where all conjuncts are Centers.

4 The SAMSA Metric

SAMSA's main premise is that a structurally correct simplification is one where: (1) each sentence contains a single event from the input (UCCA Scene), (2) the main relation of each of the events and their participants are retained in the output.

For example, consider "John wrote a book. I read that book." as a simplification of "I read the book that John wrote.". Each output sentence contains one Scene, which has the same Scene elements as the source, and would thus be deemed correct by SAMSA. On the other hand, the output "John wrote. I read the book." is an incorrect split of that sentence, since a participant of the "writing" Scene, namely "the book" is absent in the split sentence. SAMSA would indeed penalize such a case.

Similarly, Scenes which have elements across several sentences receive a zero score by SAMSA. As an example, consider the sentence "The combination of new weapons and tactics marks this battle as the end of chivalry", and erroneous split "The combination of new weapons and tactics. It is the end of chivalry." (adapted from the output of a recent system on the PWKP corpus), which does not preserve the original meaning.

4.1 Matching Scenes to Sentences

SAMSA is based on two external linguistic resources. One is a semantic annotation (UCCA in our experiments) of the source side, which can be carried out either manually or automatically, using the TUPA parser[3] (Transition-based UCCA parser; Hershcovich et al., 2017) for UCCA. UCCA decomposes each sentence s into a set of Scenes $\{sc_1, sc_2, .., sc_n\}$, where each scene sc_i contains a main relation mr_i (sub-span of sc_i) and a set of zero or more participants A_i.

The second resource is a word-to-word alignment A between the words in the input and one or zero words in the output. The monolingual alignment thus permits SAMSA not to penalize outputs that involve lexical substitutions (e.g., "com-

[3] https://github.com/danielhers/tupa

mence" might be aligned with "start"). We denote by n_{inp} the number of UCCA Scenes in the input sentence and by n_{out} the number of sentences in the output.

Given an input sentence's UCCA Scenes $sc_1, \ldots, sc_{n_{inp}}$, a non-annotated output of a simplification system split into sentences $s_1, \ldots, s_{n_{out}}$, and their word alignment A, we distinguish between two cases:

1. $n_{inp} \geq n_{out}$: in this case, we compute the maximal Many-to-1 correspondence between Scenes and sentences. A Scene is matched to a sentence in the following way. We say that a leaf l in a Scene sc is *consistent* in a Scene-sentence mapping M which maps sc to a sentence s, if there is a word $w \in s$ which l aligns to (according to the word alignment A). The score of matching a Scene sc to a sentence s is then defined to be the total number of consistent leaves in sc. We traverse the Scenes in their order of occurrence in the text, selecting for each the sentence that maximizes the score. If $n_{inp} = n_{out}$, once a sentence is matched to a Scene, it cannot be matched to another one. Ties between sentences are broken towards the sentence that appeared first in the output.

$$\mathrm{M}^*(sc_i) = \mathrm{argmax}_s score(sc_i, s)$$
$$\text{s.t. } s \notin \{M^*(sc_1), \ldots, M^*(sc_{i-1})\} \text{ if } n_{inp} = n_{out}$$

2. $n_{inp} < n_{out}$: In this case, a Scene will necessarily be split across several sentences. As this is an undesired result, we assign this instance a score of zero.

4.2 Score Computation

Minimal Centers. The minimal center of a UCCA unit u is UCCA's notion of a semantic head word, defined through recursive rules not unlike the head propagation rules used for converting constituency structures to dependency structures. More formally, we define the minimal center of a UCCA unit u (here a Participant or a Main Relation) to be the UCCA graph's leaf reached by starting from u and iteratively selecting the child tagged as Center. If a Participant (or a Center inside a Participant) is a Scene, its center is the main relation (Process or State) of the Scene.

For example, the center of the unit "The previous president of the commission" (u_1) is "president of the commission". The center of the latter is "president", which is a leaf in the graph. So the minimal center of u_1 is "president".

Given the input sentence Scenes $\{sc_1, ..., sc_{n_{inp}}\}$, the output sentences $\{s_1, ..., s_{n_{out}}\}$, and a mapping between them M^*, SAMSA is defined as:

$$\frac{n_{out}}{n_{inp}} \frac{1}{2n_{inp}} \sum_{sc_i} [\mathbb{1}_{M^*(sc_i)}(MR_i) + \frac{1}{k_i} \sum_{j=1}^{k_i} \mathbb{1}_{M^*(sc_i)}(\mathrm{Par}_i^{(j)})]$$

where $\mathrm{MR_i}$ is the minimal center of the main relation (Process or State) of sc_i, and $\mathrm{Par_i^{(j)}}$ ($j = 1, \ldots, k_i$) are the minimal centers of the Participants of sc_i.

For an output sentence s, $\mathbb{1}_s(u)$ is a function from the input units to $\{0, 1\}$, which returns 1 iff u is aligned (according to A) with a word in s.[4]

The role of the non-splitting penalty term n_{out}/n_{inp} in the SAMSA formula is to penalize cases where the number of sentences in the output is smaller than the number of Scenes. In order to isolate the effect of the non-splitting penalty, we experiment with an additional metric SAMSA$_{abl}$ (reads "SAMSA ablated"), which is identical to SAMSA but does not take this term into account. Corpus-level SAMSA and SAMSA$_{abl}$ scores are obtained by averaging their sentence scores.

In the case of implicit units i.e. omitted units that do not appear explicitly in the text (Abend and Rappoport, 2013), since the unit preservation cannot be directly captured, the score t for the relevant unit will be set to 0.5. For example, in the Scene "traveling is fun", the people who are traveling correspond to an implicit Participant. As implicit units are not covered by TUPA, this will only be relevant for the semi-automatic implementation of the metric (see Section 6).

5 Human Evaluation Benchmark

5.1 Evaluation Protocol

For testing the automatic metric, we first build a human evaluation benchmark, using 100 sentences from the complex part of the PWKP corpus and the outputs of six recent simplification systems for these sentences:[5] (1) TSM (Zhu et al., 2010) using Tree-Based SMT, (2) RevILP (Woodsend and Lapata, 2011) using Quasi-Synchronous Grammars, (3) PBMT-R (Wubben et al., 2012) using Phrase-Based SMT, (4) Hybrid (Narayan and Gardent,

[4] In some cases, the unit u can be a sequence of centers (if there are several minimal centers). In these cases, $\mathbb{1}_s(u)$ returns 1 iff the condition holds for all centers.

[5] All the data can be found here: http://homepages.inf.ed.ac.uk/snaraya2/data/simplification-2016.tgz.

2014), a supervised system using DRS, (5) UNSUP (Narayan and Gardent, 2016), an unsupervised system using DRS, and (6) Split-Deletion (Narayan and Gardent, 2016), the unsupervised system with only structural operations.

All these systems explicitly address at least one type of structural simplification operation. The last system, Split-Deletion, performs only structural (i.e., no lexical) operations. It is thus an interesting test case for SAMSA since here the aligner can be replaced by a simple match between identical words. In total we obtain 600 system outputs from the six systems, as well as 100 sentences from the simple Wikipedia side of the corpus, which serve as references. Five in-house annotators with high proficiency in English evaluated the resulting 700 input-output pairs by answering the questions in Table 1.[6]

Qa addresses grammaticality, Qb and Qc capture two complementary aspects of meaning preservation (the addition and the removal of information) and Qd addresses structural simplicity. Possible answers are: 1 ("no"), 2 ("maybe") and 3 ("yes"). Following Glavas and Štajner (2013), we used a 3 point Likert scale, which has recently been shown to be preferable over a 5 point scale through human studies on sentence compression (Toutanova et al., 2016).

Question Qd was accompanied by a negative example [7] showing a case of lexical simplification, where a complex word is replaced by a simple one. A positive example was not included so as not to bias the annotators by revealing the nature of the operations our experiments focus on (i.e., splitting and deletion).

The PWKP test corpus (Zhu et al., 2010) was selected for our experiments over the development and test sets used in (Xu et al., 2016), as the latter's selection process was explicitly biased towards input-output pairs that mainly contain lexical simplifications.

Qa	Is the output grammatical?
Qb	Does the output add information, compared to the input?
Qc	Does the output remove important information, compared to the input?
Qd	Is the output simpler than the input, ignoring the complexity of the words?

Table 1: Questions for the human evaluation

[6]Each input-output pair was rated by all five annotators.
[7]Other questions appeared without any example.

5.2 Human Score Computation

Given the annotator's answers, we consider the following scores. First, the grammaticality score $\mathcal{G}$ is the answer to Qa. By inverting (changing 1 to 3 and 3 to 1) the answer for Qb, we obtain a Non-Addition score indicating to which extent no additional information has been added. Similarly, inverting the answer to Qc yields the Non-Removal score. Averaging these two scores, we obtain the meaning preservation score $\mathcal{P}$. Finally, the structural simplicity score $\mathcal{S}$ is the answer to Qd. Each of these scores is averaged over the five annotators. We further compute an average human score:

$$\text{AvgHuman} = \frac{1}{3}(\mathcal{G} + \mathcal{P} + \mathcal{S})$$

5.3 Inter-annotator Agreement

Inter-annotator agreement rates are computed in two ways. Table 2 presents the absolute agreement and Cohen's quadratic weighted κ (Cohen, 1968). Table 3 presents Spearman's correlation (ρ) between the human ratings of the input-output pairs (top row), and between the resulting system scores (bottom row). In both cases, the agreement between the five annotators is computed as the average agreement over the 10 annotator pairs.

	Qa	Qb	Qc	Qd
Total	0.58 (0.56)	0.74 (0.30)	0.53 (0.45)	0.57 (0.10)
TSM	0.59 (0.47)	0.75 (0.27)	0.50 (0.40)	0.43 (0.08)
RevILP	0.61 (0.59)	0.78 (0.27)	0.60 (0.43)	0.62 (0.11)
PBMT-R	0.47 (0.42)	0.70 (0.20)	0.58 (0.31)	0.76 (0.10)
Hybrid	0.59 (0.46)	0.77 (0.26)	0.52 (0.48)	0.72 (0.15)
UNSUP	0.51 (0.42)	0.59 (0.10)	0.45 (0.17)	0.52 (0.04)
Split-Deletion	0.59 (0.48)	0.93 (0.02)	0.45 (0.29)	0.55 (0.04)
Reference	0.70 (0.40)	0.66 (0.46)	0.52 (0.58)	0.41 (0.12)

Table 2: Inter-annotator absolute agreement (and quadratic weighted κ), averaged over the 10 annotator pairs. Rows correspond to systems, columns to questions. The top "Total" row refers to the concatenation of the outputs of all 6 systems together with the reference sentences.

	Qa	Qb	Qc	Qd	AvgHuman
Sen.	0.63^{*}	0.30^{*}	0.48^{*}	0.11^{**}	0.49^{*}
Sys.	0.92^{**}	0.54 (0.1)	0.64 (0.06)	0.14 (0.4)	0.64 (0.06)

Table 3: Spearman's correlation (and p-values) of the system-level (top row) and sentence-level (bottom row) ratings of the five annotators. $^{*}p < 10^{-5}$, $^{**}p = 0.002$.

6 Experimental Setup

We further compute SAMSA for the 100 sentences of the PWKP test set and the corresponding system outputs. Experiments are conducted in two settings: (1) a semi-automatic setting where

UCCA annotation was carried out manually by a single expert UCCA annotator using the UCCAApp annotation software (Abend et al., 2017), and according to the standard annotation guidelines;[8] (2) an automatic setting where the UCCA annotation was carried out by the TUPA parser (Hershcovich et al., 2017). Sentence segmentation of the outputs was carried out using the NLTK package (Loper and Bird, 2002). For word alignments, we used the aligner of Sultan et al. (2014).[9]

7 Correlation with Human Evaluation

We compare the system rankings obtained by SAMSA and by the four human parameters. We find that the two leading systems according to AvgHuman and SAMSA turn out to be the same: Split-Deletion and RevILP. This is the case both for the semi-automatic and the automatic implementations of the metric. A Spearman ρ correlation between the human and SAMSA scores (comparing their rankings) is presented in Table 4.

We compare SAMSA and SAMSA$_{abl}$ to the reference-based measures SARI[10] (Xu et al., 2016) and BLEU, as well as to the negative Levenshtein distance to the reference (-LD$_{\mathrm{SR}}$). We use the only available reference for this corpus, in accordance with the standard practice. SARI is a reference-based measure, based on n-gram overlap between the source, output and reference, and focuses on lexical (rather than structural) simplification. For completeness, we include the other two measures reported in Narayan and Gardent (2016), which are measures of similarity to the input (i.e., they quantify the tendency of the systems to introduce changes to the input): the negative Levenshtein distances between the output and input compared to the original complex corpus (-LD$_{\mathrm{SC}}$), and the number of sentences split by each of the systems.

The highest correlation with AvgHuman and grammaticality is obtained by semi-automatic SAMSA (0.58 and 0.54), a high correlation especially in comparison to the inter-annotator agreement on AvgHuman (0.64, Table 3). The automatic version obtains high correlation with human judgments in these settings, where for structural simplicity, it scores somewhat higher than the semi-automatic SAMSA. The highest correlation with structural simplicity is obtained by the number of sentences with splitting, where SAMSA (automatic and semi-automatic) is second and third highest, although when restricted to multi-Scene sentences, the correlation for SAMSA (semi-automatic) is higher (0.89, $p = 0.009$ and 0.77, $p = 0.04$).

The highest correlation for meaning preservation is obtained by SAMSA$_{abl}$ which provides further evidence that the retainment of semantic structures is a strong predictor of meaning preservation (Sulem et al., 2015). SAMSA in itself does not correlate with meaning preservation, probably due to its penalization of under-splitting sentences.

Note that the standard reference-based measures for simplification, BLEU and SARI, obtain low and often negative correlation with human ratings. We believe that this is the case because SARI and BLEU admittedly focus on lexical simplification, and are difficult to use to rank systems which also perform structural simplification.

Our results thus suggest that SAMSA provides additional value in predicting the quality of a simplification system and should be reported in tandem with more lexically-oriented measures.

8 Discussion

Human evaluation parameters. The fact that the highest correlations for structural simplicity and meaning preservation are obtained by different metrics (SAMSA and SAMSA$_{abl}$ respectively) highlights the complementarity of these two parameters for evaluating TS quality but also the difficulty of capturing them together. Indeed, a given sentence-level operation could both change the original meaning by adding or removing information (affecting the $\mathcal{P}$ score) and increase simplicity ($\mathcal{S}$). On the other hand, the identity transformation perfectly preserves the meaning of the original sentence without making it simpler.

For examining this phenomenon, we compute Spearman's correlation at the system-level between the simplicity and meaning preservation human scores. We obtain a correlation of -0.77 ($p = 0.04$) between $\mathcal{S}$ and $\mathcal{P}$. The correlation between $\mathcal{S}$ and the two sub-components of $\mathcal{P}$, the Non-Addition and the Non-Removal scores, are -0.43 ($p = 0.2$) and -0.77 ($p = 0.04$) respectively. These negative correlations support our use

[8] http://www.cs.huji.ac.il/~oabend/ucca.html

[9] https://github.com/ma-sultan/monolingual-word-aligner

[10] Data and code for can be found in https://github.com/cocoxu/simplification.

	Reference-less				Reference-based			Δ from Source	
	SAMSA		SAMSA_{abl}		BLEU	SARI	-LD_{SR}	-LD_{SC}	# Split Sents.
	Semi-Auto.	Auto.	Semi-Auto.	Auto.					
$\mathcal{G}$	**0.54** (0.1)	0.37 (0.2)	0.14 (0.4)	0.14 (0.4)	0.09 (0.4)	-0.77 (0.04)	-0.43 (0.2)	-0.09 (0.4)	0.09 (0.4)
$\mathcal{P}$	-0.09 (0.4)	-0.37 (0.2)	**0.54** (0.1)	**0.54** (0.1)	0.37 (0.2)	-0.14 (0.4)	0.03 (0.5)	0.37 (0.2)	-0.49 (0.2)
$\mathcal{S}$	0.54 (0.1)	0.71 (0.06)	-0.71 (0.06)	-0.71 (0.06)	-0.60 (0.1)	-0.43 (0.2)	-0.43 (0.2)	-0.54 (0.1)	**0.83** (0.02)
AvgHuman	**0.58** (0.1)	0.35 (0.1)	0.09 (0.2)	0.09 (0.2)	0.06 (0.5)	-0.81 (0.02)	-0.46 (0.2)	-0.12 (0.4)	0.14 (0.4)

Table 4: Spearman's correlation of system scores i.e. Pearson's correlation of system rankings (and p-values), between evaluation measures (columns) and human judgments (rows). The ranking is between the six simplification systems experimented with. The left block of columns corresponds to the SAMSA and SAMSA_{abl} measures, in their semi-automatic and automatic forms. The middle block of columns corresponds to the reference-based measures SARI and BLEU, as well as -LD_{SR}, which is the negative Levenshtein distances of the system output from the reference. The right block corresponds to measures of conservatism, and reflect how well the tendency of the systems to introduce changes to the input correlates with the human rankings. The block includes -LD_{SC}, the negative Levenshtein distance from the source sentence, and the number of input sentences split by each of the systems. Levenshtein distances are taken as negative in order to capture similarity between the output and source/reference. The measure with the highest correlation in each row is boldfaced.

of an average human score for assessing the overall quality of the simplification.

Distribution at the sentence level. In addition to the system-level analysis presented in Section 7, we also investigate the behavior of SAMSA at the sentence level by examining its joint distribution with the human evaluation scores. Focusing on the AvgHuman score and the automatic implementation of SAMSA and using the same data as in Section 7, we consider a single pair of scores $(\text{AvgHuman}_i, \text{SAMSA}_i)$, $1 \leq i \leq 100$, for each of the 100 source sentences, averaging over the SAMSA and human scores obtained for the 6 simplification systems (See Figure 1).

The joint distribution indicates a positive correlation between SAMSA and AvgHuman. The corresponding Pearson correlation is indeed 0.27 ($p = 0.03$).

9 Evaluation on the QATS Benchmark

In order to provide further validation for SAMSA predictive value for quality of simplification systems, we report SAMSA's correlation with a recently proposed benchmark, used for the QATS (Quality Assessment for Text Simplification) shared task (Štajner et al., 2016).

Setup. The test corpus contains 126 sentences taken from 3 datasets described in Štajner et al. (2016)[11]: (1) EventS: original sentences from the EMM News-Brief[12] and their syntactically simplified versions (with significant content reduction) by the EventSimplify TS system (Glavas and Štajner, 2013)[13] (the test corpus contains 54 pairs from this dataset), (2) EncBrit: original sentences from the Encyclopedia Britannica (Barzilay and Elhadad, 2003) and their automatic simplifications obtained using ATS systems based on several phrase-based statistical MT systems (Štajner et al., 2015) trained on Wikipedia TS corpus (Coster and Kauchak, 2011) (24 pairs), and (3) LSLight: sentences from English Wikipedia and their automatic simplifications (Glavaš and Štajner, 2015) by three different lexical simplification systems (Biran et al., 2011; Horn et al., 2014; Glavaš and Štajner, 2015) (48 pairs).

Human evaluation is also provided by this resource, with scores for overall quality, grammaticality, meaning preservation and simplicity. Importantly, the simplicity score does not explicitly refer to the output's structural simplicity, but rather to its readability. We focus on the overall human score, and compare it to SAMSA. Since different systems were used to simplify different portions of the input, correlation is taken at the sentence level.

We use the same implementations of SAMSA. Manual UCCA annotation is here performed by one of the authors of this paper.

Results. We follow Štajner et al. (2016) and report the Pearson correlations (at the sentence level) between the rankings of the metrics and the human evaluation scores. Results show that the semi-automatic/automatic SAMSA obtains a Pearson correlation of 0.32 and 0.28 with the human scores. This places these measures in the 3rd and 4th places in the shared task, where the only two systems that surpassed it are marginally better, with scores of 0.33 and 0.34, and where the next

[11] http://qats2016.github.io/shared.html

[12] emm.newsbrief.eu/NewsBrief/clusteredition/en/latest.html

[13] takelab.fer.hr/data/symplify

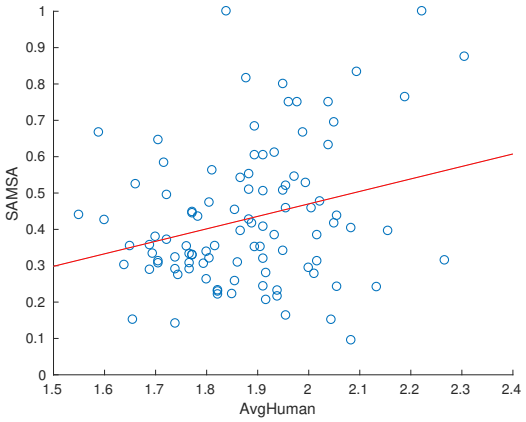


Figure 1: Joint distribution of the automatic SAMSA and the AvgHuman scores at the sentence level. Each point in the graph corresponds to a single source sentence. In addition to the scatter plot, a least-squares regression line is presented.

system in QATS obtained a correlation of 0.23.

This correlation by SAMSA was obtained in more restricted conditions, compared to the measures that competed in QATS. First, SAMSA computes its score by only considering the UCCA structure of the source, and an automatic word-to-word alignment between the source and output. Most QATS systems, including OSVCML and OSVCML2 (Nisioi and Nauze, 2016) which scored highest on the shared task, use an ensemble of classifiers based on bag-of-words, POS tags, sentiment information, negation, readability measures and other resources. Second, the systems participating in the shared task had training data available to them, annotated by the same annotators as the test data. This was used to train classifiers for predicting their score. This gives the QATS measures much predictive strength, but hampers their interpretability. SAMSA on the other hand is conceptually simple and interpretable. Third, the QATS shared task does not focus on structural simplification, but experiments on different types of systems. Indeed, some of the data was annotated by systems that exclusively perform lexical simplification, which is orthogonal to SAMSA's structural focus.

Given these factors, SAMSA's competitive correlation with the participating systems in QATS suggests that structural simplicity, as reflected by the correct splitting of UCCA Scenes, captures a major component in overall simplification quality, underscoring SAMSA's value. These promising results also motivate a future combination of SAMSA with classifier-based metrics.

10 Conclusion

We presented the first structure-aware metric for text simplification, SAMSA, and the first evaluation experiments that directly target the structural simplification component, separately from the lexical component. We argue that the structural and lexical dimensions of simplification are loosely related, and that TS evaluation protocols should assess both. We empirically demonstrate that strong measures that assess lexical simplification quality (notably SARI), fail to correlate with human judgments when structural simplification is performed by the evaluated systems. Our experiments show that SAMSA correlates well with human judgments in such settings, which demonstrates its usefulness for evaluating and tuning statistical simplification systems, and shows that structural evaluation provides a complementary perspective on simplification quality.

Acknowledgments

We would like to thank Zhemin Zhu and Sander Wubben for sharing their data, as well as the annotators for participating in our evaluation and UCCA annotation experiments. We also thank Daniel Hershcovich and the anonymous reviewers for their helpful comments. This work was partially supported by the Intel Collaborative Research Institute for Computational Intelligence (ICRI-CI) and by the Israel Science Foundation (grant No. 929/17), as well as by the HUJI Cyber Security Research Center in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office.

References

Omri Abend and Ari Rappoport. 2013. Universal Conceptual Cognitive Annotation (UCCA). In *Proc. of ACL-13*. pages 228–238. http://aclweb.org/anthology/P/P13/P13-1023.pdf.

Omri Abend and Ari Rappoport. 2017. The state of the art in semantic representation. In *Proc. of ACL'17*. pages 77–89. http://aclweb.org/anthology/P/P17/P17-1008.pdf.

Omri Abend, Shai Yerushalmi, and Ari Rappoport. 2017. UCCAApp: Web-application for syntactic and semantic phrase-based annotation. In *Proc. of ACL'17, System Demonstrations*. pages 109–114. http://www.aclweb.org/anthology/P17-4019.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Rrepresentation for sembanking. *Proc. of Linguistic Annotation Workshop and Interoperability with Discourse* pages 178–186. http://aclweb.org/anthology/W/W13/W13-2322.pdf.

Regina Barzilay and Noemie Elhadad. 2003. Sentence alignment for monolingual comparable corpora. In *Proc. of EMNLP'03*. pages 25–32. http://www.aclweb.org/anthology/W/W03/W03-1004.pdf.

Or Biran, Samuel Brody, and Noémie Elhadad. 2011. Putting it simply: a context-aware approach to lexical simplification. In *Proc. of ACL'11*. pages 465–501. http://aclweb.org/anthology/P/P11/P11-2087.pdf.

Alexandra Birch, Omri Abend, Ondřej Bojar, and Barry Haddow. 2016. HUME: Human UCCA-based evaluation of machine translation. In *Proc. of EMNLP'16*. pages 1264–1274. http://aclweb.org/anthology/D/D16/D16-1134.pdf.

Yvonne Margaret Canning. 2002. *Syntactic simplification of text*. Ph.D. thesis, University of Sunderland, UK.

Raman Chandrasekar, Christine Doran, and Bangalore Srinivas. 1996. Motivations and methods for sentence simplification. In *Proc. of COLING'96*. pages 1041–1044. http://aclweb.org/anthology/C/C96/C96-2183.pdf.

Leshem Choshen and Omri Abend. 2018. Reference-less measure of faithfulness for grammatical error correction. In *Proc. of NAACL'18 (Short papers)*. To appear.

James Clarke and Mirella Lapata. 2006. Models for sentence compression: A comparison across domains, training requirements and evaluation measures. In *Proc. of ACL-COLING'06*. pages 377–384. http://aclweb.org/anthology/P/P06/P06-1048.pdf.

Jacob Cohen. 1968. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin* 70(4):213.

William Coster and David Kauchak. 2011. Simple English Wikipedia: A new text simplification task. In *Proc. of ACL'11*. pages 665–669. http://aclweb.org/anthology/P/P11/P11-2117.pdf.

Michael Denkowski and Alon Lavie. 2011. Meteor 1.3: Automatic metric for reliable optimization and evaluation of machine translation systems. In *Proc. of WMT'11*. pages 85–91. http://aclweb.org/anthology/W/W11/W11-2107.pdf.

Robert M.W. Dixon. 2010a. *Basic Linguistic Theory: Grammatical Topics*, volume 2. Oxford University Press.

Robert M.W. Dixon. 2010b. *Basic Linguistic Theory: Methodology*, volume 1. Oxford University Press.

Robert M.W. Dixon. 2012. *Basic Linguistic Theory: Further Grammatical Topics*, volume 3. Oxford University Press.

Goran Glavas and Sanja Štajner. 2013. Event-centered simplification of news stories. In *Proc. of the Student Research Workshop associated with RANLP 2013*. pages 71–78. http://aclweb.org/anthology/R13-2011.

Goran Glavaš and Sanja Štajner. 2015. Simplifying lexical simplification: Do we need simplified corpora? In *Proc. of ACL'15 (Short papers)*. pages 63–68. http://www.aclweb.org/anthology/P15-2011.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2017. A transition-based directed acyclic graph parser for UCCA. In *Proc. of ACL'17*. pages 1127–1138. http://aclweb.org/anthology/P/P17/P17-1104.pdf.

Colby Horn, Cathryn Manduca, and David Kauchak. 2014. Learning a lexical simplifier using Wikipedia. In *Proc. of ACL'14 (Short papers)*. pages 458–463. http://aclweb.org/anthology/P/P14/P14-2075.pdf.

William Hwang, Hannaneh Hajishirzi, Mari Ostendorf, and Wei Wu. 2015. Aligning sentences from Standard Wikipedia to Simple Wikipedia. In *Proc. of NAACL'15*. pages 211–217. http://aclweb.org/anthology/N/N15/N15-1022.pdf.

Hans Kamp. 1981. A theory of truth and semantic representation. In J.A.G. Groenendijk, T.M.V. Jassen, B.J. Stokhof, and M.J.B/ Stokhof, editors, *Formal methods in the study of language*. Mathematisch Centrum. Number pt.1 in Mathematical Centre tracts.

J. Peter Kincaid, Robert P. Fishburne Jr., Richard L. Rogers, and Brad S. Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and Flesch reading ease formula) for Navy enlisted personnel. Technical report, DTIC Document.

Ronald W. Langacker. 2008. *Cognitive Grammar: A Basic Introduction*. Oxford University Press, USA.

Chi-kiu Lo, Meriem Beloucif, Markus Saers, and Dekai Wu. 2014. XMEANT: Better semantic mt evaluation without reference translations. In *Proc. of ACL'14 (Short Papers)*. pages 765–771. http://aclweb.org/anthology/P/P14/P14-2124.pdf.

Chi-kiu Lo, Anand Karthik Tumuluru, and Dekai Wu. 2012. Fully automatic semantic MT evaluation. In *Proc. of WMT'12*. pages 243–252. http://aclweb.org/anthology/W/W12/W12-3129.pdf.

Edward Loper and Steven Bird. 2002. NLTK: the natural language toolkit. In *Proc. of EMNLP'02*. pages 63–70. http://www.aclweb.org/anthology/W/W02/W02-0109.pdf.

Kshitij Mishra, Ankush Soni, Rahul Sharma, and Dipti Misra Sharma. 2014. Exploring the effects of sentence simplification on Hindi to English Machine Translation systems. In *Proc. of the Workshop on Automatic Text Simplification: Methods and Applications in the Multilingual Society*. pages 21–29. http://www.aclweb.org/anthology/W14-5603.

Shashi Narayan and Claire Gardent. 2014. Hybrid simplification using deep semantics and machine translation. In *Proc. of ACL14*. pages 435–445. http://aclweb.org/anthology/P/P14/P14-1041.pdf.

Shashi Narayan and Claire Gardent. 2016. Unsupervised sentence simplification using deep semantics. In *Proc. of INLG'16*. pages 111–120. http://aclweb.org/anthology/W/W16/W16-6620.pdf.

Shashi Narayan, Claire Gardent, Shay B. Cohen, and Anastasia Shimorina. 2017. Split and rephrase. In *Proc. of EMNLP'17*. pages 617–627. http://aclweb.org/anthology/D/D17/D17-1065.pdf.

Christina Niklaus, Bernahard Bermeitinger, Siegfried Handschuh, and André Freitas. 2016. A sentence simplification system for improving relation extraction. In *Proc. of COLING'16*. http://aclweb.org/anthology/C/C16/C16-2036.pdf.

Sergiu Nisioi and Fabrice Nauze. 2016. An ensemble method for quality assessment of text simplification. In *Workshop on Quality Assessment for Text Simplification (QATS)*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. of ACL'02*. pages 311–318. http://aclweb.org/anthology/P/P02/P02-1040.pdf.

Luz Rello, Ricardo Baeza-Yates, Stefan Bott, and Horacio Saggion. 2013. Simplify or help?: text simplification strategies for people with dyslexia. In *Proc. of the 10th International Cross-Disciplinary Conference on Web Accesibility*. pages 15:1 – 15:10.

Miguel Rios, Wilker Aziz, and Lucia Specia. 2011. TINE: A metric to assess MT adequacy. In *Proc. of WMT'11*. pages 116–122. http://aclweb.org/anthology/W/W11/W11-2112.pdf.

Advaith Siddharthan. 2002. An architecture for a text simplification system. In *Proc. of LEC*. pages 64–71.

Advaith Siddharthan and M. A. Angrosh. 2014. Hybrid text simplification using synchronous dependency grammars with hand-written and automatically harvested rules. In *Proc. of EACL'14*. pages 722–731. http://aclweb.org/anthology/E/E14/E14-1076.pdf.

Matthew Snover, Nitin Madnani, Bonnie J. Dorr, and Richard Schwartz. 2009. Fluency, adequacy, or HTER? Exploring different human judgments with a tunable MT metric. In *Proc. of WMT'09*. pages 85–91. http://www.aclweb.org/anthology/W09-0441.

Elior Sulem, Omri Abend, and Ari Rappoport. 2015. Conceptual annotations preserve structure across translations. In *Proc. of 1st Workshop on Semantics-Driven Statistical Machine Translation (S2Mt 2015)*. pages 11–22. http://aclweb.org/anthology/W/W15/W15-3502.pdf.

Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2014. Back to basics for monolingual alignment: Exploiting word similarity and contextual evidence. *TACL* 2:219–230. http://aclweb.org/anthology/Q/Q14/Q14-1018.pdf.

Hong Sun and Ming Zhou. 2012. Joint learning of a dual SMT system for paraphrase generation. In *Proc. of ACL'12*. pages 38–42. http://aclweb.org/anthology/P/P12/P12-2008.pdf.

Kristina Toutanova, Chris Brockett, Ke M. Tran, and Saleema Amershi. 2016. A dataset and evaluation metrics for abstractive compression of sentences and short paragraphs. In *Proc. of EMNLP'16*. pages 340–350. http://aclweb.org/anthology/D/D16/D16-1033.pdf.

Sanja Štajner, Hannah Bechara, and Horacio Saggion. 2015. A deeper exploration of the standard PB-SMT approach to text simplification and its evaluation. In *Proc. of ACL'15, Short papers*. pages 823–828. http://aclweb.org/anthology/P/P15/P15-2135.pdf.

Sanja Štajner, Ruslan Mitkov, and Horacio Saggion. 2014. One step closer to automatic evaluation of text simplification systems. *Proc. of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations* pages 1–10. http://www.aclweb.org/anthology/W14-1201.

Sanja Štajner, Maja Popović, Horacio Saggion, Lucia Specia, and Mark Fishel. 2016. Shared task on quality assessment for text simplification. In *Workshop on Quality Assessment for Text Simplification (QATS)*.

Kristian Woodsend and Mirella Lapata. 2011. Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proc. of EMNLP'11*. pages 409–420. http://aclweb.org/anthology/D/D11/D11-1038.pdf.

Sander Wubben, Antal van den Bosch, and Emiel Krahmer. 2012. Sentence simplification by monolingual machine translation. In *Proc. of ACL'12*. pages 1015–1024. http://aclweb.org/anthology/P/P12/P12-1107.pdf.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in current text simplification research: new data can help. *TACL* 3:283–297. http://aclweb.org/anthology/Q/Q15/Q15-1021.pdf.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *TACL* 4:401–415. http://aclweb.org/anthology/Q/Q16/Q16-1029.pdf.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *Proc. of COLING'10*. pages 1353–1361. http://aclweb.org/anthology/C/C10/C10-1152.pdf.

Entity Commonsense Representation for Neural Abstractive Summarization

Reinald Kim Amplayo* and **Seonjae Lim*** and **Seung-won Hwang**
Yonsei University, Seoul, South Korea
{rktamplayo, sun.lim, seungwonh}@yonsei.ac.kr

Abstract

A major proportion of a text summary includes important entities found in the original text. These entities build up the topic of the summary. Moreover, they hold commonsense information once they are linked to a knowledge base. Based on these observations, this paper investigates the usage of linked entities to guide the decoder of a neural text summarizer to generate concise and better summaries. To this end, we leverage on an off-the-shelf entity linking system (ELS) to extract linked entities and propose **Entity2Topic (E2T)**, a module easily attachable to a sequence-to-sequence model that transforms a list of entities into a vector representation of the topic of the summary. Current available ELS's are still not sufficiently effective, possibly introducing unresolved ambiguities and irrelevant entities. We resolve the imperfections of the ELS by (a) encoding entities with selective disambiguation, and (b) pooling entity vectors using firm attention. By applying E2T to a simple sequence-to-sequence model with attention mechanism as base model, we see significant improvements of the performance in the Gigaword (sentence to title) and CNN (long document to multi-sentence highlights) summarization datasets by at least 2 ROUGE points.

1 Introduction

Text summarization is a task to generate a shorter and concise version of a text while preserving the meaning of the original text. The task can be divided into two subtask based on the approach: extractive and abstractive summarization. Extractive summarization is a task to create summaries by pulling out snippets of text form the original text and combining them to form a summary. Abstractive summarization asks to generate summaries from scratch without the restriction to use the available words from the original text. Due to the limitations of extractive summarization on incoherent texts and unnatural methodology (Yao et al., 2017), the research trend has shifted towards abstractive summarization.

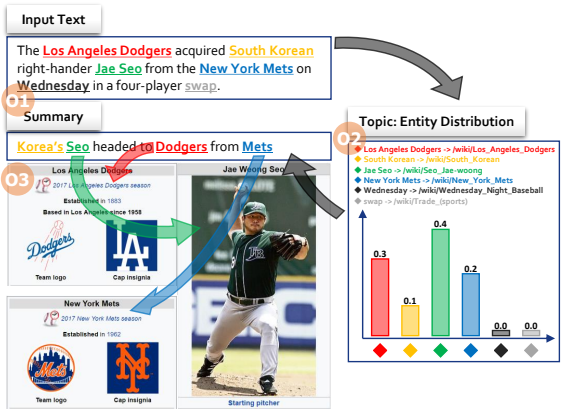


Figure 1: Observations on linked entities in summaries. **O1**: Summaries are mainly composed of entities. **O2**: Entities can be used to represent the topic of the summary. **O3**: Entity commonsense learned from a large corpus can be used.

Sequence-to-sequence models (Sutskever et al., 2014) with attention mechanism (Bahdanau et al., 2014) have found great success in generating abstractive summaries, both from a single sentence (Chopra et al., 2016) and from a long document with multiple sentences (Chen et al., 2016). However, when generating summaries, it is necessary to determine the main topic and to sift out unnecessary information that can be omitted. Sequence-to-sequence models have the tendency to include all the information, relevant or not, that are found in the original text. This may result to unconcise summaries that concentrates wrongly on irrelevant topics. The problem is especially severe when summarizing longer texts.

In this paper, we propose to use entities found in the original text to infer the summary topic, miti-

* Amplayo and Lim are co-first authors with equal contribution. Names are arranged alphabetically.

Proceedings of NAACL-HLT 2018, pages 697–707
New Orleans, Louisiana, June 1 - 6, 2018.

gating the aforementioned problem. Specifically, we leverage on linked entities extracted by employing a readily available entity linking system. The importance of using linked entities in summarization is intuitive and can be explained by looking at Figure 1 as an example. First (**O1** in the Figure), aside from auxiliary words to construct a sentence, a summary is mainly composed of linked entities extracted from the original text. Second (**O2**), we can depict the main topic of the summary as a probability distribution of relevant entities from the list of entities. Finally (**O3**), we can leverage on entity commonsense learned from a separate large knowledge base such as Wikipedia.

To this end, we present a method to effectively apply linked entities in sequence-to-sequence models, called **Entity2Topic (E2T)**. E2T is a module that can be easily attached to any sequence-to-sequence based summarization model. The module encodes the entities extracted from the original text by an entity linking system (ELS), constructs a vector representing the topic of the summary to be generated, and informs the decoder about the constructed topic vector. Due to the imperfections of current ELS's, the extracted linked entities may be too **ambiguous** and **coarse** to be considered relevant to the summary. We solve this issue by using entity encoders with **selective disambiguation** and by constructing topic vectors using **firm attention**.

We experiment on two datasets, Gigaword and CNN, with varying lengths. We show that applying our module to a sequence-to-sequence model with attention mechanism significantly increases its performance on both datasets. Moreover, when compared with the state-of-the-art models for each dataset, the model obtains a comparable performance on the Gigaword dataset where the texts are short, and outperforms all competing models on the CNN dataset where the texts are longer. Furthermore, we provide analysis on how our model effectively uses the extracted linked entities to produce concise and better summaries.

2 Usefulness of linked entities in summarization

In the next subsections, we present detailed arguments with empirical and previously examined evidences on the observations and possible issues when using linked entities extracted by an entity linking system (ELS) for generating abstractive summaries. For this purpose, we use the development sets of the Gigaword dataset provided in (Rush et al., 2015) and of the CNN dataset provided in (Hermann et al., 2015) as the experimental data for quantitative evidence and refer the readers to Figure 1 as the running example.

2.1 Observations

As discussed in Section 1, we find three observations that show the usefulness of linked entities for abstractive summarization.

First, summaries are mainly composed of linked entities extracted from the original text. In the example, it can be seen that the summary contains four words that refer to different entities. In fact, all noun phrases in the summary mention at least one linked entity. In our experimental data, we extract linked entities from the original text and compare them to the noun phrases found in the summary. We report that 77.1% and 75.1% of the noun phrases on the Gigaword and CNN datasets, respectively, contain at least one linked entity, which confirms our observation.

Second, linked entities can be used to represent the topic of the summary, defined as a multinomial distribution over entities, as graphically shown in the example, where the probabilities refer to the relevance of the entities. Entities have been previously used to represent topics (Newman et al., 2006), as they can be utilized as a controlled vocabulary of the main topics in a document (Hulpus et al., 2013). In the example, we see that the entity "*Jae Seo*" is the most relevant because it is the subject of the summary, while the entity "*South Korean*" is less relevant because it is less important when constructing the summary.

Third, we can make use of the entity commonsense that can be learned as a continuous vector representation from a separate larger corpus (Ni et al., 2016; Yamada et al., 2017). In the example, if we know that the entities "*Los Angeles Dodgers*" and "*New York Mets*" are American baseball teams and "*Jae Seo*" is a baseball player associated with the teams, then we can use this information to generate more coherent summaries. We find that 76.0% of the extracted linked entities are covered by the pre-trained vectors[1] in our experimental data, proving our third observation.

[1] https://github.com/idio/wiki2vec

2.2 Possible issues

Despite its usefulness, linked entities extracted from ELS's have issues because of low precision rates (Hasibi et al., 2016) and design challenges in training datasets (Ling et al., 2015). These issues can be summarized into two parts: ambiguity and coarseness.

First, the extracted entities may be ambiguous. In the example, the entity "*South Korean*" is ambiguous because it can refer to both the South Korean person and the South Korean language, among others[2]. In our experimental data, we extract (1) the top 100 entities based on frequency, and (2) the entities extracted from 100 randomly selected texts, and check whether they have disambiguation pages in Wikipedia or not. We discover that 71.0% of the top 100 entities and 53.6% of the entities picked at random have disambiguation pages, which shows that most entities are prone to ambiguity problems.

Second, the linked entities may also be too common to be considered an entity. This may introduce *errors* and *irrelevance* to the summary. In the example, "*Wednesday*" is erroneous because it is wrongly linked to the entity "*Wednesday Night Baseball*". Also, "*swap*" is irrelevant because although it is linked correctly to the entity "*Trade (Sports)*", it is too common and irrelevant when generating the summaries. In our experimental data, we randomly select 100 data instances and tag the correctness and relevance of extracted entities into one of four labels: A: correct and relevant, B: correct and somewhat relevant, C: correct but irrelevant, and D: incorrect. Results show that 29.4%, 13.7%, 30.0%, and 26.9% are tagged with A, B, C, and D, respectively, which shows that there is a large amount of incorrect and irrelevant entities.

3 Our model

To solve the issues described above, we present **Entity2Topic (E2T)**, a module that can be easily attached to any sequence-to-sequence based abstractive summarization model. E2T encodes the linked entities extracted from the text and transforms them into a single topic vector. This vector is ultimately concatenated to the decoder hidden state vectors. The module contains two submodules specifically for the issues presented by the entity linking systems: the entity encoding submodule with selective disambiguation and the pooling submodule with firm attention.

Overall, our full architecture can be illustrated as in Figure 2, which consists of an entity linking system (ELS), a sequence-to-sequence with attention mechanism model, and the E2T module. We note that our proposed module can be easily attached to more sophisticated abstractive summarization models (Zhou et al., 2017; Tan et al., 2017) that are based on the traditional encoder-decoder framework and consequently can produce better results. The code of the base model and the E2T are available online[3].

3.1 Base model

As our base model, we employ a basic encoder-decoder RNN used in most neural machine translation (Bahdanau et al., 2014) and text summarization (Nallapati et al., 2016) tasks. We employ a two-layer bidirectional GRU (BiGRU) as the recurrent unit of the encoder. The BiGRU consists of a forward and backward GRU, which results to sequences of forward and backward hidden states $(\overrightarrow{h}_1, \overrightarrow{h}_2, ..., \overrightarrow{h}_n)$ and $(\overleftarrow{h}_1, \overleftarrow{h}_2, ..., \overleftarrow{h}_n)$, respectively:

$$\overrightarrow{h}_i = GRU(x_i, \overrightarrow{h}_{i-1})$$
$$\overleftarrow{h}_i = GRU(x_i, \overleftarrow{h}_{i+1})$$

The forward and backward hidden states are concatenated to get the hidden state vectors of the tokens (i.e. $h_i = [\overrightarrow{h}_i; \overleftarrow{h}_i]$). The final states of the forward and backward GRU are also concatenated to create the final text representation vector of the encoder $s = [\overrightarrow{h}_n; \overleftarrow{h}_1]$. These values are calculated per layer, where x_t of the second layer is h_t of the first layer. The final text representation vectors are projected by a fully connected layer and are passed to the decoder as the initial hidden states $s_0 = s$.

For the decoder, we use a two-layer unidirectional GRU with attention. At each time step t, the previous token y_{t-1}, the previous hidden state s_{t-1}, and the previous context vector c_{t-1} are passed to a GRU to calculate the new hidden state s_t, as shown in the equation below.

$$s_t = GRU(w_{t-1}, s_{t-1}, c_{t-1})$$

[2] https://en.wikipedia.org/wiki/South_Korean

[3] https://github.com/rktamplayo/Entity2Topic

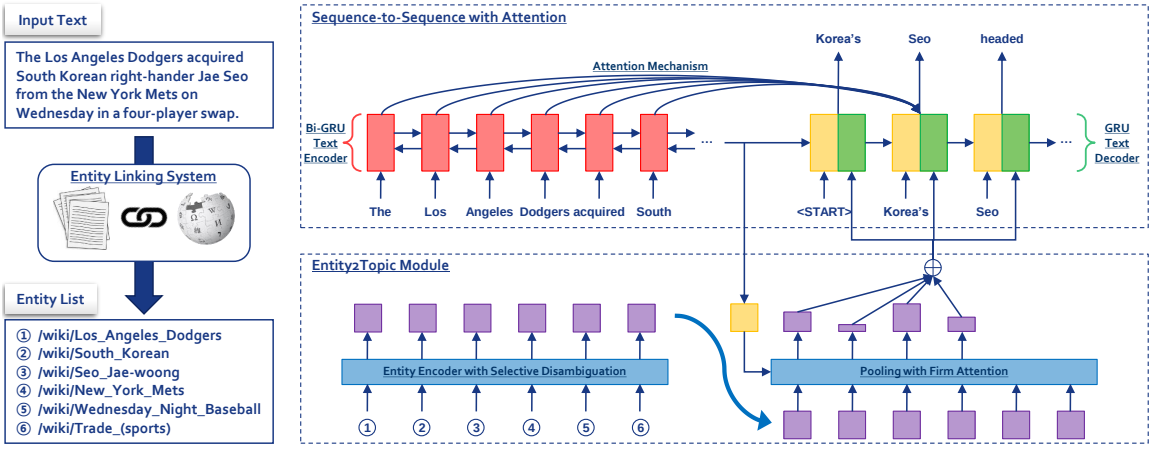


Figure 2: Full architecture of our proposed sequence-to-sequence model with Entity2Topic (E2T) module.

The context vector c_t is computed using the additive attention mechanism (Bahdanau et al., 2014), which matches the current decoder state s_t and each encoder state h_i to get an importance score. The scores are then passed to a softmax and are used to pool the encoder states using weighted sum. The final pooled vector is the context vector, as shown in the equations below.

$$g_{t,i} = v_a^\top tanh(W_a s_{t-1} + U_a h_i)$$
$$a_{t,i} = \frac{exp(g_{t,i})}{\sum_i exp(g_{t,i})}$$
$$c_t = \sum_i a_{t,i} h_i$$

Finally, the previous token y_{t-1}, the current context vector c_t, and the current decoder state s_t are used to generate the current word y_t with a softmax layer over the decoder vocabulary, as shown below.

$$o_t = W_w w_{t-1} + W_c c_t + W_s s_t$$
$$p(y_t|y_{<t}) = softmax(W_o o_t)$$

3.2 Entity encoding submodule

After performing entity linking to the input text using the ELS, we receive a sequential list of linked entities, arranged based on their location in the text. We embed these entities to d-dimensional vectors $E = \{e_1, e_2, ..., e_m\}$ where $e_i \in \mathbb{R}^d$. Since these entities may still contain ambiguity, it is necessary to resolve them before applying them to the base model. Based on the idea that an ambiguous entity can be disambiguated using its neighboring entities, we introduce two kinds of disambiguating encoders below.

Globally disambiguating encoder One way to disambiguate an entity is by using all the other entities, putting more importance to entities that are nearer. For this purpose, we employ an RNN-based model to globally disambiguate the entities. Specifically, we use BiGRU and concatenate the forward and backward hidden state vectors as the new entity vector:

$$\overrightarrow{h}_i = GRU(e_i, \overrightarrow{h}_{i-1})$$
$$\overleftarrow{h}_i = GRU(e_i, \overleftarrow{h}_{i+1})$$
$$e'_i = [\overrightarrow{h}_i; \overleftarrow{h}_i]$$

Locally disambiguating encoder Another way to disambiguate an entity is by using only the direct neighbors of the entity, putting no importance value to entities that are far. To do this, we employ a CNN-based model to locally disambiguate the entities. Specifically, we do the convolution operation using filter matrices $W_f \in \mathbb{R}^{h \times d}$ with filter size h to a window of h words. We do this for different sizes of h. This produces new feature vectors $c_{i,h}$ as shown below, where $f(.)$ is a non-linear function:

$$c_{i,h} = f([e_{i-(h-1)/2}; ...; e_{i+h(+1)/2}]^\top W_f + b_f)$$

The convolution operation reduces the number of entities differently depending on the filter size h. To prevent loss of information and to produce the same amount of feature vectors $c_{i,h}$, we pad the entity list dynamically such that when the filter size is h, the number of paddings on each side is $(h-1)/2$. The filter size h therefore refers to the number of entities used to disambiguate a middle entity. Finally, we concatenate all feature vectors

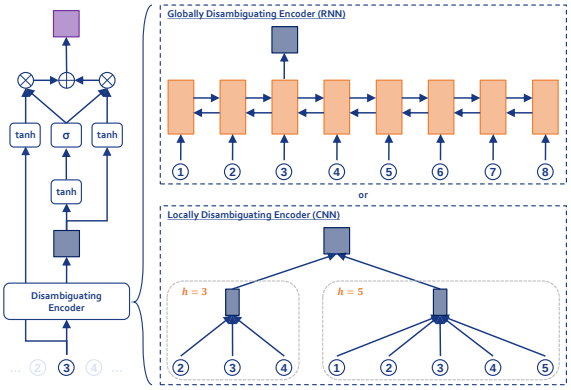


Figure 3: Entity encoding submodule with selective disambiguation applied to the entity ③. The left figure represents the full submodule while the right figure represents the two choices of disambiguating encoders.

of different h's for each i as the new entity vector:

$$e'_i = [c_{i,h_1}; c_{i,h_2}; ...]$$

The question on which disambiguating encoder is better has been a debate; some argued that using only the local context is appropriate (Lau et al., 2013) while some claimed that additionally using global context also helps (Wang et al., 2015). The RNN-based encoder is good as it smartly makes use of all entities, however it may perform bad when there are many entities as it introduces noise when using a far entity during disambiguation. The CNN-based encoder is good as it minimizes the noise by totally ignoring far entities when disambiguating, however determining the appropriate filter sizes h needs engineering. Overall, we argue that when the input text is short (e.g. a sentence), both encoders perform comparably, otherwise when the input text is long (e.g. a document), the CNN-based encoder performs better.

Selective disambiguation It is obvious that not all entities need to be disambiguated. When a correctly linked and already adequately disambiguated entity is disambiguated again, it would make the entity very context-specific and might not be suitable for the summarization task. Our entity encoding submodule therefore uses a selective mechanism that decides whether to use the disambiguating encoder or not. This is done by introducing a selective disambiguation gate d. The final entity vector $\tilde{e}_i$ is calculated as the linear transformation of e_i and e'_i:

$$\begin{aligned} e'_i &= encoder(e_i) \\ d &= \sigma(W_d e'_i + b_d) \\ \tilde{e}_i &= d \times f(W_x e_i + b_x) + \\ &\quad (1-d) \times f(W_y e'_i + b_y) \end{aligned}$$

The full entity encoding submodule is illustrated in Figure 3. Ultimately, the submodule outputs the disambiguated entity vectors $\tilde{E} = \{\tilde{e}_1, \tilde{e}_2, ..., \tilde{e}_m\}$.

3.3 Pooling submodule

The entity vectors $\tilde{E}$ are pooled to create a single topic vector t that represents the topic of the summary. One possible pooling technique is to use soft attention (Xu et al., 2015) on the vectors to determine the importance value of each vector, which can be done by matching each entity vector with the text vector s from the text encoder as the context vector. The entity vectors are then pooled using weighted sum. One problem with soft attention is that it considers *all* entity vectors when constructing the topic vector. However, not all entities are important and necessary when generating summaries. Moreover, a number of these entities may be erroneous and irrelevant, as reported in Section 2.2. Soft attention gives non-negligible important scores to these entities, thus adds unnecessary noise to the construction of the topic vector.

Our pooling submodule instead uses **firm attention** mechanism to consider only top k entities when constructing the topic vector. This is done in a differentiable way as follows:

$$\begin{aligned} G &= v_a^\top tanh(W_a \tilde{E} + U_a s) \\ K &= top\_k(G) \\ P &= sparse\_vector(K, 0, -\infty) \\ g'_i &= g_i + p_i \\ a_i &= \frac{exp(g'_i)}{\sum_i exp(g'_i)} \\ t &= \sum_i a_i \tilde{e}_i \end{aligned}$$

where the functions $K = top\_k(G)$ gets the indices of the top k vectors in G and $P = sparse\_vector(K, 0, -\infty)$ creates a sparse vector where the values of K is 0 and $-\infty$ otherwise[4]. The sparse vector P is added to the original importance score vector G to create a new importance

[4]We use -10^9 to represent $-\infty$.

score vector. In this new vector, important scores of non-top k entities are $-\infty$. When softmax is applied, this gives very small, negligible, and close-to-zero values to non-top k entities. The value k depends on the lengths of the input text and summary. Moreover, when k increases towards infinity, firm attention becomes soft attention. We decide k empirically (see Section 5).

3.4 Extending from the base model

Entity2Topic module extends the base model as follows. The final text representation vector s is used as a context vector when constructing the topic vector t in the pooling submodule. The topic vector t is then concatenated to the decoder hidden state vectors s_i, i.e. $s'_i = [s_i; t]$. The concatenated vector is finally used to create the output vector:

$$o_i = W_w w_{i-1} + W_c c_i + W_s s'_i$$

4 Related work

Due to its recent success, neural network models have been used with competitive results on abstractive summarization. A neural attention model was first applied to the task, easily achieving state-of-the-art performance on multiple datasets (Rush et al., 2015). The model has been extended to instead use recurrent neural network as decoder (Chopra et al., 2016). The model was further extended to use a full RNN encoder-decoder framework and further enhancements through lexical and statistical features (Nallapati et al., 2016). The current state-of-the-art performance is achieved by selectively encoding words as a process of distilling salient information (Zhou et al., 2017).

Neural abstractive summarization models have also been explored to summarize longer documents. Word extraction models have been previously explored, performing worse than sentence extraction models (Cheng and Lapata, 2016). Hierarchical attention-based recurrent neural networks have also been applied to the task, owing to the idea that there are multiple sentences in a document (Nallapati et al., 2016). Finally, distraction-based models were proposed to enable models to traverse the text content and grasp the overall meaning (Chen et al., 2016). The current state-of-the-art performance is achieved by a graph-based attentional neural model, considering the key factors of document summarization such as saliency, fluency and novelty (Tan et al., 2017).

Dataset	Gigaword	CNN
num(data)	4.0M	84K
avg(inputWord)	31.4	774.9
avg(outputWord)	8.2	48.1
min(inputEntity)	1	1
max(inputEntity)	36	743
avg(inputEntity)	4.5	94.6

Table 1: Dataset statistics.

Previous studies on the summarization tasks have only used entities in the preprocessing stage to anonymize the dataset (Nallapati et al., 2016) and to mitigate out-of-vocabulary problems (Tan et al., 2017). Linked entities for summarization are still not properly explored and we are the first to use linked entities to improve the performance of the summarizer.

5 Experimental settings

Datasets We use two widely used summarization datasets with different text lengths. First, we use the Annotated English Gigaword dataset as used in (Rush et al., 2015). This dataset receives the first sentence of a news article as input and use the headline title as the gold standard summary. Since the development dataset is large, we randomly selected 2000 pairs as our development dataset. We use the same held-out test dataset used in (Rush et al., 2015) for comparison. Second, we use the CNN dataset released in (Hermann et al., 2015). This dataset receives the full news article as input and use the human-generated multiple sentence highlight as the gold standard summary. The original dataset has been modified and preprocessed specifically for the document summarization task (Nallapati et al., 2016). In addition to the previously provided datasets, we extract linked entities using Dexter[5] (Ceccarelli et al., 2013), an open source ELS that links text snippets found in a given text to entities contained in Wikipedia. We use the default recommended parameters stated in the website. We summarize the statistics of both datasets in Table 1.

Implementation For both datasets, we further reduce the size of the input, output, and entity vocabularies to at most 50K as suggested in (See et al., 2017) and replace less frequent words to

[5] http://dexter.isti.cnr.it/

"<unk>". We use 300D Glove[6] (Pennington et al., 2014) and 1000D wiki2vec[7] pre-trained vectors to initialize our word and entity vectors. For GRUs, we set the state size to 500. For CNN, we set $h = 3, 4, 5$ with $400, 300, 300$ feature maps, respectively. For firm attention, k is tuned by calculating the perplexity of the model starting with smaller values (i.e. $k = 1, 2, 5, 10, 20, ...$) and stopping when the perplexity of the model becomes worse than the previous model. Our preliminary tuning showed that $k = 5$ for Gigaword dataset and $k = 10$ for CNN dataset are the best choices. We use dropout (Srivastava et al., 2014) on all non-linear connections with a dropout rate of 0.5. We set the batch sizes of Gigaword and CNN datasets to 80 and 10, respectively. Training is done via stochastic gradient descent over shuffled mini-batches with the Adadelta update rule, with l_2 constraint (Hinton et al., 2012) of 3. We perform early stopping using a subset of the given development dataset. We use beam search of size 10 to generate the summary.

Baselines For the Gigaword dataset, we compare our models with the following abstractive baselines: **ABS+** (Rush et al., 2015) is a fine tuned version of ABS which uses an attentive CNN encoder and an NNLM decoder, **Feat2s** (Nallapati et al., 2016) is an RNN sequence-to-sequence model with lexical and statistical features in the encoder, **Luong-NMT** (Luong et al., 2015) is a two-layer LSTM encoder-decoder model, **RAS-Elman** (Chopra et al., 2016) uses an attentive CNN encoder and an Elman RNN decoder, and **SEASS** (Zhou et al., 2017) uses BiGRU encoders and GRU decoders with selective encoding. For the CNN dataset, we compare our models with the following extractive and abstractive baselines: **Lead-3** is a strong baseline that extracts the first three sentences of the document as summary, **LexRank** extracts texts using LexRank (Erkan and Radev, 2004), **Bi-GRU** is a non-hierarchical one-layer sequence-to-sequence abstractive baseline, **Distraction-M3** (Chen et al., 2016) uses a sequence-to-sequence abstractive model with distraction-based networks, and **GBA** (Tan et al., 2017) is a graph-based attentional neural abstractive model. All baseline results used beam search and are gathered from previous papers. Also, we compare our final model **BASE+E2T** with the base model **BASE** and some variants of our model (without selective disambiguation, using soft attention).

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

[7] https://github.com/idio/wiki2vec

Model	RG-1	RG-2	RG-L
BASE: s2s+att	34.14	15.44	32.47
BASE+E2T$_{cnn+sd}$	**37.04**	16.66	**34.93**
BASE+E2T$_{rnn+sd}$	36.89	**16.86**	34.74
BASE+E2T$_{cnn}$	36.56	16.56	34.57
BASE+E2T$_{rnn}$	36.52	16.21	34.32
BASE+E2T$_{cnn+soft}$	36.56	16.44	34.58
BASE+E2T$_{rnn+soft}$	36.38	16.12	34.20
ABS+	29.78	11.89	26.97
Feat2s	32.67	15.59	30.64
Luong-NMT	33.10	14.45	30.71
RAS-Elman	33.78	15.97	31.15
SEASS	**36.15**	**17.54**	**33.63**

Table 2: Results on the Gigaword dataset using the full-length F1 variants of ROUGE.

Model	RG-1	RG-2	RG-L
BASE: s2s+att	25.5	5.8	20.0
BASE+E2T$_{cnn+sd}$	**31.9**	**10.1**	**23.9**
BASE+E2T$_{rnn+sd}$	27.6	7.9	21.5
BASE+E2T$_{cnn}$	26.6	7.3	20.7
BASE+E2T$_{rnn}$	26.1	6.9	20.1
BASE+E2T$_{cnn+soft}$	26.6	7.0	20.6
BASE+E2T$_{rnn+soft}$	25.0	6.7	19.8
Lead-3	26.1	9.6	17.8
LexRank	26.1	9.6	17.7
Bi-GRU	19.5	5.2	15.0
Distraction-M3	27.1	8.2	18.7
GBA	**30.3**	**9.8**	**20.0**

Table 3: Results on the CNN dataset using the full-length F1 ROUGE metric.

6 Results

We report the ROUGE F1 scores for both datasets of all the competing models using ROUGE F1 scores (Lin, 2004). We report the results on the Gigaword and the CNN dataset in Table 2 and Table 3, respectively. In Gigaword dataset where the texts are short, our best model achieves a comparable performance with the current state-of-the-art. In CNN dataset where the texts are longer, our best model outperforms all the previous models. We emphasize that E2T module is easily attachable to better models, and we expect E2T to improve

Model	1st	2nd	3rd	4th	mean
GOLD	0.27	0.34	0.21	0.18	2.38
BASE	0.14	0.15	0.28	0.43	3.00
BASE+E2T$_{rnn}$	0.12	0.24	0.39	0.25	2.77
BASE+E2T$_{cnn}$	**0.47**	0.27	0.12	0.14	**1.93**

Table 4: Human evaluations on the Gigaword dataset. Bold-faced values are the best while red-colored values are the worst among the values in the evaluation metric.

their performance as well. Overall, E2T achieves a significant improvement over the baseline model BASE, with at least 2 ROUGE-1 points increase in the Gigaword dataset and 6 ROUGE-1 points increase in the CNN dataset. In fact, all variants of E2T gain improvements over the baseline, implying that leveraging on linked entities improves the performance of the summarizer. Among the model variants, the CNN-based encoder with selective disambiguation and firm attention performs the best.

Automatic evaluation on the Gigaword dataset shows that the CNN and RNN variants of BASE+E2T have similar performance. To break the tie between both models, we also conduct human evaluation on the Gigaword dataset. We instruct two annotators to read the input sentence and rank the competing summaries from first to last according to their relevance and fluency: (a) the original summary GOLD, and from models (b) BASE, (c) BASE+E2T$_{cnn}$, and (d) BASE+E2T$_{rnn}$. We then compute (i) the proportion of every ranking of each model and (ii) the mean rank of each model. The results are reported in Table 4. The model with the best mean rank is BASE+E2T$_{cnn}$, followed by GOLD, then by BASE+E2T$_{rnn}$ and BASE, respectively. We also perform ANOVA and post-hoc Tukey tests to show that the CNN variant is significantly ($p < 0.01$) better than the RNN variant and the base model. The RNN variant does not perform as well as the CNN variant, contrary to the automatic ROUGE evaluation above. Interestingly, the CNN variant produces better (but with no significant difference) summaries than the gold summaries. We posit that this is due to the fact that the article title does not correspond to the summary of the first sentence.

Selective disambiguation of entities We show the effectiveness of the selective disambiguation gate d in selecting which entities to disambiguate or not. Table 6 shows a total of four different examples of two entities with the highest/lowest d values. In the first example, sentence **E1.1** contains the entity "*United States*" and is linked with the country entity of the same name, however the correct linked entity should be "*United States Davis Cup team*", and therefore is given a high d value. On the other hand, sentence **E1.2** is linked correctly to the country "*United States*", and thus is given a low d value.. The second example provides a similar scenario, where sentence **E2.1** is linked to the entity "*Gold*" but should be linked to the entity "*Gold medal*". Sentence **E2.2** is linked correctly to the chemical element. Hence, the former case received a high value d while the latter case received a low d value.

Entities as summary topic Finally, we provide one sample for each dataset in Table 5 for case study, comparing our final model that uses **firm** attention (BASE$_{cnn+sd}$), a variant that uses **soft** attention (BASE$_{cnn+soft}$), and the **baseline** model (BASE). We also show the attention weights of the **firm** and **soft** models.

In the Gigaword example, we find three observations. First, the base model generated a less informative summary, not mentioning "*mexico state*" and "*first edition*". Second, the soft model produced a factually wrong summary, saying that "*guadalajara*" is a mexican state, while actually it is a city. Third, the firm model is able to solve the problem by focusing only on the five most important entities, eliminating possible noise such as "*Unk*" and less crucial entities such as "*Country club*". We can also see the effectiveness of the selective disambiguation in this example, where the entity "*U.S. state*" is corrected to mean the entity "*Mexican state*" which becomes relevant and is therefore selected.

In the CNN example, we also find that the baseline model generated a very erroneous summary. We argue that this is because the length of the input text is long and the decoder is not guided as to which topics it should focus on. The soft model generated a much better summary, however it focuses on the wrong topics, specifically on "*Iran's nuclear program*", making the summary less general. A quick read of the original article tells us that the main topic of the article is all about the two political parties arguing over the deal with Iran. However, the entity "*nuclear*" appeared a lot in the article, which makes the soft model wrongly focus on the "*nuclear*" entity. The firm model produced the more relevant summary, focusing on the po-

Gigaword Dataset Example

Original	western mexico **@state** **@jalisco** will host the first edition of the **@UNK** dollar **@lorena ochoa** invitation **@golf** tournament on nov. ##-## #### , in **@guadalajara** **@country club** , the **@lorena ochoa** foundation said in a statement on wednesday .							
Gold	mexico to host lorena ochoa golf tournament in ####							
Baseline	guadalajara to host ochoa tournament tournament							
Entities:	U.S. state	Jalisco	Unk	Lorena Ochoa	Golf	Guadalajara	Country club	Lorena Ochoa
Soft	0.083	0.086	0.124	0.101	0.080	0.161	0.189	0.177
	mexico state guadalajara to host ochoa ochoa invitation							
Firm	0.173	0.197	0.000	0.213	0.215	0.000	0.00	0.202
	mexican state to host first edition of ochoa invitation							

CNN Dataset Example

Original	URL: http://edition.cnn.com/2015/04/05/politics/netanyahu-iran-deal/index.html
Gold	netanyahu says third option is " standing firm " to get a better deal . political sparring continues in u.s. over the deal with iran .
Baseline	netanyahu says he is a country of " UNK cheating " and that it is a country of " UNK cheating " netanyahu says he is a country of " UNK cheating " and that " is a very bad deal " he says he says he says the plan is a country of " UNK cheating " and that it is a country of " UNK cheating " he says the u.s. is a country of " UNK cheating " and that is a country of " UNK cheating "
Soft	benjamin netanyahu : " i think there 's a third alternative , and that is standing firm , " netanyahu tells cnn . he says he does not roll back iran 's nuclear ambitions . " it does not roll back iran 's nuclear program . "
Firm	new : netanyahu : " i think there 's a third alternative , and that is standing firm , " netanyahu says . obama 's comments come as democrats and republicans spar over the framework announced last week to lift western sanctions on iran .

Table 5: Examples from Gigaword and CNN datasets and corresponding summaries generated by competing models. The tagged part of text is marked **bold** and preceded with at sign (@). The red color fill represents the attention scores given to each entity. We only report the attention scores of entities in the Gigaword example for conciseness since there are 80 linked entities in the CNN example.

Text	d
Linked entity: https://en.wikipedia.org/wiki/United_States	
E1.1: andy roddick got the better of dmitry tursunov in straight sets on friday , assuring the **@united states** a #-# lead over defending champions russia in the #### davis cup final .	0.719
E1.2: sir alex ferguson revealed friday that david beckham 's move to the **@united states** had not surprised him because he knew the midfielder would not return to england if he could not come back to manchester united .	0.086
Linked entity: https://en.wikipedia.org/wiki/Gold	
E2.1: following is the medal standing at the ##th olympic winter games -lrb- tabulated under team , **@gold** , silver and bronze -rrb- : UNK	0.862
E2.2: **@gold** opened lower here on monday at ###.##-### .## us dollars an ounce , against friday 's closing rate of ###.##-### .## .	0.130

Table 6: Examples with highest/lowest disambiguation gate d values of two example entities (*United States* and *gold*). The tagged part of text is marked **bold** and preceded with at sign (@).

litical entities (e.g. "*republicans*", "*democrats*"). This is due to the fact that only the $k = 10$ most important elements are attended to create the summary topic vector.

7 Conclusion

We proposed to leverage on linked entities to improve the performance of sequence-to-sequence models on neural abstractive summarization task. Linked entities are used to guide the decoding process based on the summary topic and commonsense learned from a knowledge base. We introduced Entity2Topic (E2T), a module that is easily attachable to any model using an encoder-decoder framework. E2T applies linked entities into the summarizer by encoding the entities with selective disambiguation and pooling them into one summary topic vector with firm attention mechanism. We showed that by applying E2T to a basic sequence-to-sequence model, we achieve significant improvements over the base model and consequently achieve a comparable performance with more complex summarization models.

Acknowledgement

We would like to thank the three anonymous reviewers for their valuable feedback. This work was supported by Microsoft Research, and Institute for Information communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.2017-0-01778 , Development of Explainable Humanlevel Deep Machine Learning Inference Framework). S. Hwang is a corresponding author.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Diego Ceccarelli, Claudio Lucchese, Salvatore Orlando, Raffaele Perego, and Salvatore Trani. 2013. Dexter: an open source framework for entity linking. In *Proceedings of the sixth international workshop on Exploiting semantic annotations in information retrieval*. ACM, pages 17–20.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for document summarization. *arXiv preprint arXiv:1610.08462* .

Jianpeng Cheng and Mirella Lapata. 2016. Neural

summarization by extracting sentences and words. *arXiv preprint arXiv:1603.07252* .

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 93–98.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research* 22:457–479.

Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. 2016. On the reproducibility of the tagme entity linking system. In *European Conference on Information Retrieval*. Springer, pages 436–449.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*. pages 1693–1701.

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

Ioana Hulpus, Conor Hayes, Marcel Karnstedt, and Derek Greene. 2013. Unsupervised graph-based topic labelling using dbpedia. In *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM, pages 465–474.

Jey Han Lau, Paul Cook, and Timothy Baldwin. 2013. unimelb: Topic modelling-based word sense induction for web snippet clustering. In *SemEval@ NAACL-HLT*. pages 217–221.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*. Barcelona, Spain, volume 8.

Xiao Ling, Sameer Singh, and Daniel S Weld. 2015. Design challenges for entity linking. *Transactions of the Association for Computational Linguistics* 3:315–328.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* .

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023* .

David Newman, Chaitanya Chemudugunta, Padhraic Smyth, and Mark Steyvers. 2006. Analyzing entities and topics in news articles using statistical topic models. In *ISI*. Springer, pages 93–104.

Yuan Ni, Qiong Kai Xu, Feng Cao, Yosi Mass, Dafna Sheinwald, Hui Jia Zhu, and Shao Sheng Cao. 2016. Semantic documents relatedness using concept graph representation. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. ACM, pages 635–644.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pages 1532–1543.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685* .

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368* .

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research* 15(1):1929–1958.

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

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 1171–1181.

Jing Wang, Mohit Bansal, Kevin Gimpel, Brian D Ziebart, and T Yu Clement. 2015. A sense-topic model for word sense induction with unsupervised data enrichment. *Transactions of the Association for Computational Linguistics* 3:59–71.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*. pages 2048–2057.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2017. Learning distributed representations of texts and entities from knowledge base. *arXiv preprint arXiv:1705.02494* .

Jin-ge Yao, Xiaojun Wan, and Jianguo Xiao. 2017. Recent advances in document summarization. *Knowledge and Information Systems* pages 1–40.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. *arXiv preprint arXiv:1704.07073* .

NEWSROOM: A Dataset of 1.3 Million Summaries with Diverse Extractive Strategies

Max Grusky[1,2], **Mor Naaman**[2], **Yoav Artzi**[1,2]
[1]Department of Computer Science, [2]Cornell Tech
Cornell University, New York, NY 10044
{grusky@cs, mor@jacobs, yoav@cs}.cornell.edu

Abstract

We present NEWSROOM, a summarization dataset of 1.3 million articles and summaries written by authors and editors in newsrooms of 38 major news publications. Extracted from search and social media metadata between 1998 and 2017, these high-quality summaries demonstrate high diversity of summarization styles. In particular, the summaries combine *abstractive* and *extractive* strategies, borrowing words and phrases from articles at varying rates. We analyze the extraction strategies used in NEWSROOM summaries against other datasets to quantify the diversity and difficulty of our new data, and train existing methods on the data to evaluate its utility and challenges. The dataset is available online at summari.es.

1 Introduction

The development of learning methods for automatic summarization is constrained by the limited high-quality data available for training and evaluation. Large datasets have driven rapid improvement in other natural language generation tasks, such as machine translation, where data size and diversity have proven critical for modeling the alignment between source and target texts (Tiedemann, 2012). Similar challenges exist in summarization, with the additional complications introduced by the length of source texts and the diversity of summarization strategies used by writers. Access to large-scale high-quality data is an essential prerequisite for making substantial progress in summarization. In this paper, we present NEWSROOM, a dataset with 1.3 million news articles and human-written summaries.

NEWSROOM's summaries were written by authors and editors in the newsrooms of news, sports, entertainment, financial, and other publications. The summaries were published with articles as HTML metadata for social media services and search engines page descriptions. NEWSROOM summaries are written by humans, for common readers, and with the explicit purpose of summarization. As a result, NEWSROOM is a nearly two decade-long snapshot representing how single-document summarization is used in practice across a variety of sources, writers, and topics.

Abstractive Summary: South African photographer Anton Hammerl, *missing* in Libya *since* April *4th*, was *killed* in Libya *more than* a *month ago*.

Mixed Summary: A major climate protest in New York on Sunday *could* mark a seminal *shift* in the *politics* of global warming, just *ahead* of the *U.N.* Climate Summit.

Extractive Summary: A person familiar with the search *tells* The Associated Press that Texas has *offered* its head coaching job to Louisvilles Charlie Strong and he is expected to accept.

Figure 1: NEWSROOM summaries showing different extraction strategies, from time.com, mashable.com, and foxsports.com. Multi-word phrases shared between article and summary are underlined. Novel words used only in the summary are italicized.

Identifying large, high-quality resources for summarization has called for creative solutions in the past. This includes using news headlines as summaries of article prefixes (Napoles et al., 2012; Rush et al., 2015), concatenating bullet points as summaries (Hermann et al., 2015; See et al., 2017), or using librarian archival summaries (Sandhaus, 2008). While these solutions provide large scale data, it comes at the cost of how well they reflect the summarization problem or their focus on very specific styles of summarizations, as we discuss in Section 4. NEWSROOM is distinguished from these resources in its combination of size and diversity. The summaries were written with the explicit goal of concisely summarizing news articles over almost two decades.

Proceedings of NAACL-HLT 2018, pages 708–719
New Orleans, Louisiana, June 1 - 6, 2018.

Rather than rely on a single source, the dataset includes summaries from 38 major publishers. This diversity of sources and time span translate into a diversity of summarization styles.

We explore NEWSROOM to better understand the dataset and how summarization is used in practice by newsrooms. Our analysis focuses on a key dimension, *extractivenss* and *abstractiveness*: extractive summaries frequently borrow words and phrases from their source text, while abstractive summaries describe the contents of articles primarily using new language. We develop measures designed to quantify extractiveness and use these measures to subdivide the data into extractive, mixed, and abstractive subsets, as shown in Figure 1, displaying the broad set of summarization techniques practiced by different publishers.

Finally, we analyze the performance of three summarization models as baselines for NEWSROOM to better understand the challenges the dataset poses. In addition to automated ROUGE evaluation (Lin, 2004a,b), we design and execute a benchmark human evaluation protocol to quantify the output summaries relevance and quality. Our experiments demonstrate that NEWSROOM presents an open challenge for summarization systems, while providing a large resource to enable data-intensive learning methods. The dataset and evaluation protocol are available online at summari.es.

2 Existing Datasets

There are a several frequently used summarization datasets. Listed in Figure 2 are examples from four datasets. The examples are chosen to be representative: they have scores within 5% of their dataset average across our analysis measures (Section 4). To illustrate the extractive and abstractive nature of summaries, we underline multi-word phrases shared between the article and summary, and italicize *words* used only in the summary.

2.1 Document Understanding Conference

Datasets produced for the Document Understanding Conference (DUC)[1] are small, high-quality datasets developed to evaluate summarization systems (Harman and Over, 2004; Dang, 2006).

DUC data consist of newswire articles paired with human summaries written specifically for DUC. One distinctive feature of the DUC datasets

[1] http://duc.nist.gov/

DUC

Example Summary: Floods hit *north* Mozambique as aid to flooded *south continues*

Start of Article: MAPUTO, Mozambique (AP) — Just as aid agencies were making headway in feeding hundreds of thousands displaced by flooding in southern and central Mozambique, new floods hit a remote northern region Monday. The Messalo River overflowed [...]

Gigaword

Example Summary: Seve *gets* invite to US Open

Start of Article: Seve Ballesteros will be playing in next month's US Open after all. The USGA decided Tuesday to give the Spanish star a special exemption. American Ben Crenshaw was also given a special exemption by the United States Golf Association. Earlier this week [...]

New York Times Corpus

Example Summary: *Annual* New York City Toy Fair opens in Manhattan; feud between Toy Manufacturers of America and its landlord at International Toy Center *leads* to confusion and *turmoil* as registration *begins*; *dispute discussed.*

Start of Article: There was toylock when the Toy Fair opened in Manhattan yesterday. The reason? A family feud between the Toy Manufacturers of America and its landlord at Fifth Avenue and 23d Street. Toy buyers and exhibitors arriving to attend the kickoff of the [...]

CNN / Daily Mail

Example Summary:

- Eight Al Jazeera journalists are named on *an* Egyptian charge sheet, the network *says*
- The eight were *among* 20 people named 'Most are not employees of Al Jazeera," the network said
- The eight include three journalists jailed in Egypt

Start of Article: Egyptian authorities have served Al Jazeera with a charge sheet that identifies eight of its staff on a list of 20 people – all believed to be journalists – for allegedly conspiring with a terrorist group, the network said Wednesday. The 20 are wanted by Egyptian [...]

Figure 2: Example summaries for existing datasets.

is the availability of multiple reference summaries for each article. This is a major advantage of DUC compared to other datasets, especially when evaluating with ROUGE (Lin, 2004b,a), which was designed to be used with multiple references. However, DUC datasets are small, which makes it difficult to use them as training data.

DUC summaries are often used in conjunction with larger training datasets, including Gigaword (Rush et al., 2015; Chopra et al., 2016), CNN / Daily Mail (Nallapati et al., 2017; Paulus et al., 2017; See et al., 2017), or Daily Mail alone (Nallapati et al., 2016b; Cheng and Lapata, 2016). The data have also been used to evaluate

unsupervised methods (Dorr et al., 2003; Mihalcea and Tarau, 2004; Barrios et al., 2016).

2.2 Gigaword

The Gigaword Corpus (Napoles et al., 2012) contains nearly 10 million documents from seven newswire sources, including the Associated Press, New York Times Newswire Service, and Washington Post Newswire Service. Compared to other existing datasets used for summarization, the Gigaword corpus is the largest and most diverse in its sources. While Gigaword does not contain summaries, prior work uses Gigaword headlines as simulated summaries (Rush et al., 2015; Chopra et al., 2016). These systems are trained on Gigaword to recreate headlines given the first sentence of an article. When used this way, Gigaword's simulated summaries are shorter than most natural summary text. Gigaword, along with similar text-headline datasets (Filippova and Altun, 2013), are also used for the related sentence compression task (Dorr et al., 2003; Filippova et al., 2015).

2.3 New York Times Corpus

The New York Times Annotated Corpus (Sandhaus, 2008) is the largest summarization dataset currently available. It consists of carefully curated articles from a single source, The New York Times. The corpus contains several hundred thousand articles written between 1987–2007 that have paired summaries. The summaries were written for the corpus by library scientists, rather than at the time of publication. Our analysis in Section 4 reveals that the data are somewhat biased toward extractive strategies, making it particularly useful as an extractive summarization dataset. Despite this, limited work has used this dataset for summarization (Hong and Nenkova, 2014; Durrett et al., 2016; Paulus et al., 2017).

2.4 CNN / Daily Mail

The CNN / Daily Mail question answering dataset (Hermann et al., 2015) is frequently used for summarization. The dataset includes CNN and Daily Mail articles, each associated with several bullet point descriptions. When used in summarization, the bullet points are typically concatenated into a single summary.[2] The dataset has been used for summarization as is (See et al., 2017), or after pre-processing for entity anonymization (Nallapati et al., 2017). This different usage makes comparisons between systems using these data challenging. Additionally, some systems use both CNN and Daily Mail for training (Nallapati et al., 2017; Paulus et al., 2017; See et al., 2017), whereas others use only Daily Mail articles (Nallapati et al., 2016b; Cheng and Lapata, 2016). Our analysis shows that the CNN / Daily Mail summaries have strong bias toward extraction (Section 4). Similar observations about the data were made by Chen et al. (2016) with respect to the question answering task.

3 Collecting NEWSROOM Summaries

The NEWSROOM dataset was collected using social media and search engine metadata. To create the dataset, we performed a Web-scale crawling of over 100 million pages from a set of online publishers. We identify newswire articles and use the summaries provided in the HTML metadata. These summaries were created to be used in search engines and social media.

We collected HTML pages and metadata using the Internet Archive (Archive.org), accessing archived pages of a large number of popular news, sports, and entertainment sites. Using Archive.org provides two key benefits. First, the archive provides an API that allows for collection of data across time, not limited to recently available articles. Second, the archived URLs of the dataset articles are immutable, allowing distribution of this dataset using a thin, URL-only list.

The publisher sites we crawled were selected using a combination of Alexa.com top overall sites, as well as Alexa's top news sites.[3] We supplemented the lists with older lists published by Google of the highest-traffic sites on the Web.[4] We excluded sites such as Reddit that primarily aggregate rather than produce content, as well as publisher sites that proved to have few or no articles with summary metadata available, or have articles primarily in languages other than English. This process resulted in a set of 38 publishers that were included in the dataset.

[2] https://github.com/abisee/cnn-dailymail

[3] Alexa removed the extended public list in 2017, see: https://web.archive.org/web/2016/https://www.alexa.com/topsites/category/News

[4] Google removed this list in 2013, see: https://web.archive.org/web/2012/http://www.google.com/adplanner/static/top1000

3.1 Content Scraping

We used two techniques to identify article pages from the selected publishers on Archive.org: the search API and index-page crawl. The API allows queries using URL pattern matching, which focuses article crawling on high-precision subdomains or paths. We used the API to search for content from the publisher domains, using specific patterns or post-processing filtering to ensure article content. In addition, we used Archive.org to retrieve the historical versions of the home page for all publisher domains. The archive has content from 1998 to 2017 with varying degrees of time resolution. We obtained at least one snapshot of each page for every available day. For each snapshot, we retrieved all articles listed on the page.

For both search and crawled URLs, we performed article de-duplication using URLs to control for varying URL fragments, query parameters, protocols, and ports. When performing the merge, we retained only the earliest article version available to prevent the collection of stale summaries that are not updated when articles are changed.

3.2 Content Extraction

Following identification and de-duplication, we extracted the article texts and summaries and further cleaned and filtered the dataset.

Article Text We used Readability[5] to extract HTML body content. Readability uses HTML heuristics to extract the main content and title of a page, producing article text without extraneous HTML markup and images. Our preliminary testing, as well as comparison by Peters (2015), found Readability to be one of the highest accuracy content extraction algorithms available. To exclude inline advertising and image captions sometimes present in extractions, we applied additional filtering of paragraphs with fewer than five words. We excluded articles with no body text extracted.

Summary Metadata We extracted the article summaries from the metadata available in the HTML pages of articles. These summaries are often written by newsroom editors and journalists to appear in social media distribution and search results. While there is no standard metadata format for summaries online, common fields are often present in the page's HTML. Popular metadata field types include: *og:description*, *twitter:description*, and *description*. In cases where different metadata summaries were available, and were different, we used the first field available according to the order above. We excluded articles with no summary text of any type. We also removed article-summary pairs with a high amount of precisely-overlapping text to remove rule-based automatically-generated summaries fully copied from the article (e.g., the first paragraph).

[5] https://pypi.org/project/readability-lxml/0.6.2/

Dataset Size	1,321,995 articles
Training Set Size	995,041 articles
Mean Article Length	658.6 words
Mean Summary Length	26.7 words
Total Vocabulary Size	6,925,712 words
Occurring 10+ Times	784,884 words

Table 1: Dataset Statistics

3.3 Building the Dataset

Our scraping and extraction process resulted in a set of 1,321,995 article-summary pairs. Simple dataset statistics are shown in Table 1. The data are divided into training (76%), development (8%), test (8%), and unreleased test (8%) datasets using a hash function of the article URL. We use the articles' Archive.org URLs for lightweight distribution of the data. Archive.org is an ideal platform for distributing the data, encouraging its users to scrape its resources. We provide the extraction and analysis scripts used during data collection for reproducing the full dataset from the URL list.

4 Data Analysis

NEWSROOM contains summaries from different topic domains, written by many authors, over the span of more than two decades. This diversity is an important aspect of the dataset. We analyze the data to quantify the differences in summarization styles and techniques between the different publications to show the importance of reflecting this diversity. In Sections 6 and 7, we examine the effect of the dataset diversity on the performance of a variety of summarization systems.

4.1 Characterizing Summarization Strategies

We examine summarization strategies using three measures that capture the degree of text overlap between the summary and article, and the rate of compression of the information conveyed.

Given an article text $A = \langle a_1, a_2, \ldots, a_n \rangle$ consisting of a sequence of tokens a_i and the corresponding article summary $S = \langle s_1, s_2, \cdots, s_m \rangle$ consisting of tokens s_i, the set of extractive frag-

```
function F(A, S)
    F ← ∅, ⟨i, j⟩ ← ⟨1, 1⟩
    while i ≤ |S| do
        f ← ⟨ ⟩
        while j ≤ |A| do
            if s_i = a_j then
                ⟨i', j'⟩ ← ⟨i, j⟩
                while s_{i'} = a_{j'} do
                    ⟨i', j'⟩ ← ⟨i' + 1, j' + 1⟩
                if |f| < (i' − i − 1) then
                    f ← ⟨s_i ··· s_{i'−1}⟩
                j ← j'
            else
                j ← j + 1
        ⟨i, j⟩ ← ⟨i + max{|f|, 1}, 1⟩
        F ← F ∪ {f}
    return F
```

Figure 3: Procedure to compute the set $\mathcal{F}(A, S)$ of extractive phrases in summary S extracted from article A. For each sequential token of the summary, s_i, the procedure iterates through tokens of the text, a_j. If tokens s_i and a_j match, the longest shared token sequence after s_i and a_j is marked as the extraction starting at s_i.

ments $\mathcal{F}(A, S)$ is the set of shared sequences of tokens in A and S. We identify these extractive fragments of an article-summary pair using a greedy process. We process the tokens in the summary in order. At each position, if there is a sequence of tokens in the source text that is prefix of the remainder of the summary, we mark this prefix as extractive and continue. We prefer to mark the longest prefix possible at each step. Otherwise, we mark the current summary token as abstractive. The set $\mathcal{F}(A, S)$ includes all the tokens sequences identified as extractive. Figure 3 formally describes this procedure. Underlined phrases of Figures 1 and 2 are examples of fragments identified as extractive. Using $\mathcal{F}(A, S)$, we compute two measures: *extractive fragment coverage* and *extractive fragment density*.

Extractive Fragment Coverage The coverage measure quantifies the extent to which a summary is derivative of a text. $\textsc{Coverage}(A, S)$ measures the percentage of words in the summary that are part of an extractive fragment with the article:

$$\textsc{Coverage}(A, S) = \frac{1}{|S|} \sum_{f \in \mathcal{F}(A,S)} |f| \ .$$

For example, a summary with 10 words that borrows 7 words from its article text and includes 3 new words will have $\textsc{Coverage}(A, S) = 0.7$.

Extractive Fragment Density The density measure quantifies how well the word sequence of a summary can be described as a series of extractions. For instance, a summary might contain many individual words from the article and therefore have a high coverage. However, if arranged in a new order, the words of the summary could still be used to convey ideas not present in the article. We define $\textsc{Density}(A, S)$ as the average length of the extractive fragment to which each word in the summary belongs. The density formulation is similar to the coverage definition but uses a square of the fragment length:

$$\textsc{Density}(A, S) = \frac{1}{|S|} \sum_{f \in \mathcal{F}(A,S)} |f|^2 \ .$$

For example, an article with a 10-word summary made of two extractive fragments of lengths 3 and 4 would have $\textsc{Coverage}(A, S) = 0.7$ and $\textsc{Density}(A, S) = 2.5$.

Compression Ratio We use a simple dimension of summarization, *compression ratio*, to further characterize summarization strategies. We define $\textsc{Compression}$ as the word ratio between the article and summary:

$$\textsc{Compression}(A, S) = |A| \ / \ |S| \ .$$

Summarizing with higher compression is challenging as it requires capturing more precisely the critical aspects of the article text.

4.2 Analysis of Dataset Diversity

We use density, coverage, and compression to understand the distribution of human summarization techniques across different sources. Figure 4 shows the distributions of summaries for different domains in the $\textsc{Newsroom}$ dataset, along with three major existing summarization datasets: DUC 2003-2004 (combined), CNN / Daily Mail, and the New York Times Corpus.

Publication Diversity Each $\textsc{Newsroom}$ publication shows a unique distribution of summaries mixing extractive and abstractive strategies in varying amounts. For example, the third entry on the top row shows the summarization strategy used by BuzzFeed. The density (y-axis) is relatively low, meaning BuzzFeed summaries are unlikely to include long extractive fragments. While the coverage (x-axis) is more varied, BuzzFeed's coverage tends to be lower, indicating that it frequently uses novel words in summaries. The publication plots in the figure are sorted by median compression ratio. We observe that publications with lower compression ratio (top-left of the figure) exhibit higher diversity along both dimensions of extractiveness. However, as the median compression ratio increases, the distributions become more con-

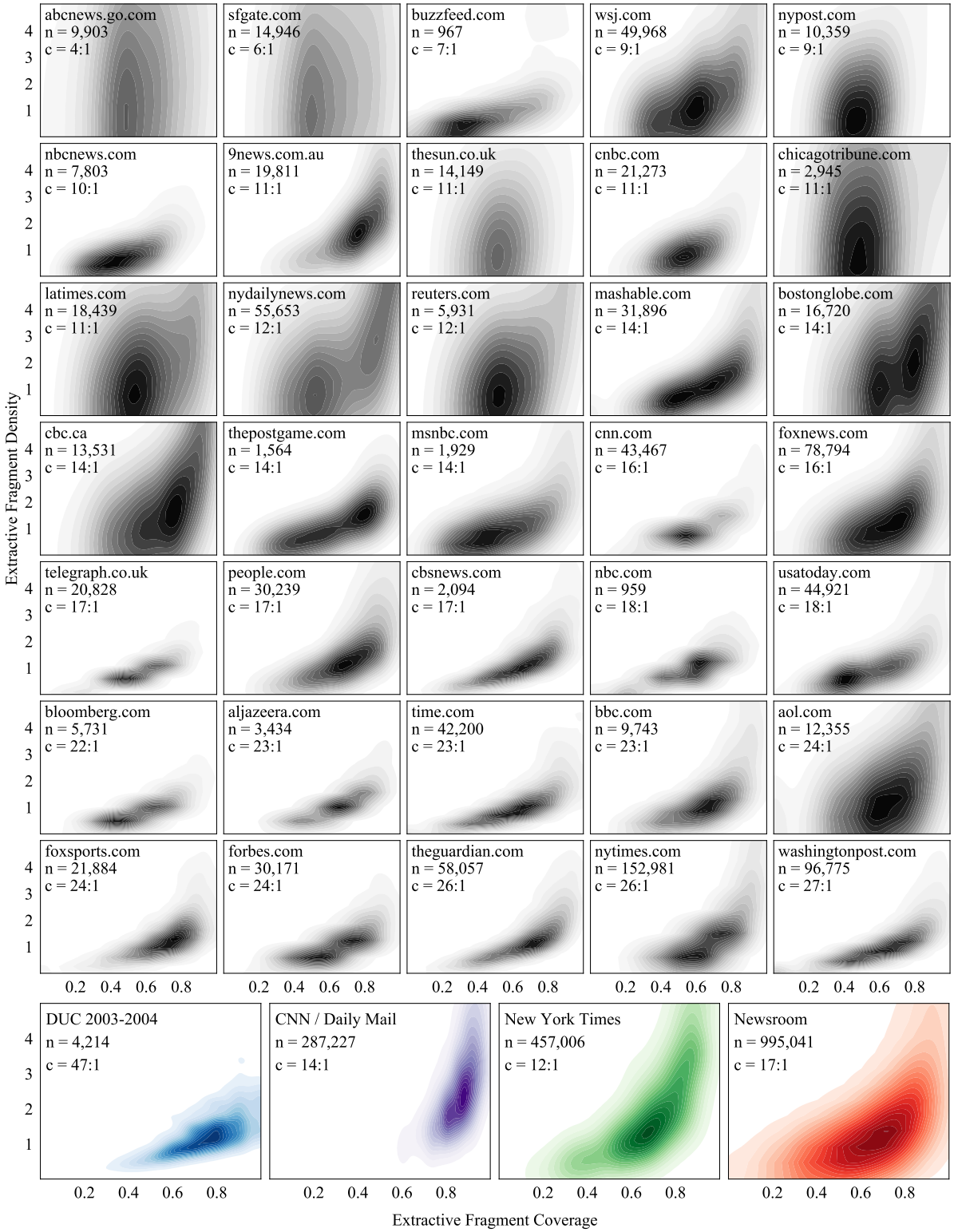


Figure 4: Density and coverage distributions across the different domains and existing datasets. NEWSROOM contains diverse summaries that exhibit a variety of summarization strategies. Each box is a normalized bivariate density plot of extractive fragment coverage (x-axis) and density (y-axis), the two measures of extraction described in Section 4.1. The top left corner of each plot shows the number of training set articles n and the median compression ratio c of the articles. For DUC and New York Times, which have no standard data splits, n is the total number of articles. *Above, top left to bottom right:* Plots for each publication in the NEWSROOM dataset. We omit TMZ, Economist, and ABC for presentation. *Below, left to right:* Plots for each summarization dataset showing increasing diversity of summaries along both dimensions of extraction in NEWSROOM.

centrated, indicating that summarization strategies become more rigid.

Dataset Diversity Figure 4 demonstrates how DUC, CNN / Daily Mail, and the New York Times exhibit different human summarization strategies. DUC summarization is fairly similar to the high-compression newsrooms shown in the lower publication plots in Figure 4. However, DUC's median compression ratio is much higher than all other datasets and NEWSROOM publications. The figure shows that CNN / Daily Mail and New York Times are skewed toward extractive summaries with lower compression ratios. CNN / Daily Mail shows higher coverage and density than all other datasets and publishers in our data. Compared to existing datasets, NEWSROOM covers a much larger range of summarization styles, ranging from both highly extractive to highly abstractive.

5 Performance of Existing Systems

We train and evaluate several summarization systems to understand the challenges of NEWSROOM and its usefulness for training systems. We evaluate three systems, each using a different summarization strategy with respect to extractiveness: fully extractive (TextRank), fully abstractive (Seq2Seq), and mixed (pointer-generator). We further study the performance of the pointer-generator model on NEWSROOM by training three systems using different dataset configurations. We compare these systems to two rule-based systems that provide baseline (Lede-3) and an extractive oracle (Fragments).

Extractive: TextRank TextRank is a sentence-level extractive summarization system. The system was originally developed by Mihalcea and Tarau (2004) and was later further developed and improved by Barrios et al. (2016). TextRank uses an unsupervised sentence-ranking approach similar to Google PageRank (Page et al., 1999). TextRank picks a sequence of sentences from a text for the summary up to a maximum allowable length. While this maximum length is typically preset by the user, in order to optimize ROUGE scoring, we tune this parameter to optimize ROUGE-1 F_1-score on the NEWSROOM training data. We experimented with values between 1–200, and found the optimal value to be 50 words. We use tuned TextRank of in Tables 2, 3, and in the supplementary material.

Abstractive: Seq2Seq / Attention Sequence-to-sequence models with attention (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2014) have been applied to various language tasks, including summarization (Chopra et al., 2016; Nallapati et al., 2016a). The process by which the model produces tokens is abstractive, as there is no explicit mechanism to copy tokens from the input text. We train a TensorFlow implementation[6] of the Rush et al. (2015) model using NEWSROOM.

Mixed: Pointer-Generator The pointer-generator model (See et al., 2017) uses abstractive token generation and extractive token copying using a pointer mechanism (Vinyals et al., 2015; Gülçehre et al., 2016), keeping track of extractions using coverage (Tu et al., 2016). We evaluate three instances of this model by varying the training data: (1) Pointer-C: trained on the CNN / Daily Mail dataset; (2) Pointer-N: trained on the NEWSROOM dataset; and (3) Pointer-S: trained on a random subset of NEWSROOM training data the same size as the CNN / Daily Mail training. The last instance aims to understand the effects of dataset size and summary diversity.

Lower Bound: Lede-3 A common automatic summarization strategy of online publications is to copy the first sentence, first paragraph, or first k words of the text and treat this as the summary. Following prior work (See et al., 2017; Nallapati et al., 2017), we use the Lede-3 baseline, in which the first three sentences of the text are returned as the summary. Though simple, this baseline is competitive with state-of-the-art systems.

Extractive Oracle: Fragments This system has access to the reference summary. Given an article A and its summary S, the system computes $\mathcal{F}(A, S)$ (Section 4). Fragments concatenates the fragments in $\mathcal{F}(A, S)$ in the order they appear in the summary, representing the best possible performance of an ideal extractive system. Only systems that are capable of abstractive reasoning can outperform the ROUGE scores of Fragments.

6 Automatic Evaluation

We study model performance of NEWSROOM, CNN / Daily Mail, and the combined DUC 2003 and 2004 datasets. We use the five systems described in Section 5, including the extractive oracle. We also evaluate the systems using subsets of

[6] https://github.com/tensorflow/models/tree/f87a58/research/textsum

	DUC 2003 & 2004			CNN / DAILY MAIL			NEWSROOM - T			NEWSROOM - U		
	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
Lede-3	12.99	3.89	11.44	38.64	17.12	35.13	30.49	21.27	28.42	30.63	21.41	28.57
Fragments	87.04	68.45	87.04	93.36	83.19	93.36	88.46	76.03	88.46	88.48	76.06	88.48
TextRank	15.75	4.06	13.02	29.06	11.14	24.57	22.77	9.79	18.98	22.76	9.80	18.97
Abs-N	2.44	0.04	2.37	5.07	0.16	4.80	5.88	0.39	5.32	5.90	0.43	5.36
Pointer-C	12.40	2.88	10.74	32.51	11.90	**28.95**	20.25	7.32	17.30	20.29	7.33	17.31
Pointer-S	15.10	4.55	12.42	**34.33**	**13.79**	28.42	24.50	12.60	20.33	24.48	12.52	20.30
Pointer-N	**17.29**	**5.01**	**14.53**	31.61	11.70	27.23	**26.02**	**13.25**	**22.43**	**26.04**	**13.24**	**22.45**

Table 2: ROUGE-1, ROUGE-2, and ROUGE-L scores for baselines and systems on two common existing datasets, the combined DUC 2003 & 2004 datasets and CNN / Daily Mail dataset, and the released (T) and unreleased (U) test sets of NEWSROOM. The best results for non-baseline systems in the lower parts of the table are in **bold**.

	EXTRACTIVE			MIXED			ABSTRACTIVE			NEWSROOM - D		
	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
Lede-3	53.05	49.01	52.37	25.15	12.88	22.08	13.69	2.42	11.24	30.72	21.53	28.65
Fragments	98.95	97.89	98.95	92.68	82.09	92.68	73.43	47.66	73.43	88.46	76.07	88.46
TextRank	32.43	19.68	28.68	22.30	7.87	17.75	13.54	1.88	10.46	22.82	9.85	19.02
Abs-N	6.08	0.21	5.42	5.67	0.15	5.08	6.21	1.07	5.68	5.98	0.48	5.39
Pointer-C	28.34	14.65	25.21	20.22	6.51	16.88	13.11	1.62	10.72	20.47	7.50	17.51
Pointer-S	37.29	26.56	33.34	23.71	10.59	18.79	13.89	2.22	10.34	24.83	12.94	20.66
Pointer-N	**39.11**	**27.95**	**36.17**	**25.48**	**11.04**	**21.06**	**14.66**	**2.26**	**11.44**	**26.27**	**13.55**	**22.72**

Table 3: Performance of the baselines and systems on the three extractiveness subsets of the NEWSROOM development set, and the overall scores of systems on the full development set (D). The best results for non-baseline systems in the lower parts of the table are in **bold**.

NEWSROOM to characterize the sensitivity of systems to different levels of extractiveness in reference summaries. We use the F_1-score variants of ROUGE-1, ROUGE-2, and ROUGE-L to account for different summary lengths. ROUGE scores are computed with the default configuration of the Lin (2004b) ROUGE v1.5.5 reference implementation. Input article text and reference summaries for all systems are tokenized using the Stanford CoreNLP tokenizer (Manning et al., 2014).

Table 2 shows results for summarization systems on DUC, CNN / Daily Mail, and NEWSROOM. In nearly all cases, the fully extractive Lede-3 baseline produces the most successful summaries, with the exception of the relatively extractive DUC. Among models, NEWSROOM-trained Pointer-N performs best on all datasets other than CNN / Daily Mail, an out-of-domain dataset. Pointer-C, which has access to only a limited subset of NEWSROOM, performs worse than Pointer-N on average. However, despite not being trained on CNN / Daily Mail, Pointer-S outperforms Pointer-C on its own data under ROUGE-N and is competitive under ROUGE-L. Finally, both Pointer-N and Pointer-S outperform other systems and baselines on DUC, whereas Pointer-C does not outperform Lede-3.

Table 3 shows development results on the NEWSROOM data for different level of extractiveness. Pointer-N outperforms the remaining models across all extractive subsets of NEWSROOM and, in the case of the abstractive subset, exceeds the performance of Lede-3. The success of Pointer-N and Pointer-S in generalizing and outperforming models on DUC and CNN / Daily Mail indicates the usefulness of NEWSROOM in generalizing to out-of-domain data. Similar subset analysis for our other two measures, coverage and compression, are included in the supplementary material.

7 Human Evaluation

ROUGE scores systems using frequencies of shared n-grams. Evaluating systems with ROUGE alone biases scoring against abstractive systems, which rely more on paraphrasing. To overcome this limitation, we provide human evaluation of the different systems on NEWSROOM. While human evaluation is still uncommon in summarization work, developing a benchmark dataset presents an opportunity for developing an accompanying protocol for human evaluation.

Our evaluation method is centered around three objectives: (1) distinguishing between syntactic and semantic summarization quality, (2) providing a reliable (consistent and replicable) measurement, and (3) allowing for portability such that the

DIMENSION	PROMPT
Informativeness	How well does the summary capture the key points of the article?
Relevance	Are the details provided by the summary consistent with details in the article?
Fluency	Are the individual sentences of the summary well-written and grammatical?
Coherence	Do phrases and sentences of the summary fit together and make sense collectively?

Table 4: The prompts given to Amazon Mechanical Turk crowdworkers for evaluating each summary.

	SEMANTIC		SYNTACTIC		
	INF	REL	FLU	COH	Avg.
Lede-3	3.98	4.13	4.13	4.08	4.08
Fragments	2.91	3.26	3.09	3.06	3.08
TextRank	3.61	3.92	**3.87**	**3.86**	**3.81**
Abs-N	2.09	2.35	2.66	2.50	2.40
Pointer-C	3.55	3.78	3.22	3.30	3.46
Pointer-S	**3.77**	**4.02**	3.56	3.56	3.73
Pointer-N	3.36	3.82	3.43	3.39	3.50

Table 5: Average performance of systems as scored by human evaluators. Each summary was scored by three different evaluators. *Dimensions, from left to right:* informativeness, relevance, fluency, and coherence, and a mean of the four dimensions for each system.

measure can be applied to other models or summarization datasets.

We select two semantic and two syntactic dimensions for evaluation based on experiments with evaluation tasks by Paulus et al. (2017) and Tan et al. (2017). The two semantic dimensions, summary *informativeness* (INF) and *relevance* (REL), measure whether the system-generated text is useful as a summary, and appropriate for the source text, respectively. The two syntactic dimensions, *fluency* (FLU) and *coherence* (COH), measure whether individual sentences or phrases of the summary are well-written and whether the summary as a whole makes sense respectively. Evaluation was performed on 60 summaries, 20 from each extractive NEWSROOM subset. Each system-article pair was evaluated by three unique raters. Exact prompts given to raters for each dimension are shown in Table 4.

Table 5 shows the mean score given to each system under each of the four dimensions, as well as the mean overall score (rightmost column). No summarization system exceeded the scores given to the Lede-3 baseline. However, the extractive oracle designed to maximize n-gram based evaluation performed worse than the majority of systems under human evaluation. While the fully abstractive Abs-N model performed very poorly under automatic evaluation, it fared slightly better when scored by humans. TextRank received the highest overall score. TextRank generates full sentences extracted from the article, and raters preferred TextRank primarily for its fluency and coherence. The pointer-generator models do not have this advantage, and raters did not find the pointer-generator models to be as syntactically sound as TextRank. However, raters preferred the informativeness and relevance of the Pointer-S and Pointer-N models, though not the Pointer-C model, over TextRank.

8 Conclusion

We present NEWSROOM, a dataset of articles and their summaries written in the newsrooms of online publications. NEWSROOM is the largest summarization dataset available to date, and exhibits a wide variety of human summarization strategies. Our proposed measures and the analysis of strategies used by different publications and articles propose new directions for evaluating the difficulty of summarization tasks and for developing future summarization models. We show that the dataset's diversity of summaries presents a new challenge to summarization systems. Finally, we find that using NEWSROOM to train an existing state-of-art mixed-strategy summarization model results in performance improvements on out-of-domain data. The NEWSROOM dataset is available online at summari.es.

Acknowledgements

This work is funded by Oath as part of the Connected Experiences Laboratory and by a Google Research Award. We thank the anonymous reviewers for their feedback.

References

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

Federico Barrios, Federico López, Luis Argerich, and Rosa Wachenchauzer. 2016. Variations of the similarity function of textrank for automated summarization. *CoRR* abs/1602.03606. http://arxiv.org/abs/1602.03606.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. http://aclweb.org/anthology/P/P16/P16-1223.pdf.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. http://aclweb.org/anthology/P/P16/P16-1046.pdf.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. https://doi.org/10.3115/v1/D14-1179.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*. pages 93–98. http://aclweb.org/anthology/N/N16/N16-1012.pdf.

Hoa Trang Dang. 2006. Duc 2005: Evaluation of question-focused summarization systems. In *Proceedings of the Workshop on Task-Focused Summarization and Question Answering*. Association for Computational Linguistics, Stroudsburg, PA, USA, SumQA '06, pages 48–55. http://dl.acm.org/citation.cfm?id=1654679.1654689.

Bonnie Dorr, David Zajic, and Richard Schwartz. 2003. Hedge trimmer: A parse-and-trim approach to headline generation. In *Proceedings of the HLT-NAACL 03 on Text Summarization Workshop - Volume 5*. Association for Computational Linguistics, Stroudsburg, PA, USA, HLT-NAACL-DUC '03, pages 1–8. https://doi.org/10.3115/1119467.1119468.

Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. 2016. Learning-based single-document summarization with compression and anaphoricity constraints. The Association for Computer Linguistics. http://www.aclweb.org/anthology/P16-1188.

Katja Filippova, Enrique Alfonseca, Carlos A. Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with lstms. In Llus Mrquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton, editors, *EMNLP*. The Association for Computational Linguistics, pages 360–368. http://aclweb.org/anthology/D/D15/D15-1042.pdf.

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 1481–1491. http://aclweb.org/anthology/D/D13/D13-1155.pdf.

Çaglar Gülçehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words http://aclweb.org/anthology/P/P16/P16-1014.pdf.

Donna Harman and Paul Over. 2004. The effects of human variation in duc summarization evaluation. http://www.aclweb.org/anthology/W04-1003.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. MIT Press, Cambridge, MA, USA, NIPS'15, pages 1693–1701. http://dl.acm.org/citation.cfm?id=2969239.2969428.

Kai Hong and Ani Nenkova. 2014. Improving the estimation of word importance for news multi-document summarization. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2014, April 26-30, 2014, Gothenburg, Sweden*. pages 712–721. http://aclweb.org/anthology/E/E14/E14-1075.pdf.

C. Y. Lin. 2004a. Looking for a few good metrics: Automatic summarization evaluation - how many samples are enough? In *Proceedings of the NTCIR Workshop 4*.

Chin-Yew Lin. 2004b. Rouge: A package for automatic evaluation of summaries. In *Proc. ACL workshop on Text Summarization Branches Out*. page 10. http://research.microsoft.com/~cyl/download/papers/WAS2004.pdf.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*. pages 55–60. http://www.aclweb.org/anthology/P/P14/P14-5010.

R. Mihalcea and P. Tarau. 2004. TextRank: Bringing order into texts. In *Proceedings of EMNLP-04and the 2004 Conference on Empirical Methods in Natural Language Processing*. http://www.aclweb.org/anthology/W04-3252.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the Thirty-First AAAI*

Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.. pages 3075–3081. http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14636.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016a. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*. pages 280–290. http://aclweb.org/anthology/K/K16/K16-1028.pdf.

Ramesh Nallapati, Bowen Zhou, and Mingbo Ma. 2016b. Classify or select: Neural architectures for extractive document summarization. *CoRR* abs/1611.04244. http://arxiv.org/abs/1611.04244.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*. Association for Computational Linguistics, Stroudsburg, PA, USA, AKBC-WEKEX '12, pages 95–100. http://dl.acm.org/citation.cfm?id=2391200.2391218.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab. Previous number = SIDL-WP-1999-0120. http://ilpubs.stanford.edu:8090/422/.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *CoRR* abs/1705.04304. http://arxiv.org/abs/1705.04304.

Matt Peters. 2015. Benchmarking python content extraction algorithms: Dragnet, readability, goose, and eatiht. https://moz.com/devblog/benchmarking-python-content-extraction-/algorithms-dragnet-readability-goose-and-eatiht/.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. pages 379–389. http://aclweb.org/anthology/D/D15/D15-1044.pdf.

E. Sandhaus. 2008. The New York Times Annotated Corpus. *Linguistic Data Consortium, Philadelphia* 6(12).

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. pages 1073–1083. https://doi.org/10.18653/v1/P17-1099.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Neural Information Processing Systems*.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *ACL*. http://www.aclweb.org/anthology/P17-1108.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In *LREC*. volume 2012, pages 2214–2218.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 76–85. https://doi.org/10.18653/v1/P16-1008.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., pages 2692–2700. http://papers.nips.cc/paper/5866-pointer-networks.pdf.

Additional Evaluation

In Section 4, we discuss three measures of summarization diversity: coverage, density, and compression. In addition to quantifying diversity of summarization strategies, these measures are helpful for system error analysis. We use the density measurement to understand how system performance varies when compared against references using different extractive strategies by subdividing NEWSROOM into three subsets by extractiveness and evaluating using ROUGE on each. We show here a similar analysis using the remaining two measures, coverage and compression. Results for subsets based on coverage and compression are shown in Tables 6 and 7.

	LOW COVERAGE			MEDIUM			HIGH COVERAGE		
	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
Lede-3	15.07	4.02	12.66	29.66	18.69	26.98	46.89	41.25	45.77
Fragments	72.45	46.16	72.45	93.41	83.08	93.41	99.13	98.16	99.13
TextRank	14.43	2.80	11.36	23.62	9.48	19.27	30.15	17.04	26.18
Abs-N	6.25	1.09	5.72	5.61	0.15	5.05	6.10	0.19	5.40
Pointer-C	13.99	2.46	11.57	21.70	8.06	18.47	25.80	12.06	22.57
Pointer-S	15.16	3.63	11.61	26.95	14.51	22.30	32.42	20.77	28.15
Pointer-N	**16.07**	**3.78**	**12.85**	**28.79**	**15.31**	**24.79**	**34.03**	**21.67**	**30.62**

Table 6: Performance of the baselines and systems on the three coverage subsets of the NEWSROOM development set. Article-summary pairs with low coverage have reference summaries that borrow words less frequently from their texts and contain more novel words and phrases. Article-summary pairs with high coverage borrow more words from their text and include fewer novel words and phrases.

	LOW COMPRESSION			MEDIUM			HIGH COMPRESSION		
	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
Lede-3	42.89	34.91	41.06	30.62	20.77	28.30	18.57	8.83	16.53
Fragments	87.78	77.20	87.78	89.73	77.66	89.73	87.88	73.34	87.88
TextRank	30.35	17.51	26.67	22.98	8.69	18.56	15.07	3.31	11.78
Abs-N	6.27	0.75	5.65	6.22	0.52	5.60	5.48	0.18	4.93
Pointer-C	27.47	13.49	24.18	20.05	6.25	16.76	14.07	2.89	11.76
Pointer-S	35.42	23.43	30.89	24.11	11.28	19.45	15.31	4.46	11.98
Pointer-N	**36.96**	**24.52**	**33.43**	**25.56**	**11.68**	**21.47**	**16.57**	**4.72**	**13.52**

Table 7: Performance of the baselines and systems on the three compression subsets of the NEWSROOM development set. Article-summary pairs with low compression have longer reference summaries with respect to their texts. Article-summary pairs with high compression have shorter reference summaries with respect to their texts.

Polyglot Semantic Parsing in APIs

Kyle Richardson[†], **Jonathan Berant**[‡], **Jonas Kuhn**[†]
[†]Institute for Natural Language Processing, University of Stuttgart, Germany
{kyle,jonas}@ims.uni-stuttgart.de
[‡]Tel-Aviv University, Israel
joberant@cs.tau.ac.il

Abstract

Traditional approaches to semantic parsing (SP) work by training individual models for each available parallel dataset of text-meaning pairs. In this paper, we explore the idea of polyglot semantic translation, or learning semantic parsing models that are trained on multiple datasets and natural languages. In particular, we focus on translating text to code signature representations using the software component datasets of Richardson and Kuhn (2017a,b). The advantage of such models is that they can be used for parsing a wide variety of input natural languages and output programming languages, or mixed input languages, using a single unified model. To facilitate modeling of this type, we develop a novel graph-based decoding framework that achieves state-of-the-art performance on the above datasets, and apply this method to two other benchmark SP tasks.

1 Introduction

Recent work by Richardson and Kuhn (2017a,b); Miceli Barone and Sennrich (2017) considers the problem of translating source code documentation to lower-level code template representations as part of an effort to model the meaning of such documentation. Example documentation for a number of programming languages is shown in Figure 1, where each *docstring* description in red describes a given function (blue) in the library. While capturing the semantics of docstrings is in general a difficult task, learning the translation from descriptions to formal code representations (e.g., formal representations of functions) is proposed as a reasonable first step towards learning more general natural language understanding models in the software domain. Under this approach, one can view a software library, or API, as a kind of parallel translation corpus for studying *text* → `code` or `code` → *text* translation.

1. (*en*, `Java`) Documentation

```
*Returns the greater of two long values
public static long max(long a, long b)
```

2. (*en*, `Python`) Documentation

```
max(self, a, b):
    """Compares two values numerically
    and returns the maximum"""
```

3. (*en*, `Haskell`) Documentation

```
--| "The largest element of a non-empty structure"
maximum :: forall z. Ord a a => t a -> a
```

4. (*de*, `PHP`) Documentation

```
*gibt den größeren dieser Werte zurück.
max (mixed $value1, mixed $value2)
```

Figure 1: Example source code documentation.

Richardson and Kuhn (2017b) extracted the standard library documentation for 10 popular programming languages across a number of natural languages to study the problem of text to function signature translation. Initially, these datasets were proposed as a resource for studying semantic parser induction (Mooney, 2007), or for building models that learn to translate text to formal meaning representations from parallel data. In follow-up work (Richardson and Kuhn, 2017a), they proposed using the resulting models to do automated question-answering (QA) and code retrieval on target APIs, and experimented with an additional set of software datasets built from 27 open-source Python projects.

As traditionally done in SP (Zettlemoyer and Collins, 2012), their approach involves learning individual models for each parallel dataset or language pair, e.g., (*en*, `Java`), (*de*, `PHP`), and (*en*, `Haskell`). Looking again at Figure 1, we notice that while programming languages differ in terms of representation conventions, there is often overlap between the functionality implemented and naming in these different languages (e.g., the `max`

Proceedings of NAACL-HLT 2018, pages 720–730
New Orleans, Louisiana, June 1 - 6, 2018.

function), and redundancy in the associated linguistic descriptions. In addition, each English description (Figure 1.1-1.3) describes `max` differently using the synonyms *greater, maximum, largest*. In this case, it would seem that training models on multiple datasets, as opposed to single language pairs, might make learning more robust, and help to capture various linguistic alternatives.

With the software QA application in mind, an additional limitation is that their approach does not allow one to freely translate a given description to multiple output languages, which would be useful for comparing how different programming languages represent the same functionality. The model also cannot translate between natural languages and programming languages that are not observed during training. While software documentation is easy to find in bulk, if a particular API is not already documented in a language other than English (e.g., `Haskell` in *de*), it is unlikely that such a translation will appear without considerable effort by experienced translators. Similarly, many individual APIs may be too small or poorly documented to build individual models or QA applications, and will in some way need to bootstrap off of more general models or resources.

To deal with these issues, we aim to learn more general text-to-code translation models that are trained on multiple datasets simultaneously. Our ultimate goal is to build *polyglot* translation models (cf. Johnson et al. (2016)), or models with shared representations that can translate any input text to any output programming language, regardless of whether such language pairs were encountered explicitly during training. Inherent in this task is the challenge of building an efficient polyglot decoder, or a translation mechanism that allows such crossing between input and output languages. A key challenge is ensuring that such a decoder generates well-formed code representations, which is not guaranteed when one simply applies standard decoding strategies from SMT and neural MT (cf. Cheng et al. (2017)). Given our ultimate interest in API QA, such a decoder must also facilitate monolingual translation, or being able to translate to specific output languages as needed.

To solve the decoding problem, we introduce a new graph-based decoding and representation framework that reduces to solving shortest path problems in directed graphs. We investigate several translation models that work within this framework, including traditional SMT models and models based on neural networks, and report state-of-the-art results on the technical documentation task of Richardson and Kuhn (2017b,a). To show the applicability of our approach to more conventional SP tasks, we apply our methods to the GeoQuery domain (Zelle and Mooney, 1996) and the Sportscaster corpus (Chen et al., 2010). These experiments also provide insight into the main technical documentation task and highlight the strengths and weaknesses of the various translation models being investigated.

2 Related Work

Our approach builds on the baseline models introduced in Richardson and Kuhn (2017b) (see also Deng and Chrupała (2014)). Their work is positioned within the broader SP literature, where traditionally SMT (Wong and Mooney, 2006a) and parsing (Zettlemoyer and Collins, 2009) methods are used to study the problem of translating text to formal meaning representations, usually centering around QA applications (Berant et al., 2013). More recently, there has been interest in using neural network approaches either in place of (Dong and Lapata, 2016; Kočiský et al., 2016) or in combination with (Misra and Artzi, 2016; Jia and Liang, 2016; Cheng et al., 2017) these traditional models, the latter idea we look at in this paper.

Work in NLP on software documentation has accelerated in recent years due in large part to the availability of new data resources through websites such as StackOverflow and Github (cf. Allamanis et al. (2017)). Most of this recent work focuses on processing large amounts of API data in bulk (Gu et al., 2016; Miceli Barone and Sennrich, 2017), either for learning longer executable programs from text (Yin and Neubig, 2017; Rabinovich et al., 2017), or solving the inverse problem of code to text generation (Iyer et al., 2016; Richardson et al., 2017). In contrast to our work, these studies do not look explicitly at translating to target APIs, or at non-English documentation.

The idea of polyglot modeling has gained some traction in recent years for a variety of problems (Tsvetkov et al., 2016) and has appeared within work in SP under the heading of *multilingual SP* (Jie and Lu, 2014; Duong et al., 2017). A related topic is learning from multiple knowledge sources or domains (Herzig and Berant, 2017), which is related to our idea of learning from multiple APIs.

When building models that can translate between unobserved language pairs, we use the term *zero-shot translation* from Johnson et al. (2016).

3 Baseline Semantic Translator

Problem Formulation Throughout the paper, we refer to target code representations as API *components*. In all cases, components will consist of formal representations of functions, or function signatures (e.g., `long max(int a, int b)`), which include a function name (`max`), a sequence of arguments (`int a, int b`), and other information such as a return value (`long`) and namespace (for more details, see Richardson (2018)). For a given API dataset $D = \{(\mathbf{x}_i, \mathbf{z}_i)\}_{i=1}^{n}$ of size n, the goal is to learn a model that can generate *exactly* a correct component sequence $\mathbf{z} = (z_1, .., z_{|\mathbf{z}|})$, within a finite space $\mathcal{C}$ of signatures (i.e., the space of all defined functions), for each input text sequence $\mathbf{x} = (x_1, ..., x_{|\mathbf{x}|})$. This involves learning a probability distribution $p(\mathbf{z} \mid \mathbf{x})$. As such, one can think of this underlying problem as a *constrained* MT task.

In this section, we describe the baseline approach of Richardson and Kuhn (2017b). Technically, their approach has two components: a simple word-based translation model and task specific decoder, which is used to generate a k-best list of candidate component representations for a given input $\mathbf{x}$. They then use a discriminative model to rerank the translation output using additional non-world level features. The goal in this section is to provide the technical details of their translation approach, which we improve in Section 4.

3.1 Word-based Translation Model

The translation models investigated in Richardson and Kuhn (2017b) use a noisy-channel formulation where $p(\mathbf{z} \mid \mathbf{x}) \propto p(\mathbf{x} \mid \mathbf{z})p(\mathbf{z})$ via Bayes rule. By assuming a uniform prior on output components, $p(\mathbf{z})$, the model therefore involves estimating $p(\mathbf{x} \mid \mathbf{z})$, which under a word-translation model is computed using the following formula: $p(\mathbf{x} \mid \mathbf{z}) = \sum_{a \in \mathcal{A}} p(\mathbf{x}, a \mid \mathbf{z})$, where the summation ranges over the set of all many-to-one word alignments $\mathcal{A}$ from $\mathbf{x} \rightarrow \mathbf{z}$, with $|\mathcal{A}|$ equal to $(|\mathbf{z}| + 1)^{|\mathbf{x}|}$. They investigate various types of sequence-based alignment models (Och and Ney, 2003), and find that the classic IBM Model 1 outperforms more complex word models. This model factors in the following way and assumes an *independent word generation* process:

$$p(\mathbf{x} \mid \mathbf{z}) = \frac{1}{|\mathcal{A}|} \prod_{j=1}^{|\mathbf{x}|} \sum_{i=0}^{|\mathbf{z}|} p_t(x_j \mid z_i) \qquad (1)$$

where each p_t defines a multinomial distribution over a given component term z for all words x.

The decoding problem for the above translation model involves finding the most likely output $\hat{z}$, which requires solving an $\arg\max_{\mathbf{z}}$ over Equation 1. In the general case, this problem is known to be $\mathcal{NP}$-complete for the models under consideration (Knight, 1999) largely due to the large space of possible predictions $\mathbf{z}$. Richardson and Kuhn (2017b) avoid these issues by exploiting the finiteness of the target component search space (an idea we also pursue here and discuss more below), and describe a constrained decoding algorithm that runs in time $O(|\mathcal{C}| \log |\mathcal{C}|)$. While this works well for small APIs, it becomes less feasible when dealing with large sets of APIs, as in the polyglot case, or with more complex semantic languages typically used in SP (Liang, 2013).

4 Shortest Path Framework

To improve the baseline translation approach used previously (Section 3.1), we pursue a graph based approach. Given the formulation above and the finiteness of our prediction space $\mathcal{C}$, our approach exploits the fact that we can represent the complete component search space for any set of APIs as a directed acyclic finite-state automaton (`DAFSA`), such as the one shown graphically in Figure 2. The underlying graph is constructed by concatenating all of the component representations for each API of interest and applying standard finite-state construction and minimization techniques (Mohri, 1996). Each path in the resulting compact automaton is therefore a well-formed component representation.

Using an idea from Johnson et al. (2016), we add to each component representation an *artificial* token that identifies the output programming language or library. For example, the two edges from the initial state 0 in Figure 2 are labeled as *2C* and *2Clojure*, which identify the C and Clojure programming languages respectively. All paths starting from the right of these edges are therefore valid paths in each respective programming language. The paths starting from the initial state 0, in contrast, correspond to all valid component representations in all languages.

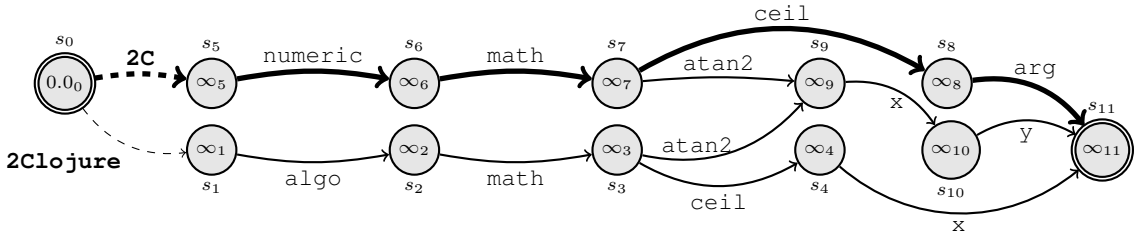


Figure 2: A DAFSA representation for a portion of the component sequence search space $\mathcal{C}$ that includes math functions in **C** and **Clojure**, and an example path/translation (in **bold**): **2C** `numeric math ceil arg`.

Decoding reduces to the problem of finding a path for a given text input **x**. For example, given the input *the ceiling of a number*, we would want to find the paths corresponding to the component translations `numeric math ceil arg` (in C) and `algo math ceil x` (in Clojure) in the graph shown in Figure 2. Using the trick above, our setup facilitates both monolingual decoding, i.e., generating components specific to a particular output language (e.g., the C language via the path shown in bold), and polyglot decoding, i.e., generating any output language by starting at the initial state 0 (e.g., C and Clojure).

We formulate the decoding problem using a variant of the well-known single source shortest path (SSSP) algorithm for directed acyclic graphs (DAGs) (Johnson (1977)). This involves a graph $\mathcal{G} = (V, E)$ (nodes V and labeled edges E, see graph in Figure 2), and taking an off-line topological sort of the graph's vertices. Using a data structure $d \in \mathbb{R}^{|V|}$ (initialized as $\infty^{|V|}$, as shown in Figure 2), the standard SSSP algorithm (which is the *forward update* variant of the Viterbi algorithm (Huang, 2008)) works by searching forward through the graph in sorted order and finding for each node v an incoming labeled edge u, with label z, that solves the following recurrence:

$$d(v) = \min_{(u,z):(u,v,z)\in E} \Big\{ d(u) + w(u, v, z) \Big\} \quad (2)$$

where $d(u)$ is shortest path score from a unique source node b to the incoming node u (computed recursively) and $w(u, v, z)$ is the weight of the particular labeled edge. The weight of the resulting shortest path is commonly taken to be the sum of the path edge weights as given by w, and the output translation is the sequence of labels associated with each edge. This algorithm runs in linear time over the size of the graph's adjacency matrix (`Adj`) and can be extended to find k SSSPs. In the standard case, a weighting function w is provided by assuming a static weighted graph. In our translation context, we replace w with a translation model, which is used to dynamically generate edge weights during the SSSP search for each input **x** by scoring the translation between **x** and each edge label z encountered.

Algorithm 1 Lexical Shortest Path Search

Input: Input **x** of size n, DAG $\mathcal{G} = (V, E)$, lexical translation function p_t, source node b with initial score o.
Output: Shortest component path

$d[V[\mathcal{G}]] \leftarrow \infty, \pi[V[\mathcal{G}]] \leftarrow Nil, d[b] \leftarrow o$
$s[V[\mathcal{G}], n] \leftarrow 0.0$ ▷ Shortest path sums at each node
for each vertex $u \geq b \in V[\mathcal{G}]$ in sorted order **do**
  for each vertex and label $(v, z) \in \mathtt{Adj}[u]$ **do**
    $\mathtt{score} \leftarrow -log\big[\ \prod_i^n p_t(x_i \mid z) + s[u, i]\ \big]$
    if $d[v] > \mathtt{score}$ **then**
      $d[v] \leftarrow \mathtt{score}, \pi[v] \leftarrow u$
      for i in $1, .., n$ **do** ▷ Update scores
        $s[v, i] \leftarrow p_t(x_i \mid z) + s[u, i]$
return FINDPATH$(\pi, |V|, b)$

Given this general framework, many different translation models can be used for scoring. In what follows, we describe two types of decoders based on lexical translation (or unigram) and neural sequence models. Technically, each decoding algorithm involves modifying the standard SSSP search procedure by adding an additional data structure s to each node (see Figure 2), which is used to store information about translations (e.g., running lexical translation scores, RNN state information) associated with particular shortest paths. By using these two very different models, we can get insight into the challenges associated with the technical documentation translation task. As we show in Section 6, each model achieves varying levels of success when subjected to a wider range of SP tasks, which reveals differences between our task and other SP tasks.

4.1 Lexical Translation Shortest Path

In our first model, we use the lexical translation model and probability function p_t in Equation 1 as

the weighting function, which can be learned efficiently off-line using the EM algorithm. When attempting to use the SSSP procedure to compute this equation for a given source input $\mathbf{x}$, we immediately have the problem that such a computation requires a complete component representation $\mathbf{z}$ (Knight and Al-Onaizan, 1998). We use an approximation[1] that involves ignoring the normalizer $|\mathcal{A}|$ and exploiting the word independence assumption of the model, which allows us to incrementally compute translation scores for individual source words given output translations corresponding to shortest paths during the SSSP search.

The full decoding algorithm in shown in Algorithm 1, where the red highlights the adjustments made to the standard SSSP search as presented in Cormen et al. (2009). The main modification involves adding a data structure $s \in \mathbb{R}^{|V| \times |\mathbf{x}|}$ (initialized as $0.0^{|V| \times |\mathbf{x}|}$ at line 2) that stores a running sum of source word scores given the best translations at each node, which can be used for computing the inner sum in Equation 1. For example, given an input utterance *ceiling function*, s_6 in Figure 2 contains the *independent* translation scores for words *ceiling* and *function* given the edge label `numeric` and p_t. Later on in the search, these scores are used to compute s_7, which will provide translation scores for each word given the edge sequence *numeric math*. Taking the product over any given s_j (as done in line 7 to get `score`) will give the probability of the shortest path translation at the particular point j. Here, the transformation into $-\log$ space is used to find the *minimum* incoming path. Standardly, the data structure π can be used to retrieve the shortest path back to the source node b (done via the FINDPATH method).

4.2 Neural Shortest Path

Our second set of models use neural networks to compute the weighting function in Equation 2. We use an encoder-decoder model with global attention (Bahdanau et al., 2014; Luong et al., 2015), which has the following two components:

Encoder Model The first is an *encoder* network, which uses a bi-directional recurrent neural network architecture with LSTM units (Hochreiter and Schmidhuber, 1997) to compute a sequence of forward annotations or hidden states $(\overrightarrow{h}_1, ..., \overrightarrow{h}_{|\mathbf{x}|})$ and a sequence of backward hidden states $(\overleftarrow{h}_1, ..., \overleftarrow{h}_{|\mathbf{x}|})$ for the input sequence $(x_1, ..., x_{|\mathbf{x}|})$. Standardly, each word is then represented as the concatenation of its forward and backward states: $h_j = [\overrightarrow{h}_j, \overleftarrow{h}_j]$.

Decoder Model The second component is a *decoder* network, which directly computes the conditional distribution $p(\mathbf{z} \mid \mathbf{x})$ as follows:

$$p(\mathbf{z} \mid \mathbf{x}) = \sum_{i=1}^{|\mathbf{z}|} \log p_\Theta(z_i \mid z_{<i}, \mathbf{x}) \quad (3)$$

$$p_\Theta(z_i \mid z_{<i}, \mathbf{x}) \sim \texttt{softmax}(f(\Theta, z_{<i}, \mathbf{x})) \quad (4)$$

where f is a non-linear function that encodes information about the sequence $z_{<i}$ and the input $\mathbf{x}$ given the model parameters Θ. We can think of this model as an ordinary recurrent language model that is additionally conditioned on the input $\mathbf{x}$ using information from our encoder. We implement the function f in the following way:

$$f(\Theta, z_{<i}, \mathbf{x}) = \mathbf{W}_o \eta_i + \mathbf{b}_o \quad (5)$$

$$\eta_i = \texttt{MLP}(c_i, g_i) \quad (6)$$

$$g_i = \texttt{LSTM}_{dec}(g_{i-1}, \mathbf{E}^{out}_{z_{i-1}}, c_i) \quad (7)$$

where `MLP` is a multi-layer perceptron model with a single hidden layer, $\mathbf{E}^{out} \in \mathbb{R}^{|\Sigma_{dec}| \times e}$ is a randomly initialized embedding matrix, g_i is the decoder's hidden state at step i, and c_i is a context-vector that encodes information about the input $\mathbf{x}$ and the encoder annotations. Each context vector c_i in turn is a weighted sum of each annotation h_j against an attention vector $\alpha_{i,j}$, or $c_i = \sum_{j=1}^{|\mathbf{x}|} \alpha_{i,j} h_j$, which is jointly learned using an additional single layered multi-layer perceptron defined in the following way:

$$\alpha_{i,j} \propto \exp(e_{i,j}); \quad e_{i,j} = \texttt{MLP}(g_{i-1}, h_j) \quad (8)$$

Lexical Bias and Copying In contrast to standard MT tasks, we are dealing with a relatively low-resource setting where the sparseness of the target vocabulary is an issue. For this reason, we experimented with integrating lexical translation scores using a biasing technique from Arthur et al. (2016). Their method is based on the following computation for each token z_i:

$$\texttt{bias}_i = \begin{bmatrix} p_{t'}(z_1 \mid x_1) & \dots & p_{t'}(z_1 \mid x_{|\mathbf{x}|}) \\ \vdots & \ddots & \vdots \\ p_{t'}(z_{|\Sigma_{dec}|} \mid x_1) & \dots & p_{t'}(z_{|\Sigma_{dec}|} \mid x_{|\mathbf{x}|}) \end{bmatrix} \begin{bmatrix} \alpha_{i,1} \\ \vdots \\ \alpha_{i,|\mathbf{x}|} \end{bmatrix}$$

[1] Details about the approx. are provided as supp. material.

Algorithm 2 Neural Shortest Path Search

Input: Input $\mathbf{x}$, DAG $\mathcal{G}$, neural parameters Θ and non-linear function f, beam size l, source node b with init. score o.
Output: Shortest component path

1: $d[V[\mathcal{G}]] \leftarrow \infty, d[b] \leftarrow o, \pi[V[\mathcal{G}]] \leftarrow Nil$
2: $s[V[\mathcal{G}]] \leftarrow Nil$ ▷ Path state information
3: $s[b] \leftarrow \texttt{InitState()}$ ▷ Initialize source state
4: **for** each vertex $u \geq b \in V[\mathcal{G}]$ in sorted order **do**
5: **if** $\texttt{isinf}(d[u])$ **then** continue
6: $p \leftarrow s[u]$ ▷ Current state at node u, or $z_{<i}$
7: $L^1_{[l]} \leftarrow \underset{(v_1,\ldots,v_k)\in \texttt{Adj}[u]}{\arg\max} \texttt{softmax}(f(\Theta, p, \mathbf{x}))$
8: **for** each vertex and label $(v, z) \in L$ **do**
9: $\texttt{score} \leftarrow -\log p_\Theta(z \mid p, \mathbf{x}) + d[u]$
10: **if** $d[v] > \texttt{score}$ **then**
11: $d[v] \leftarrow \texttt{score}, \pi[v] \leftarrow u$
12: $s[v] \leftarrow \texttt{UpdateState}(p, z)$
13: **return** FINDPATH$(\pi, |V|, b)$

The first matrix uses the inverse ($p_{t'}$) of the lexical translation function p_t already introduced to compute the probability of each word in the target vocabulary Σ_{dec} (the columns) with each word in the input $\mathbf{x}$ (the rows), which is then weighted by the attention vector from Equation 8. $bias_i$ is then used to modify Equation 5 in the following way:

$$f_{bias}(\Theta, z_{<i}, \mathbf{x}) = \mathbf{W}_o \eta_i + \mathbf{b}_o + \log(\texttt{bias}_i + \epsilon)$$

where ϵ is a hyper-parameter that helps to preserve numerical stability and biases more heavily on the lexical model when set lower.

We also experiment with the *copying* mechanism from Jia and Liang (2016), which works by allowing the decoder to choose from a set of latent actions, a_j, that includes writing target words according to Equation 5, as done standardly, or copying source words from $\mathbf{x}$, or $\texttt{copy}[x_i]$ according to the attention scores in Equation 8. A distribution is then computed over these actions using a $\texttt{softmax}$ function and particular actions are chosen accordingly during training and decoding.

Decoding and Learning The full decoding procedure is shown in Algorithm 2, where the differences with the standard SSSP are again shown in red. We change the data structure s to contain the decoder's RNN state at each node. We also modify the scoring (line 7, which uses Equation 4) to consider only the top l edges or translations at that point, as opposed to imposing a full search. When l is set to 1, for example, the procedure does a greedy search through the graph, whereas when l is large the procedure is closer to a full search.

In general terms, the decoder described above works like an ordinary neural decoder with the difference that each decision (i.e., new target-side word translation) is constrained (in line 7) by the transitions allowed in the underlying graph in order to ensure wellformedness of each component output. Standardly, we optimize these models using stochastic gradient descent with the objective of finding parameters $\hat{\Theta}$ that minimize the negative conditional log-likelihood of the training dataset.

4.3 Monolingual vs. Polyglot Decoding

Our framework facilitates both monolingual and polyglot decoding. In the first case, the decoder requires a graph associated with the output semantic language (more details in next section) and a trained translation model. The latter case requires taking the union of all datasets and graphs (with artificial identifier tokens) for a collection of target datasets and training a single model over this global dataset. In this setting, we can then decode to a particular language using the language identifiers or decode without specifying the output language. The main focus in this paper is investigating polyglot decoding, and in particular the effect of training models on multiple datasets when translating to individuals APIs or SP datasets.

When evaluating our models and building QA applications, it is important to be able to generate the k best translations. This can easily be done in our framework by applying standard k SSSP algorithms (Brander and Sinclair, 1995). We use an implementation of the algorithm of Yen (1971), which works on top of the SSSP algorithms introduced above by iteratively finding deviating or branching paths from an initial SSSP (more details provided in supplementary materials).

5 Experiments

We experimented with two main types of resources: 45 API documentation datasets and two multilingual benchmark SP datasets. In the former case, our main objective is to test whether training polyglot models (shown as **polyglot** in Tables 1-2) on multiple datasets leads to an improvement when compared to training individual monolingual models (shown as **monolingual** in Tables 1-2). Experiments involving the latter datasets are meant to test the applicability of our general graph and polyglot method to related SP tasks, and are also used for comparison against our main technical documentation task.

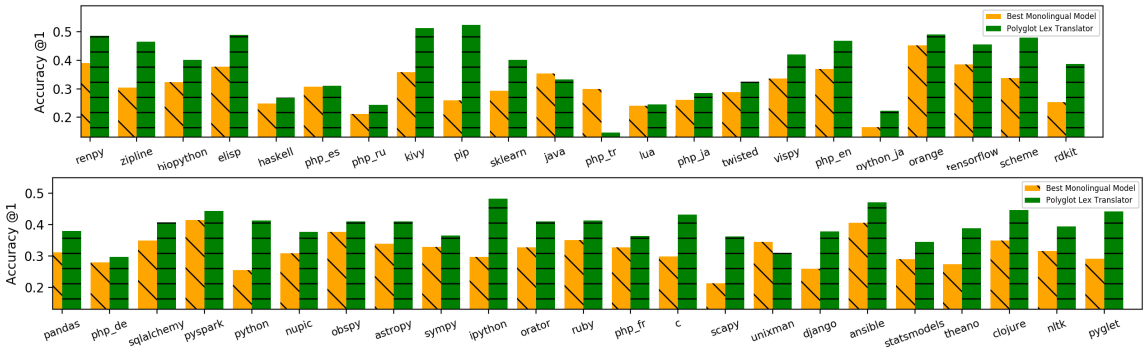


Figure 3: Test **Acc@1** for the best monolingual models (in yellow/left) compared with the best lexical polyglot model (green/right) across all 45 technical documentation datasets.

5.1 Datasets

Technical API Docs The first dataset includes the Stdlib and Py27 datasets of Richardson and Kuhn (2017b,a), which are publicly available via Richardson (2017). Stdlib consists of short description and function signature pairs for 10 programming languages in 7 languages, and Py27 contains the same type of data for 27 popular Python projects in English mined from Github. We also built new datasets from the Japanese translation of the Python 2.7 standard library, as well as the Lua stdlib documentation in a mixture of Russian, Portuguese, German, Spanish and English.

Taken together, these resources consist of 79,885 training pairs, and we experiment with training models on Stdlib and Py27 separately as well as together (shown as **+ more** in Table 1). We use a BPE subword encoding (Sennrich et al., 2015) of both input and output words to make the representations more similar and transliterated all datasets (excluding Japanese datasets) to an 8-bit latin encoding. Graphs were built by concatenating all function representations into a single word list and compiling this list into a minimized DAFSA. For our global polyglot dataset, this resulted in a graph with 218,505 nodes, 313,288 edges, and 112,107 paths or component representations over an output vocabulary of 9,324 words.

Mixed GeoQuery and Sportscaster We run experiments on the GeoQuery 880 corpus using the splits from Andreas et al. (2013), which includes geography queries for English, Greek, Thai, and German paired with formal database queries, as well as a seed lexicon or *NP list* for each language. In addition to training models on each individual dataset, we also learn polyglot models trained on all datasets concatenated together. We also created a new mixed language test set that was built by replacing NPs in 803 test examples with one or more NPs from a different language using the NP lists mentioned above (see examples in Figure 4). The goal in the last case is to test our model's ability to handle mixed language input. We also ran monolingual experiments on the English Sportscaster corpus, which contains human generated soccer commentary paired with symbolic meaning representation produced by a simulation of four games.

For GeoQuery graph construction, we built a single graph for all languages by extracting general rule templates from all representations in the dataset, and exploited additional information and patterns using the Geobase database and the semantic grammars used in (Wong and Mooney, 2006b). This resulted in a graph with 2,419 nodes, 4,936 edges and 39,482 paths over an output vocabulary of 164. For Sportscaster, we directly translated the semantic grammar provided in Chen and Mooney (2008) to a DAFSA, which resulted in a graph with 98 nodes, 86 edges and 830 paths.

5.2 Experimental Setup

For the technical datasets, the goal is to see if our model generates correct signature representations from unobserved descriptions using exact match. We follow exactly the experimental setup and data splits from Richardson and Kuhn (2017b), and measure the accuracy at 1 (**Acc@1**), accuracy in top 10 (**Acc@10**), and **MRR**.

For the GeoQuery and Sportscaster experiments, the goal is to see if our models can generate correct meaning representations for unseen input. For GeoQuery, we follow Andreas et al. (2013) in evaluating extrinsically by checking that each representation evaluates to the same answer as the gold representation when executed against the Geobase database. For Sportscaster, we evaluate by exact match to a gold representation.

		Method	Acc@1	Acc@10	MRR
stdlib	mono.	RK Trans + *rerank*	29.9	69.2	43.1
		Lexical SP	**33.2**	70.7	45.9
	poly.	Lexical SP + *more*	33.1	69.7	45.5
		Neural SP + *bias*	12.1	34.3	19.5
		Neural SP + *copy_bias*	13.9	36.5	21.5
py27	mono.	RK Trans + *rerank*	32.4	73.5	46.5
		Lexical SP	**41.3**	77.7	54.1
	poly.	Lexical SP + *more*	40.5	76.7	53.1
		Neural SP + *bias*	8.7	25.5	14.2
		Neural SP + *copy_bias*	9.0	26.9	15.1

Table 1: Test results on the Stdlib and Py27 tasks averaged over all datasets and compared against the best monolingual results from Richardson and Kuhn (2017b,a), or RK

5.3 Implementation and Model Details

We use the Foma finite-state toolkit of Hulden (2009) to construct all graphs used in our experiments. We also use the Cython version of Dynet (Neubig et al., 2017) to implement all the neural models (see supp. materials for more details).

In the results tables, we refer to the lexical and neural models introduced in Section 4 as *Lexical Shortest Path* and *Neural Shortest Path*, where models that use copying (+ *copy*) and lexical biasing (+ *bias*) are marked accordingly. We also experimented with adding a discriminative reranker to our lexical models (+ *rerank*), using the approach from Richardson and Kuhn (2017b), which uses additional lexical (e.g., word match and alignment) features and other phrase-level and syntax features. The goal here is to see if these additional (mostly non-word level) features help improve on the baseline lexical models.

6 Results and Discussion

Technical Documentation Results Table 1 shows the results for Stdlib and Py27. In the monolingual case, we compare against the best performing models in Richardson and Kuhn (2017b,a). As summarized in Figure 3, our experiments show that training polyglot models on multiple datasets can lead to large improvements over training individual models, especially on the Py27 datasets where using a polyglot model resulted in a nearly 9% average increase in accuracy @1. In both cases, however, the best performing lexical models are those trained only on the datasets they are evaluated on, as opposed to training on all datasets (i.e., + *more*). This is surprising given that training on all datasets doubles the size of the training data, and shows that adding more data does not necessarily boost performance when the additional data is from another distribution.

		Method	Acc@1	Acc@10
Standard Geoquery	monolingual	UBL (Kwiatkowski et al., 2010)	74.2	–
		TreeTrans (Jones et al., 2012)	76.8	–
		nHT (Susanto and Lu, 2017)	**83.3**	–
		Lexical Shortest Path	68.6	92.4
		Lexical Shortest Path + *rerank*	74.2	94.1
		Neural Shortest Path	73.5	91.1
		Neural Shortest Path + *bias*	78.0	92.8
		Neural Shortest Path + *copy_bias*	77.8	92.1
	polyglot	Lexical Shortest Path	67.3	92.9
		Lexical Shortest Path + *rerank*	75.2	94.7
		Neural Shortest Path	78.0	91.4
		Neural Shortest Path + *bias*	78.9	91.7
		Neural Shortest Path + *copy_bias*	**79.6**	91.9
		Best Monolingual Model	4.2	18.2
Mixed	poly.	Lexical Shortest Path + *rerank*	71.1	94.3
		Neural Shortest Path + *copy_bias*	**75.2**	90.0
Sportscaster	mono.	PCFG (Börschinger et al., 2011)	74.2	–
		wo-PCFG (Börschinger et al., 2011)	**86.0**	–
		Lexical Shortest Path	40.3	86.8
		Lexical Shortest Path + *rerank*	70.3	90.2
		Neural Shortest Path	81.9	94.8
		Neural Shortest Path + *bias*	83.4	93.9
		Neural Shortest Path + *copy_bias*	83.3	90.5

Table 2: Test results for the standard (above) and mixed (middle) GeoQuery tasks averaged over all languages, and results for the English Sportscaster task (below).

The neural models are strongly outperformed by all other models both in the monolingual and polyglot case (only the latter results shown), even when lexical biasing is applied. While surprising, this is consistent with other studies on *low-resource* neural MT (Zoph et al., 2016; Östling and Tiedemann, 2017), where datasets of comparable size to ours (e.g., 1 million tokens or less) typically fail against classical SMT models. This result has also been found in relation to neural AMR semantic parsing, where similar issues of sparsity are encountered (Peng et al., 2017). Even by doubling the amount of training data by training on all datasets (results not shown), this did not improve the accuracy, suggesting that much more data is needed (more discussion below).

Beyond increases in accuracy, our polyglot models support zero-shot translation as shown in Figure 4, which can be used for translating between unobserved language pairs (e.g., (*es*,`Clojure`), (*ru*,`Haskell`) as shown in 1-2), or for finding related functionality across different software projects (as shown in 3). These results were obtained by running our decoder model without specifying the output language. We note, however, that the decoder can be constrained to selectively translate to any specific programming language or project (e.g., in a QA setting). Future work will further investigate the decoder's polyglot capabilities, which is currently hard to evaluate since we do not have an annotated set of function equivalences between different APIs.

1.	Source API (stdlib): (***es***, **PHP**)	**Input**: Devuelve el mensaje asociado al objeto lanzado.
Output	Language: **PHP**	Function Translation: `public string Throwable::getMessage ( void )`
	Language: **Java**	Function Translation: `public String lang.getMessage( void )`
	Language: **Clojure**	Function Translation: `(tools.logging.fatal throwable message & more)`
2.	Source API (stdlib): (***ru***, **PHP**)	**Input**: конвертирует строку из формата UTF-32 в формат UTF-16.
Output	Language: **PHP**	Function Translation: `string PDF_utf32_to_utf16 ( ... )`
	Language: **Ruby**	Function Translation: `String#toutf16 => string`
	Language: **Haskell**	Function Translation: `Encoding.encodeUtf16LE :: Text -> ByteString`
3.	Source API (py): (***en***, **stats**)	**Input:** Compute the Moore-Penrose pseudo-inverse of a matrix.
Output	Project: **sympy**	Function Translation: `matrices.matrix.base.pinv_solve( B, ... )`
	Project: **sklearn**	Function Translation: `utils.pinvh( a, cond=None,rcond=None,... )`
	Project: **stats**	Function Translation: `tools.pinv2( a,cond=None,rcond=None )`
4.	Mixed GeoQuery (de/gr)	**Input**: Wie hoch liegt der höchstgelegene punkt in Αλαμπάμα?
	Logical Form Translation: `answer(elevation_1(highest(place(loc_2(stateid('alabama'))))))`	

Figure 4: Examples of zero-shot translation when running in polyglot mode (1-3, function representations shown in a conventionalized format), and mixed language parsing (4).

Semantic Parsing Results SP results are summarized in Table 2. In contrast, the neural models, especially those with biasing and copying, strongly outperform all other models and are competitive with related work. In the GeoQuery case, we compare against two classic grammar-based models, UBL and TreeTrans, as well as a feature rich, neural hybrid tree model (nHT). We also see that the polyglot Geo achieves the best performance, demonstrating that training on multiple datasets helps in this domain as well. In the Sportscaster case we compare against two PCFG learning approaches, where the second model (wo-PCFG) involves a grammar with complex word-order constraints.

The advantage of training a polyglot model is shown on the results related to mixed language parsing (i.e., the middle set of results). Here we compared against the best performing monolingual English model (**Best Mono. Model**), which does not have a way to deal with multilingual NPs. We also find the neural model to be more robust than the lexical models with reranking.

While the lexical models overall perform poorly on both tasks, the weakness of this model is particularly acute in the Sportscaster case. We found that mistakes are largely related to the ordering of arguments, which these lexical (unigram) models are blind to. That these models still perform reasonably well on the Geo task shows that such ordering issues are less of a factor in this domain.

Discussion Having results across related SP tasks allows us to reflect on the nature of the main technical documentation task. Consistent with recent findings (Dong and Lapata, 2016), we show that relatively simple neural sequence models are competitive with, and in some cases outperform, traditional grammar-based SP methods on benchmark SP tasks. However, this result is not observed in our technical documentation task, in part because this problem is much harder for neural learners given the sparseness of the target data and lack of redundancy. For this reason, we believe our datasets provide new challenges for neural-based SP, and serve as a cautionary tale about the scalability and applicability of commonly used neural models to lower-resource SP problems.

In general, we believe that focusing on polyglot and mixed language decoding is not only of interest to applications (e.g, mixed language API QA) but also allows for new forms of SP evaluation that are more revealing than only translation accuracy. When comparing the accuracy of the best monolingual Geo model and the worst performing neural polyglot model, one could mistakingly think that these models have equal abilities, though the polyglot model is much more robust and general. Moving forward, we hope that our work helps to motivate more diverse evaluations of this type.

7 Conclusion

We look at learning from multiple API libraries and datasets in the context of learning to translate text to code representations and other SP tasks. To support *polyglot* modeling of this type, we developed a novel graph based decoding method and experimented with various SMT and neural MT models that work in this framework. We report a mixture of positive results specific to each task and set of models, some of which reveal interesting limitations of different approaches to SP. We also introduced new API and mixed language datasets to facilitate further work on polyglot SP.

Acknowledgements

This work was supported by the German Research Foundation (DFG) in project D2 of SFB 732.

References

Miltiadis Allamanis, Earl T Barr, Premkumar Devanbu, and Charles Sutton. 2017. A Survey of Machine Learning for Big Code and Naturalness. *arXiv preprint arXiv:1709.06182* .

Jacob Andreas, Andreas Vlachos, and Stephen Clark. 2013. Semantic parsing as machine translation. In *in Proceedings of ACL-2013*. pages 47–52.

Philip Arthur, Graham Neubig, and Satoshi Nakamura. 2016. Incorporating Discrete Translation Lexicons into Neural Machine Translation. *arXiv preprint arXiv:1606.02006* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv preprint arXiv:1409.0473* .

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic Parsing on Freebase from Question-Answer Pairs. In *in Proceedings of EMNLP-2013*. pages 1533–1544.

Benjamin Börschinger, Bevan K. Jones, and Mark Johnson. 2011. Reducing grounded learning tasks to grammatical inference. In *Proceedings of EMNLP-2011*. pages 1416–1425.

AW Brander and MC Sinclair. 1995. A Comparative Study of k-Shortest Path Algorithms. In *In Proc. of 11th UK Performance Engineering Workshop*.

David L. Chen, Joohyun Kim, and Raymond J. Mooney. 2010. Training a Multilingual Sportscaster: Using Perceptual Context to Learn Language. *Journal of Artificial Intelligence Research* 37:397–435.

David L. Chen and Raymond J. Mooney. 2008. Learning to Sportscast: A Test of Grounded Language Acquisition. In *Proceedings of ICML-2008*. pages 128–135.

Jianpeng Cheng, Siva Reddy, Vijay Saraswat, and Mirella Lapata. 2017. Learning Structured Natural Language Representations for Semantic Parsing. *arXiv preprint arXiv:1704.08387* .

T Cormen, C Leiserson, R Rivest, and C Stein. 2009. *Introduction to Algorithms*. MIT Press.

Huijing Deng and Grzegorz Chrupała. 2014. Semantic Approaches to Software Component Retrieval with English Queries. In *Proceedings of LREC-14*. pages 441–450.

Li Dong and Mirella Lapata. 2016. Language to Logical Form with Neural Attention. *arXiv preprint arXiv:1601.01280* .

Long Duong, Hadi Afshar, Dominique Estival, Glen Pink, Philip Cohen, and Mark Johnson. 2017. Multilingual Semantic Parsing and Code-Switching. *CoNLL 2017* page 379.

Xiaodong Gu, Hongyu Zhang, Dongmei Zhang, and Sunghun Kim. 2016. Deep API Learning. *arXiv preprint arXiv:1605.08535* .

Jonathan Herzig and Jonathan Berant. 2017. Neural Semantic Parsing over Multiple Knowledge-Bases. *arXiv preprint arXiv:1702.01569* .

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural computation* 9(8).

Liang Huang. 2008. Advanced Dynamic Programming in Semiring and Hypergraph Frameworks. In *Proceedings of COLING-2008 (tutorial notes)*.

Mans Hulden. 2009. Foma: a Finite-State Compiler and Library. In *Proceedings of EACL*.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2016. Summarizing Source Code using a Neural Attention Model. In *Proceedings of ACL*.

Robin Jia and Percy Liang. 2016. Data Recombination for Neural Semantic Parsing. *arXiv preprint arXiv:1606.03622* .

Zhanming Jie and Wei Lu. 2014. Multilingual Semantic Parsing: Parsing Multiple Languages into Semantic Representations. In *COLING*. pages 1291–1301.

Donald B Johnson. 1977. Efficient Algorithms for Shortest Paths in Sparse Networks. *Journal of the ACM (JACM)* 24(1):1–13.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2016. Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. *arXiv preprint arXiv:1611.04558* .

Bevan Keeley Jones, Mark Johnson, and Sharon Goldwater. 2012. Semantic Parsing with Bayesian Tree Transducers. In *Proceedings of ACL-2012*. pages 488–496.

Kevin Knight. 1999. Decoding Complexity in Word-Replacement Translation Models. *Computational linguistics* 25(4):607–615.

Kevin Knight and Yaser Al-Onaizan. 1998. Translation with Finite-state Devices. In *Proceedings of AMTA*.

Tomáš Kočiský, Gábor Melis, Edward Grefenstette, Chris Dyer, Wang Ling, Phil Blunsom, and Karl Moritz Hermann. 2016. Semantic Parsing with Semi-Supervised Sequential Autoencoders. In *Proceedings of EMNLP-16*. pages 1078–1087.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing Probabilistic CCG Grammars from Logical Form with Higher-order Unification. In *Proceedings of EMNLP-2010*. pages 1223–1233.

Percy Liang. 2013. Lambda Dependency-Based Compositional Semantics. *arXiv preprint arXiv:1309.4408* .

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective Approaches to Attention-Based Neural Machine Translation. *arXiv preprint arXiv:1508.04025* .

Antonio Valerio Miceli Barone and Rico Sennrich. 2017. A Parallel Corpus of Python Functions and Documentation Strings for Automated Code Documentation and Code Generation. *arXiv preprint arXiv:1707.02275* .

Dipendra Kumar Misra and Yoav Artzi. 2016. Neural Shift-Reduce CCG Semantic Parsing. In *EMNLP*. pages 1775–1786.

Mehryar Mohri. 1996. On Some Applications of Finite-State Automata Theory to Natural Language Processing. *Natural Language Engineering* 2(1):61–80.

Raymond Mooney. 2007. Learning for Semantic Parsing. In *Proceedings of CICLING*.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The Dynamic Neural Network Toolkit. *arXiv preprint arXiv:1701.03980* https://github.com/clab/dynet.

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational linguistics* 29(1):19–51.

Robert Östling and Jörg Tiedemann. 2017. Neural Machine Translation for Low-resource Languages. *arXiv preprint arXiv:1708.05729* .

Xiaochang Peng, Chuan Wang, Daniel Gildea, and Nianwen Xue. 2017. Addressing the Data Sparsity Issue in Neural AMR Parsing. *Proceedings of ACL* .

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract Syntax Networks for Code Generation and Semantic Parsing. In *Proceedings of ACL*.

Kyle Richardson. 2017. Code-Datasets. https://github.com/yakazimir/Code-Datasets.

Kyle Richardson. 2018. A Language for Function Signature Representations. *arXiv preprint arXiv:1804.00987* .

Kyle Richardson and Jonas Kuhn. 2017a. Function Assistant: A Tool for NL Querying of APIs. In *Proceedings of EMNLP*.

Kyle Richardson and Jonas Kuhn. 2017b. Learning Semantic Correspondences in Technical Documentation. In *Proceedings of ACL*.

Kyle Richardson, Sina Zarrieß, and Jonas Kuhn. 2017. The Code2Text Challenge: Text Generation in Source Code Libraries. In *Proceedings of INLG*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural Machine Translation of Rare Words with Subword Units. *arXiv preprint arXiv:1508.07909* .

Raymond Hendy Susanto and Wei Lu. 2017. Semantic Parsing with Neural Hybrid Trees. In *AAAI*. pages 3309–3315.

Yulia Tsvetkov, Sunayana Sitaram, Manaal Faruqui, Guillaume Lample, Patrick Littell, David Mortensen, Alan W Black, Lori Levin, and Chris Dyer. 2016. Polyglot Neural Language Models: A Case Study in Cross-Lingual Phonetic Representation Learning. *arXiv preprint arXiv:1605.03832* .

Yuk Wah Wong and Raymond J. Mooney. 2006a. Learning for semantic parsing with statistical machine translation. In *Proceedings of HLT-NAACL-2006*. pages 439–446.

Yuk Wah Wong and Raymond J Mooney. 2006b. Learning for Semantic Parsing with Statistical Machine Translation. In *Proceedings of NACL*.

Jin Y Yen. 1971. Finding the k Shortest Loopless Paths in a Network. *Management Science* 17(11):712–716.

Pengcheng Yin and Graham Neubig. 2017. A Syntactic Neural Model for General-Purpose Code Generation. In *Proceedings of ACL*.

John M Zelle and Raymond J Mooney. 1996. Learning to Parse Database Queries using Inductive Logic Programming. In *Proceedings of AAAI-1996*. pages 1050–1055.

Luke S. Zettlemoyer and Michael Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *Proceedings of ACL-2009*. pages 976–984.

Luke S Zettlemoyer and Michael Collins. 2012. Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars. *arXiv preprint arXiv:1207.1420* .

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer Learning for Low-resource Neural Machine Translation. *Proceedings of ACL* .

Neural models of factuality

Rachel Rudinger
Johns Hopkins University

Aaron Steven White
University of Rochester

Benjamin Van Durme
Johns Hopkins University

Abstract

We present two neural models for event factuality prediction, which yield significant performance gains over previous models on three event factuality datasets: FactBank, UW, and MEANTIME. We also present a substantial expansion of the It Happened portion of the Universal Decompositional Semantics dataset, yielding the largest event factuality dataset to date. We report model results on this extended factuality dataset as well.

1 Introduction

A central function of natural language is to convey information about the properties of events. Perhaps the most fundamental of these properties is *factuality*: whether an event happened or not.

A natural language understanding system's ability to accurately predict event factuality is important for supporting downstream inferences that are based on those events. For instance, if we aim to construct a knowledge base of events and their participants, it is crucial that we know which events to include and which ones not to.

The *event factuality prediction* task (EFP) involves labeling event-denoting phrases (or their heads) with the (non)factuality of the events denoted by those phrases (Saurí and Pustejovsky, 2009, 2012; de Marneffe et al., 2012). Figure 1 exemplifies such an annotation for the phrase headed by *leave* in (1), which denotes a factual event (⊕=factual, ⊖=nonfactual).

(1) Jo failed to **leave** no trace. ⊕

In this paper, we present two neural models of event factuality (and several variants thereof). We show that these models significantly outperform previous systems on four existing event factuality datasets – FactBank (Saurí and Pustejovsky, 2009), the UW dataset (Lee et al., 2015), MEANTIME (Minard et al., 2016), and Universal Decompositional Semantics It Happened v1 (UDS-IH1; White et al., 2016) – and we demonstrate the efficacy of multi-task training and ensembling in this setting. In addition, we collect and release an extension of the UDS-IH1 dataset, which we refer to as UDS-IH2, to cover the entirety of the English Universal Dependencies v1.2 (EUD1.2) treebank (Nivre et al., 2015), thereby yielding the largest event factuality dataset to date.[1]

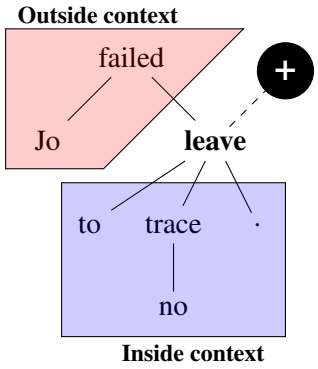


Figure 1: Event factuality (⊕=factual) and inside v. outside context for *leave* in the dependency tree.

We begin with theoretical motivation for the models we propose as well as discussion of prior EFP datasets and systems (§2). We then describe our own extension of the UDS-IH1 dataset (§3), followed by our neural models (§4). Using the data we collect, along with the existing datasets, we evaluate our models (§6) in five experimental settings (§5) and analyze the results (§7).

2 Background

2.1 Linguistic description

Words from effectively every syntactic category can convey information about the factuality of an event. For instance, negation (2a), modal auxiliaries (2b), determiners (2c), adverbs (2d), verbs (2e), adjectives (2f), and nouns (2g) can all con-

[1] Data available at decomp.net.

Proceedings of NAACL-HLT 2018, pages 731–744
New Orleans, Louisiana, June 1 - 6, 2018.

vey that a particular event – in the case of (2), a leaving event – did not happen.

(2) a. Jo didn't **leave**.
b. Jo might **leave**.
c. Jo **left** no trace.
d. Jo never **left**.
e. Jo failed to **leave**.
f. Jo's **leaving** was fake.
g. Jo's **leaving** was a hallucination.

Further, such words can interact to yield nontrivial effects on factuality inferences: (3a) conveys that the leaving didn't happen, while the superficially similar (3b) does not.

(3) a. Jo didn't remember to **leave**. ⊖
b. Jo didn't remember **leaving**. ⊕

A main goal of many theoretical treatments of factuality is to explain why these sorts of interactions occur and how to predict them. It is not possible to cover all the relevant literature in depth, and so we focus instead on the broader kind of interactions our models need to be able to capture in order to correctly predict the factuality of an event denoted by a particular predicate—namely, interactions between that predicate's *outside* and *inside* context, exemplified in Figure 1.

Outside context Factuality information coming from the outside context is well-studied in the domain of clause-embedding predicates, which break into at least four categories: factives, like *know* and *love* (Kiparsky and Kiparsky, 1970; Karttunen, 1971b; Hintikka, 1975); implicatives, like *manage* and *fail* (Karttunen, 1971a, 2012, 2013; Karttunen et al., 2014), veridicals, like *prove* and *verify* (Egré, 2008; Spector and Egré, 2015), and non-veridicals, like *hope* and *want*.

Consider the factive-implicative verb *forget* (Karttunen, 1971a; White, 2014).

(4) a. Jo forgot that Bo **left**. ⊕
b. Jo forgot to **leave**. ⊖

(5) a. Jo didn't forget that Bo **left**. ⊕
b. Jo didn't forget to **leave**. ⊕

When a predicate directly embedded by *forget* is tensed, as in (4a) and (5a), we infer that that predicate denotes a factual event, regardless of whether *forget* is negated. In contrast, when a predicate directly embedded by *forget* is untensed, as in (4b) and (5b), our inference is dependent on whether *forget* is negated. Thus, any model that correctly predicts factuality will need to not only be able to represent the effect of individual words in the outside context on factuality inferences, it will furthermore need to represent their interaction.

Inside context Knowledge of the inside context is important for integrating factuality information coming from a predicate's arguments—e.g. from determiners, like *some* and *no*.

(6) a. Some girl **ate** some dessert. ⊕
b. Some girl **ate** no dessert. ⊖
c. No girl **ate** no dessert. ⊕

In simple monoclausal sentences like those in (6), the number of arguments that contain a negative quantifier, like *no*, determine the factuality of the event denoted by the verb. An even number (or zero) will yield a factuality inference and an odd number will yield a nonfactuality inference. Thus, as for outside context, any model that correctly predicts factuality will need to integrate interactions between words in the inside context.

The (non)necessity of syntactic information One question that arises in the context of inside and outside information is whether syntactic information is strictly necessary for capturing the relevant interactions between the two. To what extent is linear precedence sufficient for accurately computing factuality?

We address these questions using two bidirectional LSTMs—one that has a linear chain topology and another that has a dependency tree topology. Both networks capture context on either side of an event-denoting word, but each does it in a different way, depending on its topology. We show below that, while both networks outperform previous models that rely on deterministic rules and/or hand-engineered features, the linear chain-structured network reliably outperforms the tree-structured network.

2.2 Event factuality datasets

Saurí and Pustejovsky (2009) present the FactBank corpus of event factuality annotations, built on top of the TimeBank corpus (Pustejovsky et al., 2006). These annotations (performed by trained annotators) are discrete, consisting of an epistemic modal $\{$*certain*, *probable*, *possible*$\}$ and a polarity $\{+,-\}$. In FactBank, factuality judgments are with respect to a *source*; following recent work, here we consider only judgments with respect to a single source: the author. The smaller MEANTIME corpus (Minard et al., 2016) includes sim-

Dataset	Train	Dev	Test	Total
FactBank	6636	2462	663	9761
MEANTIME	967	210	218	1395
UW	9422	3358	864	13644
UDS-IH2	22108	2642	2539	27289

Table 1: Number of annotated predicates.

ilar discrete factuality annotations. de Marneffe et al. (2012) re-annotate a portion of FactBank using crowd-sourced ordinal judgments to capture pragmatic effects on readers' factuality judgments.

Lee et al. (2015) construct an event factuality dataset – henceforth, UW – on the TempEval-3 data (UzZaman et al., 2013) using crowdsourced annotations on a $[-3, 3]$ scale (*certainly did not happen* to *certainly did*), with over 13,000 predicates. Adopting the $[-3, 3]$ scale of Lee et al. (2015), Stanovsky et al. (2017) assemble a Unified Factuality dataset, mapping the discrete annotations of both FactBank and MEANTIME onto the UW scale. Each scalar annotation corresponds to a token representing the event, and each sentence may have more than one annotated token.

The UDS-IH1 dataset (White et al., 2016) consists of factuality annotations over 6,920 event tokens, obtained with another crowdsourcing protocol. We adopt this protocol, described in §3, to collect roughly triple this number of annotations. We train and evaluate our factuality prediction models on this new dataset, UDS-IH2, as well as the unified versions of UW, FactBank, and MEANTIME.

Table 1 shows the number of annotated predicates in each split of each factuality dataset used in this paper. Annotations relevant to event factuality and polarity appear in a number of other resources, including the Penn Discourse Treebank (Prasad et al., 2008), MPQA Opinion Corpus (Wiebe and Riloff, 2005), the LU corpus of author belief commitments (Diab et al., 2009), and the ACE and ERE formalisms. Soni et al. (2014) annotate Twitter data for factuality.

2.3 Event factuality systems

Nairn et al. (2006) propose a deterministic algorithm based on hand-engineered lexical features for determining event factuality. They associate certain clause-embedding verbs with *implication signatures* (Table 2), which are used in a recursive polarity propagation algorithm. TruthTeller is also a recursive rule-based system for factuality ("predicate truth") prediction using implication signatures, as well as other lexical- and dependency tree-based features (Lotan et al., 2013).

Several systems use supervised models trained over rule-based features. Diab et al. (2009) and Prabhakaran et al. (2010) use SVMs and CRFs over lexical and dependency features for predicting author belief commitments, which they treat as a sequence tagging problem. Lee et al. (2015) train an SVM on lexical and dependency path features for their factuality dataset. Saurí and Pustejovsky (2012) and Stanovsky et al. (2017) train support vector models over the outputs of rule-based systems, the latter with TruthTeller.

3 Data collection

Even the largest currently existing event factuality datasets are extremely small from the perspective of related tasks, like natural language inference (NLI). Where FactBank, UW, MEANTIME, and the original UDS-IH1 dataset have on the order of 30,000 labeled examples combined, standard NLI datasets, like the Stanford Natural Language Inference (SNLI; Bowman et al. 2015) dataset, have on the order of 500,000.

To begin to remedy this situation, we collect an extension of the UDS-IH1 dataset. The resulting UDS-IH2 dataset covers all predicates in EUD1.2. Beyond substantially expanding the amount of publicly available event factuality annotations, another major benefit is that EUD1.2 consists entirely of gold parses and has a variety of other annotations built on top of it, making future multitask modeling possible.

We use the protocol described by White et al. (2016) to construct UDS-IH2. This protocol involves four kinds of questions for a particular predicate candidate:

1. UNDERSTANDABLE: whether the sentence is understandable
2. PREDICATE: whether or not a particular word refers to an eventuality (event or state)
3. HAPPENED: whether or not, according to the author, the event has already happened or is currently happening
4. CONFIDENCE: how confident the annotator is about their answer to HAPPENED from 0-4

If an annotator answers *no* to either UNDERSTANDABLE or PREDICATE, HAPPENED and CONFIDENCE do not appear.

The main differences between this protocol and the others discussed above are: (i) instead of asking about annotator confidence, the other proto-

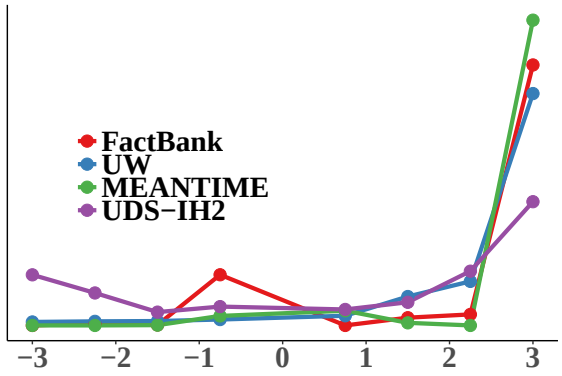


Figure 2: Relative frequency of factuality ratings in training and development sets.

cols ask the annotator to judge either source confidence or likelihood; and (ii) factuality and confidence are separated into two questions. We choose to retain White et al.'s protocol to maintain consistency with the portions of EUD1.2 that were already annotated in UDS-IH1.

Annotators We recruited 32 unique annotators through Amazon's Mechanical Turk to annotate 20,580 total predicates in groups of 10. Each predicate was annotated by two distinct annotators. Including UDS-IH1, this brings the total number of annotated predicates to 27,289.

Raw inter-annotator agreement for the HAPPENED question was 0.84 (Cohen's κ=0.66) among the predicates annotated only for UDS-IH2. This compares to the raw agreement score of 0.82 reported by White et al. (2016) for UDS-IH1.

To improve the overall quality of the annotations, we filter annotations from annotators that display particularly low agreement with other annotators on HAPPENED and CONFIDENCE. (See the Supplementary Materials for details.)

Pre-processing To compare model results on UDS-IH2 to those found in the unified datasets of Stanovsky et al. (2017), we map the HAPPENED and CONFIDENCE ratings to a single FACTUALITY value in [-3,3] by first taking the mean confidence rating for each predicate and mapping FACTUALITY to $\frac{3}{4}$CONFIDENCE if HAPPENED and -$\frac{3}{4}$CONFIDENCE otherwise.

Response distribution Figure 2 plots the distribution of factuality ratings in the train and dev splits for UDS-IH2, alongside those of FactBank, UW, and MEANTIME. One striking feature of these distributions is that UDS-IH2 displays a much more entropic distribution than the other datasets. This may be due to the fact that, unlike the newswire-heavy corpora that the other datasets annotate, EUD1.2 contains text from genres – weblogs, newsgroups, email, reviews, and question-answers – that tend to involve less reporting of raw facts. One consequence of this more entropic distribution is that, unlike the datasets discussed above, it is much harder for systems that always guess 3 – i.e. factual with high confidence/likelihood – to perform well.

4 Models

We consider two neural models of factuality: a stacked bidirectional linear chain LSTM (§4.1) and a stacked bidirectional child-sum dependency tree LSTM (§4.2). To predict the factuality v_t for the event referred to by a word w_t, we use the hidden state at t from the final layer of the stack as the input to a two-layer regression model (§4.3).

4.1 Stacked bidirectional linear LSTM

We use a standard stacked bidirectional linear chain LSTM (stacked L-biLSTM), which extends the unidirectional linear chain LSTM (Hochreiter and Schmidhuber, 1997) by adding the notion of a layer $l \in \{1, \ldots, L\}$ and a direction $d \in \{\rightarrow, \leftarrow\}$ (Graves et al., 2013; Sutskever et al., 2014; Zaremba and Sutskever, 2014).

$$\mathbf{f}_t^{(l,d)} = \sigma\left(\mathbf{W}_{\text{f}}^{(l,d)}\left[\mathbf{h}_{\mathbf{prev}_d(t)}^{(l,d)}; \mathbf{x}_t^{(l,d)}\right] + \mathbf{b}_{\text{f}}^{(l,d)}\right)$$
$$\mathbf{i}_t^{(l,d)} = \sigma\left(\mathbf{W}_{\text{i}}^{(l,d)}\left[\mathbf{h}_{\mathbf{prev}_d(t)}^{(l,d)}; \mathbf{x}_t^{(l,d)}\right] + \mathbf{b}_{\text{i}}^{(l,d)}\right)$$
$$\mathbf{o}_t^{(l,d)} = \sigma\left(\mathbf{W}_{\text{o}}^{(l,d)}\left[\mathbf{h}_{\mathbf{prev}_d(t)}^{(l,d)}; \mathbf{x}_t^{(l,d)}\right] + \mathbf{b}_{\text{o}}^{(l,d)}\right)$$
$$\hat{\mathbf{c}}_t^{(l,d)} = g\left(\mathbf{W}_{\text{c}}^{(l,d)}\left[\mathbf{h}_{\mathbf{prev}_d(t)}^{(l,d)}; \mathbf{x}_t^{(l,d)}\right] + \mathbf{b}_{\text{c}}^{(l,d)}\right)$$
$$\mathbf{c}_t^{(l,d)} = \mathbf{i}_t^{(l,d)} \circ \hat{\mathbf{c}}_t^{(l,d)} + \mathbf{f}_t^{(l,d)} \circ \mathbf{c}_{\mathbf{prev}_d(t)}^{(l,d)}$$
$$\mathbf{h}_t^{(l,d)} = \mathbf{o}_t^{(l,d)} \circ g\left(\mathbf{c}_t^{(l,d)}\right)$$

where $\circ$ is the Hadamard product; $\mathbf{prev}_{\rightarrow}(t) = t-1$ and $\mathbf{prev}_{\leftarrow}(t) = t+1$, and $\mathbf{x}_t^{(l,d)} = \mathbf{x}_t$ if $l = 1$; and $\mathbf{x}_t^{(l,d)} = [\mathbf{h}_t^{(l-1,\rightarrow)}; \mathbf{h}_t^{(l-1,\leftarrow)}]$ otherwise. We set g to the pointwise nonlinearity tanh.

4.2 Stacked bidirectional tree LSTM

We use a stacked bidirectional extension to the child-sum dependency tree LSTM (T-LSTM; Tai et al., 2015), which is itself an extension of a standard unidirectional linear chain LSTM (L-LSTM). One way to view the difference between the L-LSTM and the T-LSTM is that the T-LSTM redefines $\mathbf{prev}_{\rightarrow}(t)$ to return the set of indices that

correspond to the children of w_t in some dependency tree. Because the cardinality of these sets varies with t, it is necessary to specify how multiple children are combined. The basic idea, which we make explicit in the equations for our extension, is to define $\mathbf{f}_{tk}$ for each child index $k \in \mathbf{prev}_{\rightarrow}(t)$ in a way analogous to the equations in §4.1 – i.e. as though each child were the only child – and then sum across k within the equations for $\mathbf{i}_t$, $\mathbf{o}_t$, $\hat{\mathbf{c}}_t$, $\mathbf{c}_t$, and $\mathbf{h}_t$.

Our stacked bidirectional extension (stacked T-biLSTM) is a minimal extension to the T-LSTM in the sense that we merely define the *downward* computation in terms of a $\mathbf{prev}_{\leftarrow}(t)$ that returns the set of indices that correspond to the *parents* of w_t in some dependency tree (cf. Miwa and Bansal 2016, who propose a similar, but less minimal, model for relation extraction). The same method for combining children in the upward computation can then be used for combining parents in the downward computation. This yields a minimal change to the stacked L-biLSTM equations.

$$\mathbf{f}_{tk}^{(l,d)} = \sigma\left(\mathbf{W}_{\mathrm{f}}^{(l,d)}\left[\mathbf{h}_k^{(l,d)}; \mathbf{x}_t^{(l,d)}\right] + \mathbf{b}_{\mathrm{f}}^{(l,d)}\right)$$
$$\hat{\mathbf{h}}_t^{(l,d)} = \sum_{k \in \mathbf{prev}_d(t)} \mathbf{h}_k^{(l,d)}$$
$$\mathbf{i}_t^{(l,d)} = \sigma\left(\mathbf{W}_{\mathrm{i}}^{(l,d)}\left[\hat{\mathbf{h}}_t^{(l,d)}; \mathbf{x}_t^{(l,d)}\right] + \mathbf{b}_{\mathrm{i}}^{(l,d)}\right)$$
$$\mathbf{o}_t^{(l,d)} = \sigma\left(\mathbf{W}_{\mathrm{o}}^{(l,d)}\left[\hat{\mathbf{h}}_t^{(l,d)}; \mathbf{x}_t^{(l,d)}\right] + \mathbf{b}_{\mathrm{o}}^{(l,d)}\right)$$
$$\hat{\mathbf{c}}_t^{(l,d)} = g\left(\mathbf{W}_{\mathrm{c}}^{(l,d)}\left[\hat{\mathbf{h}}_t^{(l,d)}; \mathbf{x}_t^{(l,d)}\right] + \mathbf{b}_{\mathrm{c}}^{(l,d)}\right)$$
$$\mathbf{c}_t^{(l,d)} = \mathbf{i}_t^{(l,d)} \circ \hat{\mathbf{c}}_t^{(l,d)} + \sum_{k \in \mathbf{prev}_d(t)} \mathbf{f}_{tk}^{(l,d)} \circ \mathbf{c}_k^{(l,d)}$$
$$\mathbf{h}_t^{(l,d)} = \mathbf{o}_t^{(l,d)} \circ g\left(\mathbf{c}_t^{(l,d)}\right)$$

We use a ReLU pointwise nonlinearity for g. These minimal changes allow us to represent the inside and the outside contexts of word t (at layer l) as single vectors: $\hat{\mathbf{h}}_t^{(l,\rightarrow)}$ and $\hat{\mathbf{h}}_t^{(l,\leftarrow)}$.

An important thing to note here is that – in contrast to other dependency tree-structured T-LSTMs (Socher et al., 2014; Iyyer et al., 2014) – this T-biLSTM definition does not use the dependency labels in any way. Such labels could be straightforwardly incorporated to determine which parameters are used in a particular cell, but for current purposes, we retain the simpler structure (i) to more directly compare the L- and T-biLSTMs and (ii) because a model that uses dependency labels substantially increases the number of trainable parameters, relative to the size of our datasets.

4.3 Regression model

To predict the factuality v_t for the event referred to by a word w_t, we use the hidden states from the final layer of the stacked L- or T-biLSTM as the input to a two-layer regression model.

$$\mathbf{h}_t^{(L)} = [\mathbf{h}_t^{(L,\rightarrow)}; \mathbf{h}_t^{(L,\leftarrow)}]$$
$$\hat{v_t} = \mathbf{V}_2\, g\left(\mathbf{V}_1 \mathbf{h}_t^{(L)} + \mathbf{b}_1\right) + \mathbf{b}_2$$

where $\hat{v_t}$ is passed to a loss function $\mathbb{L}(\hat{v_t}, v_t)$: in this case, smooth L1 – i.e. Huber loss with $\delta = 1$. This loss function is effectively a smooth variant of the hinge loss used by Lee et al. (2015) and Stanovsky et al. (2017).

We also consider a simple ensemble method, wherein the hidden states from the final layers of both the stacked L-biLSTM and the stacked T-biLSTM are concatenated and passed through the same two-layer regression model. We refer to this as the H(ybrid)-biLSTM.[2]

5 Experiments

Implementation We implement both the L-biLSTM and T-biLSTM models using `pytorch 0.2.0`. The L-biLSTM model uses the stock implementation of the stacked bidirectional linear chain LSTM found in `pytorch`, and the T-biLSTM model uses a custom implementation, which we make available at decomp.net.

Word embeddings We use the 300-dimensional GloVe 42B uncased word embeddings (Pennington et al., 2014) with an UNK embedding whose dimensions are sampled iid from a Uniform[-1,1]. We do not tune these embeddings during training.

Hidden state sizes We set the dimension of the hidden states $\mathbf{h}_t^{(l,d)}$ and cell states $\mathbf{c}_t^{(l,d)}$ to 300 for all layers of the stacked L- and stacked T-biLSTMs – the same size as the input word embeddings. This means that the input to the regression model is 600-dimensional, for the stacked L- and T-biLSTMs, and 1200-dimensional, for the stacked H-biLSTM. For the hidden layer of the regression component, we set the dimension to half the size of the input hidden state: 300, for

[2] See Miwa and Bansal 2016; Bowman et al. 2016 for alternative ways of hybridizing linear and tree LSTMs for semantic tasks. We use the current method since it allows us to make minimal changes to the architectures of each model, which in turn allows us to assess the two models' ability to capture different aspects of factuality.

Verb	Signature	Type	Example
know	$+\|+$	fact.	Jo knew that Bo ate.
manage	$+\|-$	impl.	Jo managed to go.
neglect	$-\|+$	impl.	Jo neglected to call Bo.
hesitate	$\circ\|+$	impl.	Jo didn't hesitate to go.
attempt	$\circ\|-$	impl.	Jo didn't attempt to go.

Table 2: Implication signature features from Nairn et al. (2006). As an example, a signature of $-|+$ indicates negative implication under positive polarity (left side) and positive implication under negative polarity (right side); $\circ$ indicates neither positive nor negative implication.

the stacked L- and T-biLSTMs, and 600, for the stacked H-biLSTM.

Bidirectional layers We consider stacked L-, T-, and H-biLSTMs with either one or two layers. In preliminary experiments, we found that networks with three layers badly overfit the training data.

Dependency parses For the T- and H-biLSTMs, we use the gold dependency parses provided in EUD1.2 when training and testing on UDS-IH2. On FactBank, MEANTIME, and UW, we follow Stanovsky et al. (2017) in using the automatic dependency parses generated by the parser in `spaCy` (Honnibal and Johnson, 2015).[3]

Lexical features Recent work on neural models in the closely related domain of genericity/habituality prediction suggests that inclusion of hand-annotated lexical features can improve classification performance (Becker et al., 2017). To assess whether similar performance gains can be obtained here, we experiment with lexical features for simple factive and implicative verbs (Kiparsky and Kiparsky, 1970; Karttunen, 1971a). When in use, these features are concatenated to the network's input word embeddings so that, in principle, they may interact with one another and inform other hidden states in the biLSTM, akin to how verbal implicatives and factives are observed to influence the factuality of their complements. The hidden state size is increased to match the input embedding size. We consider two types:

Signature features We compute binary features based on a curated list of 92 simple implicative and 95 factive verbs including their their type-level "implication signatures," as compiled by Nairn et al. (2006).[4] These signatures characterize the implicative or factive behavior of a verb with respect to its complement clause, how this behavior changes (or does not change) under negation, and how it composes with other such verbs under nested recursion. We create one indicator feature for each signature type.

Mined features Using a simplified set of pattern matching rules over Common Crawl data (Buck et al., 2014), we follow the insights of Pavlick and Callison-Burch (2016) – henceforth, PC – and use corpus mining to automatically score verbs for implicativeness. The insight of PC lies in Karttunen's (1971a) observation that "the main sentence containing an implicative predicate and the complement sentence necessarily agree in tense."

Accordingly, PC devise a *tense agreement score* – effectively, the ratio of times an embedding predicate's tense matches the tense of the predicate it embeds – to predict implicativeness in English verbs. Their scoring method involves the use of fine-grained POS tags, the Stanford Temporal Tagger (Chang and Manning, 2012), and a number of heuristic rules, which resulted in a confirmation that tense agreement statistics are predictive of implicativeness, illustrated in part by observing a near perfect separation of a list of implicative and non-implicative verbs from Karttunen (1971a).

dare to	1.00	intend to	0.83
bother to	1.00	want to	0.77
happen to	0.99	decide to	0.75
forget to	0.99	promise to	0.75
manage to	0.97	agree to	0.35
try to	0.96	plan to	0.20
get to	0.90	hope to	0.05
venture to	0.85		

Table 3: Implicative (bold) and non-implicative (not bold) verbs from Karttunen (1971a) are nearly separable by our tense agreement scores, replicating the results of PC.

We replicate this finding by employing a simplified pattern matching method over 3B sentences of raw Common Crawl text. We efficiently search for instances of any pattern of the form: `I $VERB to * $TIME`, where `$VERB` and `$TIME` are pre-instantiated variables so their corresponding tenses are known, and '`*`' matches any one to three whitespace-separated tokens at runtime (not pre-instantiated).[5] Our results in Table 3 are a close

[3] In rebuilding the Unified Factuality dataset (Stanovsky et al., 2017), we found that sentence splitting was potentially sensitive to the version of `spaCy` used. We used v1.9.0.

[4] `http://web.stanford.edu/group/csli_lnr/Lexical_Resources`

[5] To instantiate `$VERB`, we use a list of 1K clause-embedding verbs compiled by (White and Rawlins, 2016) as well as the python package `pattern-en` to conjugate each verb in past, present progressive, and future tenses; all conjugations are first-person singular. `$TIME` is instantiated

	FactBank		UW		Meantime		UDS-IH2	
	MAE	r	MAE	r	MAE	r	MAE	r
All-3.0	0.8	NAN	0.78	NAN	0.31	NAN	2.255	NAN
Lee et al. 2015	-	-	0.511	0.708	-	-	-	-
Stanovsky et al. 2017	0.59	0.71	**0.42**†	0.66	0.34	0.47	-	-
L-biLSTM(2)-S	**0.427**	**0.826**	0.508	**0.719**	0.427	0.335	**0.960**†	**0.768**
T-biLSTM(2)-S	**0.577**	**0.752**	0.600	0.645	0.428	0.094	**1.101**	**0.704**
L-biLSTM(2)-G	**0.412**	**0.812**	0.523	0.703	0.409	0.462	-	-
T-biLSTM(2)-G	**0.455**	**0.809**	0.567	0.688	0.396	0.368	-	-
L-biLSTM(2)-S+lexfeats	**0.429**	**0.796**	0.495	**0.730**	0.427	0.322	**1.000**	**0.755**
T-biLSTM(2)-S+lexfeats	**0.542**	**0.744**	0.567	0.676	0.375	0.242	**1.087**	**0.719**
L-biLSTM(2)-MultiSimp	**0.353**	**0.843**	0.503	**0.725**	0.345	**0.540**	-	-
T-biLSTM(2)-MultiSimp	**0.482**	**0.803**	0.599	0.645	0.545	0.237	-	-
L-biLSTM(2)-MultiBal	**0.391**	**0.821**	0.496	**0.724**	**0.278**	**0.613**†	-	-
T-biLSTM(2)-MultiBal	**0.517**	**0.788**	0.573	0.659	0.400	0.405	-	-
L-biLSTM(1)-MultiFoc	**0.343**	**0.823**	0.516	0.698	**0.229**†	**0.599**	-	-
L-biLSTM(2)-MultiFoc	**0.314**	**0.846**	0.502	**0.710**	**0.305**	0.377	-	-
T-biLSTM(2)-MultiFoc	1.100	0.234	0.615	0.616	0.395	0.300	-	-
L-biLSTM(2)-MultiSimp w/UDS-IH2	**0.377**	**0.828**	0.508	**0.722**	0.367	0.469	**0.965**	**0.771**†
T-biLSTM(2)-MultiSimp w/UDS-IH2	0.595	**0.716**	0.598	0.609	0.467	0.345	**1.072**	**0.723**
H-biLSTM(2)-S	0.488	**0.775**	0.526	**0.714**	0.442	0.255	**0.967**	**0.768**
H-biLSTM(1)-MultiSimp	**0.313**†	**0.857**†	0.528	0.704	0.314	0.545	-	-
H-biLSTM(2)-MultiSimp	**0.431**	**0.808**	0.514	**0.723**	0.401	0.461	-	-
H-biLSTM(2)-MultiBal	**0.386**	**0.825**	0.502	**0.713**	0.352	**0.564**	-	-
H-biLSTM(2)-MultiSimp w/UDS-IH2	**0.393**	**0.820**	0.481	**0.749**†	0.374	**0.495**	**0.969**	**0.760**

Table 4: All 2-layer systems, and 1-layer systems if best in column. State-of-the-art in bold; † is best in column (with row shaded in purple). Key: L=linear, T=tree, H=hybrid, (1,2)=# layers, S=single-task specific, G=single-task general, +lexfeats=with all lexical features, MultiSimp=multi-task simple, MultiBal=multi-task balanced, MultiFoc=multi-task focused, w/UDS-IH2=trained on all data incl. UDS-IH2. All-3.0 is the constant baseline.

replication of PC's findings. Prior work such as by PC is motivated in part by the potential for corpus-linguistic findings to be used as fodder in downstream predictive tasks: we include these agreement scores as potential input features to our networks to test whether contemporary models do in fact benefit from this information.

Training For all experiments, we use stochastic gradient descent to train the LSTM parameters and regression parameters end-to-end with the Adam optimizer (Kingma and Ba, 2015), using the default learning rate in `pytorch` (`1e-3`). We consider five training regimes:[6]

1. SINGLE-TASK SPECIFIC (-S) Train a separate instance of the network for each dataset, training only on that dataset.
2. SINGLE-TASK GENERAL (-G) Train one instance of the network on the simple concatenation of all unified factuality datasets, {FactBank, UW, MEANTIME}.
3. MULTI-TASK SIMPLE (-MULTISIMP) Same as SINGLE-TASK GENERAL, except the network maintains a distinct set of regression parameters for each dataset; all other parameters (LSTM) remain tied. "w/UDS-IH2" is specified if UDS-IH2 is included in training.
4. MULTI-TASK BALANCED (-MULTIBAL) Same as MULTI-TASK SIMPLE but upsampling examples from the smaller datasets to ensure that examples from those datasets are seen at the same rate.
5. MULTI-TASK FOCUSED (-MULTIFOC) Same as MULTI-TASK SIMPLE but upsampling examples from a particular target dataset to ensure that examples from that dataset are seen 50% of the time and examples from the other datasets are seen 50% (evenly distributed across the other datasets).

Calibration Post-training, network predictions are monotonically re-adjusted to a specific dataset using isotonic regression (fit on train split only).

Evaluation Following Lee et al. (2015) and Stanovsky et al. (2017), we report two evaluation measures: mean absolute error (MAE) and Pearson correlation (r). We would like to note, however, that we believe correlation to be a better indicator of performance for two reasons: (i) for datasets with a high degree of label imbalance

with each of five past tense phrases ("yesterday," "last week," etc.) and five corresponding future tense phrases ("tomorrow," "next week," etc). See Supplement for further details.

[6] *Multi-task* can have subtly different meanings in the NLP community; following terminology from Mou et al. (2016), our use is best described as "semantically equivalent transfer" with simultaneous (MULT) network training.

Modal	Negated	Mean Label	Linear MAE	Tree MAE	#
NONE	no	1.00	0.93	1.03	2244
NONE	yes	-0.19	1.40	1.69	98
may	no	-0.38	1.00	0.99	14
would	no	-0.61	0.85	0.99	39
ca(n't)	yes	-0.72	1.28	1.55	11
can	yes	-0.75	0.99	0.86	6
(wi)'ll	no	-0.94	1.47	1.14	8
could	no	-1.03	0.97	1.32	20
can	no	-1.25	1.02	1.21	73
might	no	-1.25	0.66	1.06	6
would	yes	-1.27	0.40	0.86	5
should	no	-1.31	1.20	1.01	22
will	no	-1.88	0.75	0.86	75

Table 5: Mean gold labels, counts, and MAE for L-biLSTM(2)-S and T-biLSTM(2)-S model predictions on UDS-IH2-dev, grouped by modals and negation.

(Figure 2), a baseline that always guesses the mean or mode label can be difficult to beat in terms of MAE but not correlation, and (ii) MAE is harder to meaningfully compare across datasets with different label mean and variance.

Development Under all regimes, we train the model for 20 epochs – by which time all models appear to converge. We save the parameter values after the completion of each epoch and then score each set of saved parameter values on the development set for each dataset. The set of parameter values that performed best on dev in terms of Pearson correlation for a particular dataset were then used to score the test set for that dataset.

6 Results

Table 4 reports the results for all of the 2-layer L-, T-, and H-biLSTMs.[7] The best-performing system for each dataset and metric are highlighted in purple, and when the best-performing system for a particular dataset was a 1-layer model, that system is included in Table 4.

New state of the art For each dataset and metric, with the exception of MAE on UW, we achieve state of the art results with multiple systems. The highest-performing system for each is reported in Table 4. Our results on UDS-IH2 are the first reported numbers for this new factuality resource.

Linear v. tree topology On its own, the biLSTM with linear topology (L-biLSTM) performs consistently better than the biLSTM with tree topology (T-biLSTM). However, the hybrid topology (H-biLSTM), consisting of both a L- and T-biLSTM is the top-performing system on UW for correlation (Table 4). This suggests that the T-biLSTM may be contributing something complementary to the L-biLSTM.

[7] Full results are reported in the Supplementary Materials. Note that the 2-layer networks do not strictly dominate the 1-layer networks in terms of MAE and correlation.

Relation	Mean Label	L-biLSTM	T-biLSTM	#
root	1.07	1.03	0.96	949
conj	0.37	0.44	0.46	316
advcl	0.46	0.53	0.45	303
xcomp	-0.42	-0.57	-0.49	234
acl:relcl	1.28	1.40	1.31	193
ccomp	0.11	0.31	0.34	191
acl	0.77	0.59	0.58	159
parataxis	0.44	0.63	0.79	127
amod	1.92	1.88	1.81	76
csubj	0.36	0.38	0.27	37

Table 6: Mean predictions for linear (L-biLSTM-S(2)) and tree models (T-biLSTM-S(2)) on UDS-IH2-dev, grouped by governing dependency relation. Only the 10 most frequent governing dependency relations in UDS-IH2-dev are shown.

Evidence of this complementarity can be seen in Table 6, which contains a breakdown of system performance by governing dependency relation, for both linear and tree models, on UDS-IH2-dev. In most cases, the L-biLSTM's mean prediction is closer to the true mean. This appears to arise in part because the T-biLSTM is less confident in its predictions – i.e. its mean prediction tends to be closer to 0. This results in the L-biLSTM being too confident in certain cases – e.g. in the case of the `xcomp` governing relation, where the T-biLSTM mean prediction is closer to the true mean.

Lexical features have minimal impact Adding all lexical features (both SIGNATURE and MINED) yields mixed results. We see slight improvements on UW, while performance on the other datasets mostly declines (compare with SINGLE-TASK SPECIFIC). Factuality prediction is precisely the kind of NLP task one would expect these types of features to assist with, so it is notable that, in our experiments, they do not.

Multi-task helps Though our methods achieve state of the art in the single-task setting, the best performing systems are mostly multi-task (Table 4 and Supplementary Materials). This is an ideal setting for multi-task training: each dataset is relatively small, and their labels capture closely-related (if not identical) linguistic phenomena. UDS-IH2, the largest by a factor of two, reaps the smallest gains from multi-task.

Attribute	#
Grammatical error present, incl. run-ons	16
Is an auxiliary or light verb	14
Annotation is incorrect	13
Future event	12
Is a question	5
Is an imperative	3
Is not an event or state	2
One or more of the above	43

Table 7: Notable attributes of 50 instances from UDS-IH2-dev with highest absolute prediction error (using H-biLSTM(2)-MultiSim w/UDS-IH2).

manage to	2.78	agree to	-1.00
happen to	2.34	**forget to**	-1.18
dare to	1.50	want to	-1.48
bother to	1.50	intend to	-2.02
decide to	0.10	promise to	-2.34
get to	-0.23	plan to	-2.42
try to	-0.24	hope to	-2.49

Table 8: UDS-IH2-train: Infinitival-taking verbs sorted by the mean annotation scores of their complements (`xcomp`), with direct negation filtered out. Implicatives are in bold.

Verb	L-biLSTM(2)-S	+lexfeats	#
decide to	3.28	2.66	2
forget to	0.67	0.48	2
get to	1.55	1.43	9
hope to	1.35	1.23	5
intend to	1.18	0.61	1
promise to	0.40	0.49	1
try to	1.14	1.42	12
want to	1.22	1.17	24

Table 9: MAE of L-biLSTM(2)-S and L-biLSTM(2)-S+lexfeats, for predictions on events in UDS-IH2-dev that are `xcomp`-governed by an infinitival-taking verb.

7 Analysis

As discussed in §2, many discrete linguistic phenomena interact with event factuality. Here we provide a brief analysis of some of those interactions, both as they manifest in the UDS-IH2 dataset, as well as in the behavior of our models. This analysis employs the gold dependency parses present in EUD1.2.

Table 5 illustrates the influence of modals and negation on the factuality of the events they have direct scope over. The context with the highest factuality on average is *no direct modal* and *no negation* (first row); all other modal contexts have varying degrees of negative mean factuality scores, with *will* as the most negative. This is likely a result of UDS-IH2 annotation instructions to mark future events as not having happened.

Table 7 shows results from a manual error analysis on 50 events from UDS-IH2-dev with highest absolute prediction error (using H-biLSTM(2)-MultiSim w/UDS-IH2). Grammatical errors (such as run-on sentences) in the underlying text of UDS-IH2 appear to pose a particular challenge for these models; informal language and grammatical errors in UDS-IH2 is a substantial distinction from the other factuality datasets used here.

In §6 we observe that the linguistically-motivated lexical features that we test (+lexfeats) do not have a big impact on overall performance. Tables 8 and 9 help nuance this observation.

Table 8 shows that we can achieve similar separation between implicatives and non-implicatives as the feature mining strategy presented in §5. That is, those features may be redundant with information already learnable from factuality datasets (UDS-IH2). Despite the underperformance of these features overall, Table 9 shows that they may still improve performance in the subset of instances where they appear.

8 Conclusion

We have proposed two neural models of event factuality prediction – a bidirectional linear-chain LSTM (L-biLSTM) and a bidirectional child-sum dependency tree LSTM (T-biLSTM) – which yield substantial gains over previous models based on deterministic rules and hand-engineered features. We found that both models yield such gains, though the L-biLSTM outperforms the T-biLSTM; for some datasets, an ensemble of the two (H-biLSTM) improves over either alone.

We have also extended the UDS-IH1 dataset, yielding the largest publicly-available factuality dataset to date: UDS-IH2. In experiments, we see substantial gains from multi-task training over the three factuality datasets unified by Stanovsky et al. (2017), as well as UDS-IH2. Future work will further probe the behavior of these models, or extend them to learn other aspects of event semantics.

Acknowledgments

This research was supported by the JHU HLTCOE, DARPA LORELEI, DARPA AIDA, and NSF-GRFP (1232825). The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes. The views and conclusions contained in this publication are those of the authors and should not be interpreted as representing official policies or endorsements of DARPA or the U.S. Government.

References

Maria Becker, Michael Staniek, Vivi Nastase, Alexis Palmer, and Anette Frank. 2017. Classifying Semantic Clause Types: Modeling Context and Genre Characteristics with Recurrent Neural Networks and Attention. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*. pages 230–240.

Samuel R Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Berlin, Germany, pages 1466–1477.

Samuel R. Bowman, Christopher Potts, and Christopher D. Manning. 2015. Learning distributed word representations for natural logic reasoning. In *Proceedings of the AAAI Spring Symposium on Knowledge Representation and Reasoning*.

Christian Buck, Kenneth Heafield, and Bas van Ooyen. 2014. N-gram Counts and Language Models from the Common Crawl. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. European Language Resources Association (ELRA), Reykjavik, Iceland.

Angel X. Chang and Christopher Manning. 2012. Sutime: A library for recognizing and normalizing time expressions. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*. European Language Resources Association (ELRA), Istanbul, Turkey.

Marie-Catherine de Marneffe, Christopher D. Manning, and Christopher Potts. 2012. Did it happen? The pragmatic complexity of veridicality assessment. *Computational Linguistics* 38(2):301–333.

Mona T Diab, Lori Levin, Teruko Mitamura, Owen Rambow, Vinodkumar Prabhakaran, and Weiwei Guo. 2009. Committed belief annotation and tagging. In *Proceedings of the Third Linguistic Annotation Workshop*. Association for Computational Linguistics, pages 68–73.

Paul Egré. 2008. Question-embedding and factivity. *Grazer Philosophische Studien* 77(1):85–125.

Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional LSTM. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, pages 273–278.

Jaakko Hintikka. 1975. Different Constructions in Terms of the Basic Epistemological Verbs: A Survey of Some Problems and Proposals. In *The Intentions of Intentionality and Other New Models for Modalities*, Dordrecht: D. Reidel, pages 1–25.

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

Matthew Honnibal and Mark Johnson. 2015. An Improved Non-monotonic Transition System for Dependency Parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1373–1378.

Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. 2014. A neural network for factoid question answering over paragraphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 633–644.

Lauri Karttunen. 1971a. Implicative verbs. *Language* pages 340–358.

Lauri Karttunen. 1971b. Some observations on factivity. *Papers in Linguistics* 4(1):55–69.

Lauri Karttunen. 2012. Simple and phrasal implicatives. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*. Association for Computational Linguistics, pages 124–131.

Lauri Karttunen. 2013. You will be lucky to break even. In Tracy Holloway King and Valeria dePaiva, editors, *From Quirky Case to Representing Space: Papers in Honor of Annie Zaenen*, pages 167–180.

Lauri Karttunen, Stanley Peters, Annie Zaenen, and Cleo Condoravdi. 2014. The Chameleon-like Nature of Evaluative Adjectives. In Christopher Piñón, editor, *Empirical Issues in Syntax and Semantics 10*. CSSP-CNRS, pages 233–250.

Diederik Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*. San Diego, CA, USA.

Paul Kiparsky and Carol Kiparsky. 1970. Fact. In Manfred Bierwisch and Karl Erich Heidolph, editors, *Progress in Linguistics: A collection of papers*, Mouton, The Hague, pages 143–173.

Kenton Lee, Yoav Artzi, Yejin Choi, and Luke Zettlemoyer. 2015. Event Detection and Factuality Assessment with Non-Expert Supervision. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1643–1648.

Amnon Lotan, Asher Stern, and Ido Dagan. 2013. TruthTeller: Annotating Predicate Truth. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Atlanta, Georgia, pages 752–757.

Anne-Lyse Minard, Manuela Speranza, Ruben Urizar, Begoña Altuna, Marieke van Erp, Anneleen Schoen, and Chantal van Son. 2016. MEANTIME, the NewsReader Multilingual Event and Time Corpus. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association (ELRA), Paris, France, pages 23–28.

Makoto Miwa and Mohit Bansal. 2016. End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Berlin, Germany, pages 1105–1116.

Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How transferable are neural networks in nlp applications? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 479–489.

Rowan Nairn, Cleo Condoravdi, and Lauri Karttunen. 2006. Computing relative polarity for textual inference. In *Proceedings of the Fifth International Workshop on Inference in Computational Semantics (ICoS-5)*. Association for Computational Linguistics, Buxton, England, pages 20–21.

Joakim Nivre, Zeljko Agic, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Cristina Bosco, Sam Bowman, Giuseppe G. A. Celano, Miriam Connor, Marie-Catherine de Marneffe, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Tomaž Erjavec, Richárd Farkas, Jennifer Foster, Daniel Galbraith, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Yoav Goldberg, Berta Gonzales, Bruno Guillaume, Jan Hajič, Dag Haug, Radu Ion, Elena Irimia, Anders Johannsen, Hiroshi Kanayama, Jenna Kanerva, Simon Krek, Veronika Laippala, Alessandro Lenci, Nikola Ljubešić, Teresa Lynn, Christopher Manning, Cătălina Mărănduc, David Mareček, Héctor Martínez Alonso, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Shunsuke Mori, Hanna Nurmi, Petya Osenova, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Prokopis Prokopidis, Sampo Pyysalo, Loganathan Ramasamy, Rudolf Rosa, Shadi Saleh, Sebastian Schuster, Wolfgang Seeker, Mojgan Seraji, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Kiril Simov, Aaron Smith, Jan Štěpánek, Alane Suhr, Zsolt Szántó, Takaaki Tanaka, Reut Tsarfaty, Sumire Uematsu, Larraitz Uria, Viktor Varga, Veronika Vincze, Zdeněk Žabokrtský, Daniel Zeman, and Hanzhi Zhu. 2015. Universal Dependencies 1.2. *http://universaldependencies.github.io/docs/* .

Ellie Pavlick and Chris Callison-Burch. 2016. Tense Manages to Predict Implicative Behavior in Verbs. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 2225–2229.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1532–1543.

Vinodkumar Prabhakaran, Owen Rambow, and Mona Diab. 2010. Automatic committed belief tagging. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. Association for Computational Linguistics, pages 1014–1022.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn Discourse Treebank 2.0. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC)*. European Language Resources Association (ELRA), Marrakech, Morocco.

James Pustejovsky, Marc Verhagen, Roser Saurí, Jessica Littman, Robert Gaizauskas, Graham Katz, Inderjeet Mani, Robert Knippen, and Andrea Setzer. 2006. TimeBank 1.2. *Linguistic Data Consortium* 40.

Roser Saurí and James Pustejovsky. 2009. FactBank: a corpus annotated with event factuality. *Language Resources and Evaluation* 43(3):227.

Roser Saurí and James Pustejovsky. 2012. Are you sure that this happened? assessing the factuality degree of events in text. *Computational Linguistics* 38(2):261–299.

Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association of Computational Linguistics* 2(1):207–218.

Sandeep Soni, Tanushree Mitra, Eric Gilbert, and Jacob Eisenstein. 2014. Modeling Factuality Judgments in Social Media Text. In *Proceedings of the*

52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). Association for Computational Linguistics, Baltimore, Maryland, pages 415–420.

Benjamin Spector and Paul Egré. 2015. A uniform semantics for embedded interrogatives: An answer, not necessarily the answer. *Synthese* 192(6):1729–1784.

Gabriel Stanovsky, Judith Eckle-Kohler, Yevgeniy Puzikov, Ido Dagan, and Iryna Gurevych. 2017. Integrating Deep Linguistic Features in Factuality Prediction over Unified Datasets. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 352–357.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*. pages 3104–3112.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, Beijing, China, pages 1556–1566.

Naushad UzZaman, Hector Llorens, James Allen, Leon Derczynski, Marc Verhagen, and James Pustejovsky. 2013. SemEval-2013 Task 1: TempEval-3: Evaluating Time Expressions, Events, and Temporal Relations. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. Association for Computational Linguistics, Atlanta, Georgia, pages 1–9.

Aaron Steven White. 2014. Factive-implicatives and modalized complements. In Jyoti Iyer and Leland Kusmer, editors, *Proceedings of the 44th annual meeting of the North East Linguistic Society*. University of Connecticut, pages 267–278.

Aaron Steven White and Kyle Rawlins. 2016. A computational model of S-selection. *Semantics and Linguistic Theory* 26:641–663.

Aaron Steven White, Drew Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme. 2016. Universal decompositional semantics on universal dependencies. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, TX, pages 1713–1723.

Janyce Wiebe and Ellen Riloff. 2005. Creating Subjective and Objective Sentence Classifiers from Unannotated Texts. In *Proceedings of the 6th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing-05)*. Springer-Verlag, Mexico City, Mexico, pages 486–497.

Wojciech Zaremba and Ilya Sutskever. 2014. Learning to execute. *arXiv preprint arXiv:1410.4615* .

A Appendix

A.1 Dataset filtering

We filter our dataset to remove annotators with very low agreement in two ways: (i) based on the their agreement with other annotators on the HAPPENED question; and (ii) based on the their agreement with other annotators on the CONFIDENCE question.

For the HAPPENED question, we computed, for each pair of annotators and each item that both of those annotators annotated, whether the two responses were equal. We then fit a random effects logistic regression to response equality with random intercepts for annotator. The Best Linear Unbiased Predictors (BLUPs) for each annotator were then extracted and z-scored. Annotators were removed if their z-scored BLUP was less than -2.

For the CONFIDENCE question, we first ridit-scored the ratings by annotator; and for each pair of annotators and each item that both of those annotators annotated, we computed the difference between the two ridit-scored confidences. We then fit a random effects linear regression to the resulting difference after logit-transformation with random intercepts for annotator. The same BLUP-based exclusion procedure was then used.

This filtering results in the exclusion of one annotator, who is excluded for low agreement on HAPPENED. 4,179 annotations are removed in the filtering, but because we remove only a single annotator, there remains at least one annotation for every predicate.

A.2 Mining Implicatives

All options for instantiating the `$TIME` pattern variable, described in §5, are listed here.

- Past Tense Phrases: *earlier today*, *yesterday*, *last week*, *last month*, *last year*
- Future Tense Phrases: *later today*, *tomorrow*, *next week*, *next month*, *next year*

A.3 Full Results

Table 10 presents the full set of results, including all 1-layer and 2-layer models, and performance on development splits.

	FactBank				UW				Meantime				UD			
	dev		test		dev		test		dev		test		dev		test	
	MAE	r	MAE	r	MAE	r	MAE	r	MAE	r	MAE	r	MAE	r	MAE	r
All-3.0	-	-	0.8	0	-	-	0.78	0	-	-	0.31	0	-	-	2.255	0
Lee et al. 2015	-	-	-	-	0.462	0.749	0.511	0.708	-	-	-	-	-	-	-	-
Stanovsky et al. 2017	-	-	0.59	0.71	-	-	0.42	0.66	-	-	0.31	0.47	-	-	-	-
L-biLSTM(1)-S	0.411	0.78	0.399	0.816	0.435	0.797	0.508	0.718	0.239	0.631	0.337	0.359	0.978	0.768	0.986	0.765
L-biLSTM(2)-S	0.482	0.772	0.427	0.826	0.426	0.799	0.508	0.719	0.357	0.601	0.427	0.335	0.95	0.771	0.96	0.768
T-biLSTM(1)-S	0.564	0.711	0.48	0.748	0.491	0.734	0.574	0.652	0.297	0.564	0.36	0.155	1.043	0.739	1.053	0.725
T-biLSTM(2)-S	0.595	0.71	0.577	0.752	0.497	0.735	0.6	0.645	0.351	0.371	0.428	0.094	1.06	0.719	1.101	0.704
L-biLSTM(1)-G	0.432	0.798	0.383	0.819	0.426	0.807	0.517	0.717	0.252	0.625	0.343	0.505	-	-	-	-
L-biLSTM(2)-G	0.439	0.799	0.412	0.812	0.42	0.809	0.523	0.703	0.291	0.604	0.409	0.462	-	-	-	-
T-biLSTM(1)-G	0.468	0.758	0.405	0.82	0.472	0.76	0.571	0.662	0.336	0.509	0.408	0.384	-	-	-	-
T-biLSTM(2)-G	0.498	0.764	0.455	0.809	0.481	0.757	0.567	0.688	0.298	0.527	0.396	0.368	-	-	-	-
L-biLSTM(1)-S+lexfeats:sign	0.423	0.78	0.396	0.805	-	-	-	-	-	-	-	-	-	-	-	-
L-biLSTM(2)-S+lexfeats:sign	0.459	0.768	0.423	0.82	-	-	-	-	-	-	-	-	-	-	-	-
T-biLSTM(1)-S+lexfeats:sign	0.54	0.718	0.51	0.762	-	-	-	-	-	-	-	-	-	-	-	-
T-biLSTM(2)-S+lexfeats:sign	0.552	0.731	0.558	0.748	-	-	-	-	-	-	-	-	-	-	-	-
L-biLSTM(1)-S+lexfeats:mine	0.468	0.781	0.453	0.801	-	-	-	-	-	-	-	-	-	-	-	-
L-biLSTM(2)-S+lexfeats:mine	0.416	0.768	0.373	0.808	-	-	-	-	-	-	-	-	-	-	-	-
T-biLSTM(1)-S+lexfeats:mine	0.546	0.725	0.525	0.751	-	-	-	-	-	-	-	-	-	-	-	-
T-biLSTM(2)-S+lexfeats:mine	0.567	0.727	0.573	0.72	-	-	-	-	-	-	-	-	-	-	-	-
L-biLSTM(1)-S+lexfeats:both	0.443	0.781	0.413	0.805	0.428	0.803	0.507	0.722	0.319	0.481	0.373	0.369	1.002	0.762	1.002	0.766
L-biLSTM(2)-S+lexfeats:both	0.485	0.764	0.429	0.796	0.433	0.792	0.495	0.73	0.356	0.662	0.427	0.322	0.977	0.767	1	0.755
T-biLSTM(1)-S+lexfeats:both	0.503	0.728	0.449	0.793	0.485	0.743	0.589	0.643	0.282	0.493	0.348	0.191	1.04	0.738	1.073	0.718
T-biLSTM(2)-S+lexfeats:both	0.565	0.724	0.542	0.744	0.481	0.747	0.567	0.676	0.352	0.404	0.375	0.242	1.049	0.738	1.087	0.719
L-biLSTM(1)-MultiSimp	0.408	0.804	0.365	0.834	0.414	0.825	0.506	0.736	0.241	0.506	0.286	0.453	-	-	-	-
L-biLSTM(2)-MultiSimp	0.393	0.811	0.353	0.843	0.417	0.817	0.503	0.725	0.314	0.56	0.345	0.54	-	-	-	-
T-biLSTM(1)-MultiSimp	0.464	0.756	0.408	0.807	0.472	0.754	0.555	0.67	0.248	0.546	0.318	0.357	-	-	-	-
T-biLSTM(2)-MultiSimp	0.517	0.753	0.482	0.803	0.493	0.754	0.599	0.645	0.474	0.52	0.545	0.237	-	-	-	-
L-biLSTM(1)-MultiBal	0.387	0.805	0.332	0.841	0.412	0.822	0.52	0.722	0.232	0.57	0.256	0.544	-	-	-	-
L-biLSTM(2)-MultiBal	0.441	0.8	0.391	0.821	0.414	0.815	0.496	0.724	0.251	0.624	0.278	0.613	-	-	-	-
T-biLSTM(1)-MultiBal	0.475	0.746	0.405	0.817	0.472	0.752	0.578	0.629	0.237	0.56	0.344	0.266	-	-	-	-
T-biLSTM(2)-MultiBal	0.56	0.73	0.517	0.788	0.499	0.734	0.573	0.659	0.252	0.567	0.4	0.405	-	-	-	-
L-biLSTM(1)-MultiFoc	0.378	0.79	0.343	0.823	0.414	0.813	0.516	0.698	0.256	0.48	0.229	0.599	-	-	-	-
L-biLSTM(2)-MultiFoc	0.379	0.808	0.314	0.846	0.409	0.81	0.502	0.71	0.227	0.524	0.305	0.377	-	-	-	-
T-biLSTM(1)-MultiFoc	0.469	0.748	0.401	0.81	0.474	0.754	0.579	0.654	0.29	0.533	0.354	0.293	-	-	-	-
T-biLSTM(2)-MultiFoc	1.091	0.231	1.1	0.234	0.508	0.731	0.615	0.616	0.293	0.456	0.395	0.3	-	-	-	-
L-biLSTM(1)-MultiSimp w/UDS-IH2	0.417	0.802	0.381	0.813	0.421	0.802	0.486	0.741	0.385	0.51	0.353	0.565	0.972	0.771	0.977	0.765
L-biLSTM(2)-MultiSimp w/UDS-IH2	0.439	0.794	0.377	0.828	0.418	0.803	0.508	0.722	0.305	0.541	0.367	0.469	0.959	0.774	0.965	0.771
T-biLSTM(1)-MultiSimp w/UDS-IH2	0.535	0.732	0.498	0.778	0.492	0.746	0.611	0.61	0.377	0.44	0.413	0.395	1.061	0.735	1.069	0.728
T-biLSTM(2)-MultiSimp w/UDS-IH2	0.597	0.717	0.595	0.716	0.526	0.706	0.598	0.609	0.427	0.471	0.467	0.345	1.048	0.736	1.072	0.723
H-biLSTM(1)-S	0.42	0.789	0.378	0.831	0.427	0.804	0.518	0.704	0.349	0.437	0.405	0.085	0.989	0.767	0.992	0.765
H-biLSTM(2)-S	0.505	0.739	0.488	0.775	0.467	0.765	0.526	0.714	0.352	0.595	0.442	0.255	0.948	0.775	0.967	0.768
H-biLSTM(1)-MultiSimp	0.395	0.802	0.313	0.857	0.417	0.821	0.528	0.704	0.267	0.601	0.314	0.545	-	-	-	-
H-biLSTM(2)-MultiSimp	0.472	0.77	0.431	0.808	0.431	0.792	0.514	0.723	0.359	0.547	0.401	0.461	-	-	-	-
H-biLSTM(1)-MultiBal	0.398	0.803	0.334	0.853	0.402	0.829	0.497	0.733	0.229	0.59	0.264	0.432	-	-	-	-
H-biLSTM(2)-MultiBal	0.42	0.797	0.386	0.825	0.418	0.811	0.502	0.713	0.302	0.571	0.352	0.564	-	-	-	-
H-biLSTM(1)-MultiSimp w/UDS-IH2	0.431	0.785	0.365	0.833	0.431	0.8	0.513	0.733	0.277	0.569	0.341	0.286	0.982	0.765	0.98	0.761
H-biLSTM(2)-MultiSimp w/UDS-IH2	0.44	0.79	0.393	0.82	0.422	0.815	0.481	0.749	0.306	0.556	0.374	0.495	0.97	0.764	0.969	0.76

Table 10: Full table of results, including all 1-layer and 2-layer models.

Accurate Text-Enhanced Knowledge Graph Representation Learning

Bo An[1,2], **Bo Chen**[1,2], **Xianpei Han**[1], **Le Sun**[1]
[1]State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences, Beijing, China
[2]University of Chinese Academy of Sciences, Beijing, China
{anbo, chenbo, xianpei, sunle}@iscas.ac.cn

Abstract

Previous representation learning techniques for knowledge graph representation usually represent the same entity or relation in different triples with the same representation, without considering the ambiguity of relations and entities. To appropriately handle the semantic variety of entities/relations in distinct triples, we propose an accurate text-enhanced knowledge graph representation learning method, which can represent a relation/entity with different representations in different triples by exploiting additional textual information. Specifically, our method enhances representations by exploiting the entity descriptions and triple-specific relation mention. And a mutual attention mechanism between relation mention and entity description is proposed to learn more accurate textual representations for further improving knowledge graph representation. Experimental results show that our method achieves the state-of-the-art performance on both link prediction and triple classification tasks, and significantly outperforms previous text-enhanced knowledge representation models.

1 Introduction

Knowledge graphs such as Freebase (Bollacker et al., 2008), YAGO (Suchanek et al., 2007) and WordNet (Miller, 1995) are among the most widely used resources in NLP applications. Typically, a knowledge graph consists of a set of triples $\{(h, r, t)\}$, where h, r, t stand for head entity, relation and tail entity respectively.

Learning distributional representation of knowledge graph has attracted many research attentions in recent years. By projecting all elements in a knowledge graph into a dense vector space, the semantic distance between all elements can be easily calculated, and thus enables many applications

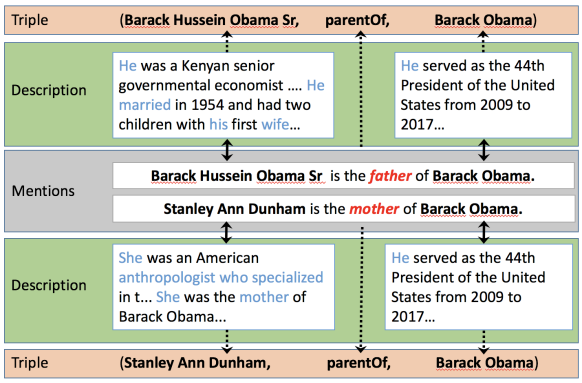


Figure 1: A demonstration of our accurate text-enhanced model. The meanings of relation `parentOf` in different triples are distinguished by their entity descriptions, and the relation `parentOf` emphasizes words which describe its social relationship in entity descriptions.

such as link prediction and triple classification (Socher et al., 2013).

Recently, translation-based models, including TransE (Bordes et al., 2013), TransH (Wang et al., 2014), TransD (Ji et al., 2015) and TransR (Lin et al., 2015b), have achieved promising results in distributional representation learning of knowledge graph. ComplEx (Trouillon et al., 2016) has achieved the state-of-the-art performance on multiple tasks, such as triple classification and link prediction. Unfortunately, all of these methods only utilize the structure information of knowledge graph, which inevitably suffer from the sparseness and incompleteness of knowledge graph. Even worse, structure information usually cannot distinguish the different meanings of relations and entities in different triples.

To address the above problem, additional information is introduced to enrich the knowledge representations, including entity types and logic rules. However, most researches of this line are limited by manually constructed logic rules, which

Proceedings of NAACL-HLT 2018, pages 745–755
New Orleans, Louisiana, June 1 - 6, 2018.

are knowledge graph sensitive and require the expert knowledge. Another type of widely used resources is textual information, such as entity descriptions and words co-occurrence with entities (Socher et al., 2013; Wang et al., 2014; Zhong et al., 2015).

The main drawback of the above methods is that they represent the same entity/relation in different triples with a unique representation. Unfortunately, by detailed analyzing the triples in knowledge graph, we find two problems of the unique representation: (1) Relations are ambiguous, i.e., the accurate semantic meaning of a relation in a specific triple is related to the entities in the same triple. For example, the relation "`parentOf`" may refer to two different meanings of (i.e., "*father*" and "*mother*"), depending on the entities in triples. (2) Because different relations may concern different attributes of an entity, the same entity may express different aspects in different triples. For example, different words in the description of "*Barack Obama*" should be emphasized by relations "`parentOf`" and "`professionOf`". The ambiguity of entity/relation has been considered as one of the primary reasons why translation-based models cannot handle 1-to-N, N-to-1 and N-to-N categories of relations (Wang et al., 2014). Wang et al. (2016) tried to solve the two issues using words co-occurrence with the entities in the same sentences. Despite its apparent success, there remains a major drawback: this method suffers from noisy text, which reduces the value of textual information.

To solve above problems, this paper proposes an accurate text-enhanced knowledge representation model, which can enhance the representations of entities and relations by incorporating accurate textual information for each triple. To learn the representation of a given triple, we first extract its accurate relation mentions from text corpus, which reflect the specific relationship between its head entity and tail entity. Then a mutual attention mechanism between relation mention and entity descriptions (extracted from knowledge graph), is introduced to enhance the representations of entities and relations. For example, the two triples in Figure 1 have the same "`parentOf`" relationship, but have different underlying semantics "*was the father of*" and "*was the mother of*" respectively. Besides, our mutual attention mechanism enables knowledge representation focusing more on related information from text information. For example, the "`parentOf`" relation will concern more about the social relations and gender attributes of a person, rather than his/her jobs, which are also contained in its descriptions. And such a relation-specific entity description will make an entity has more appropriate, relation-specific representations in different triples.

Concretely, we employ BiLSTM model (Schuster and Paliwal, 1997; Graves and Schmidhuber, 2005) with mutual attention mechanism (Zhou et al., 2016) to learn representations for relation mentions and entity descriptions. Specifically, in order to generate triple-specific textual representation of entities and relation, a mutual attention mechanism is proposed to model relation between entity descriptions and relation mention of one triple. Then the learned textual representations are incorporated with previous traditional transition-based representations, which are, learned from structural information of knowledge graph, directly to obtain enhanced triple specific representations of elements.

We evaluate our method on both link prediction task and triple classification task, using benchmark datasets from Freebase[1] and WordNet[2] with the text corpus. Experimental results show that, our model achieves the state-of-the-art performance, and significantly outperforms previous text-enhanced models.

The main contributions are threefold: (i) To the best of our knowledge, this is the first work which simultaneously exploits both relation mention and entity description to handle the ambiguity of relations and entities (Section 3). (ii) We propose a mutual attention mechanism which exploits the textual representations of relation and entity to enhance each other (Section 3.2). (iii) This paper achieves new state-of-the-art performances on triple classification tasks over two most widely used benchmarks (Section 4).

2 Related Work

Currently, a lot of structural-based knowledge representation learning methods have been proposed for knowledge graph completion, including Bilinear Model (Sutskever, 2009), Distance Model (Bordes et al., 2011), Unstructured Model (Bordes et al., 2012), Neural Tensor Network (Socher

[1] http://www.freebase.com
[2] http://www.princeton.edu/wordnet

et al., 2013), Single Layer Model (Socher et al., 2013). And many translation-based methods are introduced, including TransE (Bordes et al., 2013) and its extensions like TransH (Wang et al., 2014), TransD (Ji et al., 2015), TransR (Lin et al., 2015b). Xiao et al. (2016a) proposed a manifold-based embedding principle to deal with the overstrict geometric form of translation-based assumption. Trouillon et al. (2016) employed complex value embeddings to understand the structural information.

In recent years, many methods improve the knowledge representation by exploiting additional information. For example, both the path information and logic rules have been proved to be beneficial for knowledge representation (Lin et al., 2015a; Toutanova et al., 2016; Xiong et al., 2017; Xie et al., 2016; Xu et al., 2016).

One other direction to enhance knowledge representation is to utilize entity descriptions of entities and relations. Socher et al. (2013) proposed a neural tensor network model which enhances an entity's representation using the average of the word embeddings in its name. Wang et al. (2014) proposed a model which combines entity embeddings with word embeddings using its names and Wikipedia anchors. Zhong et al. (2015) further improved the model of Wang et al. (2014) by aligning entity and text using entity descriptions. Zhang et al. (2015) proposed to model entities with word embeddings of entity names or entity descriptions. Xie et al. (2016) proposed a model to learn the embeddings of a knowledge graph by modelling both knowledge triples and entity descriptions. Xu et al. (2016) learns different representations for entities based on the attention from relation. The textual mentions of relations are also explored by Fan et al. (2014). The universal schema based models (Riedel et al., 2013; Toutanova et al., 2015) enhance knowledge representation by incorporating textual triples, which assume that all the extracted triples express a relationship between the entities and they treat each pattern as a separate relation. The main drawback of these methods is that they assume all the relation mentions will express relationship between entity pairs, which inevitably introduces a lot of noisy information. For example, the sentence "`Miami Dolphins in 1966 and the Cincinnati Bengals in 1968`" does not express any relationship between "`miami_dolphins`" and "`cincinnati_bengals`". Even worse, the diversity of language often leads to the data sparsity problem.

To resolve the ambiguity of entities and relations in different triples (i.e., a relation/entity may have different meanings in different triples), Xiao et al. (2016b) proposed a generative model to handle the ambiguous relations. Wang et al. (2016) extended the translation-based models by textual information, which assigns a relation with different representations for different entity pairs, using words co-occurred with both entities in a triple. However, the words co-occur with an entity pair nay also not express the meanings of the relation between them, which will inevitably introduce noisy information for the specific triple. Compared with these methods the main advantages of our methods are: (i) We filters out noisy textual information for accurate enrich knowledge representation. (ii) We simultaneously take the ambiguity of entities and relations in various triples into consideration.

3 Accurate Text-enhanced Knowledge Graph Representation

This section presents our accurate text-enhanced knowledge graph representation learning framework. We first describe how to extract accurate textual information for a given triple, and then we propose a textual representation learning model, which generates textual representations for both entities and relation in a specific triple. Finally, we describe how to enhance knowledge representations based on the textual representations.

The framework of the proposed approach is illustrated in Figure 2.

3.1 Text Information Extraction

Given a triple, our method will first extract accurate textual mentions of its relation from a text corpus. For example, we will extract the relation mention "*Barack Hussein Obama Sr was the father of Barack Obama.*" for the triple *(Barack Hussein Obama Sr, parentOf, Barack Obama)*]]. We collect relation mentions by two steps: (1) Entity linking: linking entity names in a text corpus to entities in a knowledge graph. (2) Relation mention extraction: collecting accurate relation mentions which express the meanings of the relation in a given triple.

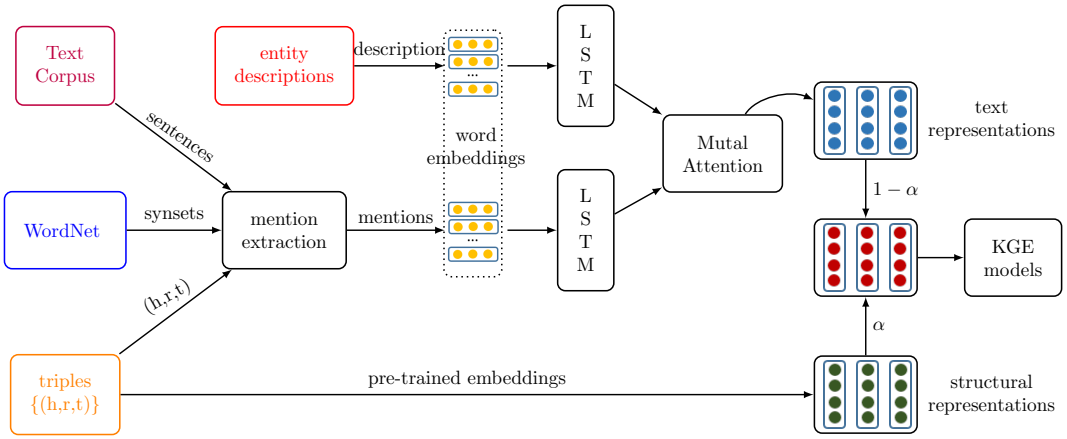


Figure 2: Framework of the proposed approach.

Entity Linking. Given a sentence $D = (w_1, w_2, ..., w_n)$, and an entity set $E = (e_1, e_2, ..., e_m)$, we first recognize entities of E in D to construct a new sentence $D' = (w_1, ..., e_1, ..., e_m, ..., w_n)$, where w_i represents the ith word in D and e_j corresponds to the jth entity in E. There are many general entity linking tools can be used for this purpose. The proposed method employs a simple and precise method to link entities of Freebase and WordNet as Wang et al. (2016). Concretely, we link a Wikipedia inner-link as an entity of Freebase if they have the same titles, and link a word in the corpus as a WordNet entity if the word belongs to one of its synsets.

Relation Mention Extraction. To extract accurate relation text mentions for a specific triple, we first collect all sentences containing both entities of the triple as candidate text mentions. And then, we calculate the similarity between a text mention and the relation based on WordNet. For example, for the triple of *(Steve Jobs, /people/person/parents, Paul Jobs)*, we treat a sentence as its accurate relation mention only if the sentence contains both of its entities and at least one hyponym/synonyms word of the relation. We collect accurate relation mentions for triples in WordNet in a similar way.

In this way, we can extract accurate relation mentions for triples with high precision. However, if a relation mention doesn't contain any hyponym/synonym words of the relation, our method would be unable to identify it. For example, the sentence "*In 1961 Obama was born in Hawaii, US*" expresses the meanings of `/people/person/nationality` in the triple `(Barack Obama,/people/person/nationality, USA )` but without any words belonging to the hyponym or synonyms of "*nationality*". For this, we further employ word embeddings to compute the similarity. Concretely, we represent a relation by averaging the pre-trained word embeddings of its last two words. Then we extract a sentence as an accurate relation mention of a given triple if the similarity between a word in the sentence and the relation representation is above a threshold, with the similarity between a word and a relation is calculated by the cosine similarity of their representations.

3.2 Learning Textual Representation

As mentioned above, the underlying semantics of entities and relations vary from different triples, and different attributes of an entity are concerned by different relations. In this section, we first utilize BiLSTM to encode relation mentions and entity descriptions. And then, we propose a mutual attention mechanism to learn more accurate text representations of relations and entities. Our model contains four layers including Embedding layer, BiLSTM layer and Mutual Attention Layer, and the details of these layers are described as follows.

Embedding Layer. To learn the distributional representation of relation mentions and entity descriptions, we convert words into distributional representations based on lookup word embeddings matrix (Mikolov et al., 2013). Concretely, given a relation mention $m = \{w_1, w_2, w_3, ..., w_n\}$, we transform the word w_i into its distributional representation $\vec{e_i} \in d^w$ using a word embeddings ma-

trix. We use the same pre-trained word embeddings as input for the BiLSTM networks of relation mentions and entity descriptions.

BiLSTM Layer. To learn the representation of text mentions, we utilize a BiLSTM (Long Short-Term Memory) (Hochreiter and Schmidhuber, 1997; Le and Zuidema, 2015; Zhou et al., 2016) model to compose the words in a sequence into the distributional representation. Concretely, we employ a two layer Bidirectional LSTM network to generate text representations. The detailed description of LSTM is presented in (Hochreiter and Schmidhuber, 1997). Two different BiLSTM networks are employed to encode relation mentions and entity descriptions respectively.

Mutual Attention Layer. Attention based neural networks have recently achieved success in a wide range of tasks, including machine translation, speech recognition and paraphrase detection (Luong et al., 2015; Yang et al., 2016; Yin et al., 2016; Vaswani et al., 2017). In this paper, we introduce a mutual attention to improve text representations. Given a triple, the goal of our mutual attention mechanism is two-fold. On one hand, our model wants to identify words in relation mention associated with the entity descriptions in the same triple. On the other hand, our model wants to recognize words in entity descriptions which are emphasized by its relation. To achieve the above goal, we first infer the representations of entity descriptions using relation representation as attention:

$$a_i(e) = \frac{exp(score(\vec{h_i}, \vec{r'}))}{\sum_{i'} exp(score(\vec{h_{i'}}, \vec{r'}))} \quad (1)$$

$$score(\vec{h_i}, \vec{r'}) = \vec{h_i}^T W_e \vec{r'} \quad (2)$$

where $\vec{r'} \in d^w$ is the representation of the relation mention by averaging all the hidden vectors of BiLSTM, $\vec{h_i}$ is the hidden representation of w_i, and $W_e \in d^{w \times 2 \times h}$ is a trained parameter matrix. The relation-sensitive representation of the entity description is generated as follows:

$$\vec{e^*} = tanh(\vec{a_e}^T H_e) \quad (3)$$

where $\vec{a_e} \in d^m$ is the relation-specific attention vector over the words in the entity description, d^m is the length of the description, $H_e \in d^{m \times h}$ is the hidden representation matrix generated by BiLSTM, and $\vec{e^*} \in d^h$ is the representation of the description. In this way, we learn the representations of entity descriptions of head entity $\vec{e_h^*} \in d^h$ and tail entity $\vec{e_t^*} \in d^h$ with the attention from relation representation.

The above two entity description representations are utilized as the attention for learning the triple-sensitive relation mention representation as follows:

$$\vec{e} = \vec{e_h^*} + \vec{e_t^*} \quad (4)$$

$$a_i(r) = \frac{exp(score(\vec{h_i}, \vec{e}))}{\sum_{i'} exp(score(\vec{h_{i'}}, \vec{e}))} \quad (5)$$

$$score(\vec{h_i}, \vec{e}) = \vec{h_i}^T W_r \vec{e} \quad (6)$$

where $\vec{e_h^*}$ and $\vec{e_t^*}$ are representations of head entity description and tail entity description respectively, $\vec{h_i}$ is the hidden vector of w_i for each word in the text mention, and $W_r \in d^{w \times 2 \times h}$ is a trained parameter matrix. The representation of the triple-sensitive relation mention is generated as Formula (7):

$$\vec{r^*} = tanh(\vec{a_r}^T H_r) \quad (7)$$

where $\vec{a_r}^T \in d^n$ is the triple-sensitive attention vector over the words in the relation mention, d^n is the length of the relation mention, $H_r \in d^{n \times h}$ is the hidden representation matrix generated by BiLSTM, and $\vec{r^*} \in d^h$ is the representation of the mention. In this way, we learn the triple-attention representation of all text mentions.

3.3 Text-Enhanced Representation Learning

In this section, we introduce how to incorporate the learned textual representations with representations learned from knowledge graph structure using previous methods.

For each given triple and its accurate textual information, we enhance the representations of the relation and entities based on the text representations of entities $\vec{e_h^*} \in d^h$, $\vec{e_t^*} \in d^h$ and relation $\vec{r^*} \in d^h$. Specifically, we enhance the relation and entity representations as follows:

$$Re(\vec{r_{ate}}) = \alpha \cdot Re(\vec{r_{kg}}) + (1-\alpha) \cdot \vec{r^*}, 0 \leq \alpha \leq 1 \quad (8)$$

$$Re(\vec{h_{ate}}) = \alpha \cdot Re(\vec{h_{kg}}) + (1-\alpha) \cdot \vec{e_h^*}, 0 \leq \alpha \leq 1 \quad (9)$$

$$Re(\vec{t_{ate}}) = \alpha \cdot Re(\vec{t_{kg}}) + (1-\alpha) \cdot \vec{e_t^*}, 0 \leq \alpha \leq 1 \quad (10)$$

where α represents the weight factor for the structural representations, $\vec{r_{kg}} \in d^h$, $\vec{h_{kg}} \in d^h$ and $\vec{t_{kg}} \in d^h$ represent the distributional representations of relation r head entity h and tail entity t

learned from structural information of knowledge graph, $\vec{r^*} \in d^h$, $\vec{e_h^*} \in d^h$ and $\vec{e_t^*} \in d^h$ represent the vectors of the text mention, head and tail entity descriptions for the triple, $\vec{r}_{ate} \in d^h$, $\vec{h}_{ate}$ and $\vec{t}_{ate}$ are the accurate text-enhanced representations of relation, head and tail entity, respectively. Note that, we enhance the real part vector of an entity with the textual representation of the entity as Formula (9) and (10), and treat the matrix representation of a relation as a vector with each element the same as the element in diagonal matrix, and then enhance its real part as Formula (8). In this way, we enhance the representation of knowledge graph, and calculate the plausibility of a triple based on their score functions.

If there is no accurate relation mention extracted for a triple, we only utilize the knowledge embeddings to estimate the plausibility of the triple, and the weight factor α is set to 1 in this case. For example, if there is no accurate relation mention extracted for triple `(Su Shi, /people/person/profession, Artist)`, then only its structural representations will be utilized to compute the plausibility of the triple. And α is set to 1 for the triples if none of the entities in it is linked.

3.4 Model Training

In the training process, the *(h, r, t, h_t, r_t, t_t)* tuples are used as supervision, where h_t, r_t and t_t are the description of head entity, relation text mention and the description of tail entity, respectively. Since there are only correct triples in the knowledge graph, following Lin et al. (2015a), we construct the corrupted tuples $(h', r, t', h_t, r_t, t_t) \in KG'$ for a $(h, r, t, h_t, r_t, t_t) \in KG$ by randomly replacing head/tail entity with entities from knowledge graph using Bernoulli Sampling Method (Wang et al., 2014). Furthermore, to train the model of text representation model, we construct the corrupted tuples $(h, r, t, h_t', r_t', t_t') \in KG'$ for a $(h, r, t, h_t, r_t, t_t) \in KG$ by random replacing the text information. We use the following margin-based ranking loss:

$$L = \sum_{q \in KG} \sum_{q' \in KG'} max(0, \gamma + f(q) - f(q')) \quad (11)$$

where f is the score function of our model, and $\gamma > 0$ is the margin between golden tuples and negative tuples, KG is the set of tuples in training dataset, and KG' is the corrupted set of tuples. The parameters of our model are optimized using the stochastic gradient descent (SGD) algorithm. To accelerate the training process and avoid overfitting, we initialize the representations of entities and relations using base models and initialize word representations with the pre-trained word embeddings, and all these embeddings are fine-tuned during training.

4 Experiments

In this section, we first describe the settings in our experiments, and then we conduct experiments of link prediction and triple classification tasks and compare our method with base models and the state-of-the-art baselines.

4.1 Experiment Settings

In this paper, we evaluate our model on four benchmark datasets: WN11, WN18, FB13 and FB15k (Bordes et al., 2013; Socher et al., 2013; Wang et al., 2014). For the text corpus, we use a snapshot of the English Wikipedia (Wiki) (Shaoul and Westbury, 2010) [3] dump in April 2016, which contains more than 1.2 billion tokens. We link entities in the text corpus to entities in Freebase and synsets in WordNet as described above, and replace entities with HEAD_TAG and TAIL_TAG. The text descriptions of entities are freely available[4]. In addition, we pre-process the word-entity corpus, including stemming, lowercasing and removing words with fewer than 5 occurrences. The statistics of the datasets and linked-entities in text corpus are shown in Table 1.

Dataset	WN11	WN18	FB13	FB15K
#Train	112,581	141,442	316,232	483,142
#Valid	2,609	5,000	5,908	50,000
#Test	10,544	5,000	23,733	59,071
# Entities	38,696	40,943	75,043	14,951
# Relations	11	18	13	1,345
#1-to-1	0	0	0	247
#1-to-N	0	0	0	179
#N-to-1	0	0	1	225
#N-to-N	11	18	12	694
#Linked	31,432	34,159	66,328	13,567

Table 1: Statistics of different datasets and the number of entities linked in Wikipedia, *#Linked* represents the number of entities linked in the text corpus. *#N-to-1* is the number of N-to-1 type of relations.

As introduced above, we implement our framework using TransE, TransH, TransR and ComplEx as base models, and evaluate on two classi-

[3] https://www.wikipedia.org/

[4] https://github.com/xrb92/DKRL

cal tasks: link prediction and triple classification. We refer $AATE\_E$ as the proposed model which enhances TransE with accurate textual informations and mutual attention mechanism, and refer ATE_E as the proposed model without mutual attention mechanism to reveal the effect of our attention mechanism.

To speed up training and reduce overfitting, we employ the SkipGram model of word2vec (Mikolov et al., 2013) to pre-train the word embeddings with the dimension of word embeddings is $d^w = 200$, the windows size is 5, the number of iterations is 5, and the number of negative samples is 10. And we pre-train the representations of entities and relations of knowledge graph using the mentioned base models, and the parameters are empirically tuned as follows: the dimension of vectors is $d^{kg} = 200$, the number of epochs is 2000 and the margin is 1.0. We implement our model based on the OpenKE [5] framework.

In our experiments, the hyper-parameters of BiLSTM are empirically set as follows: the number of hidden units is $d^h = 200$, the learning rates for SGD are among $\{0.1, 0.001, 0.0001\}$, the margin λ values are among $\{0.5, 1.0, 2.0\}$ and the batch sizes are among $\{100, 500, 2000\}$. We employ two different BiLSTM networks with the same hyper-parameters to learn the representations of text mentions and entity descriptions. And all the parameters are learned jointly, including BiLSTM networks and knowledge representations.

4.2 Link Prediction

Link prediction aims to predict missing head or tail entity of a triple, which is a widely employed evaluation task for knowledge graph completion models (Bordes et al., 2011; Wang et al., 2016). Concretely, given a head entity `h` (or tail entity `t`) and a relation `r`, the system will return a rank list of candidate entities for tail entity. Following (Bordes et al., 2013; Lin et al., 2015b), we conduct the link prediction task on WN18 and FB15k datasets.

In the testing phase, for each triple (h, r, t), we replace its head/tail entity with all entities to construct candidate triples, and extract text mentions from the text corpus for each candidate triple. Then we rank all these entities in descending order of the scores, which are calculated by our score function. Based on the entity ranking list, we employ two evaluation metrics from (Bordes et al., 2013): (1) mean rank of correct entities (MR); and (2) proportion of correct entities in top-10 rank entities Hit@10 ($Hit10$). A good link predictor should achieve low MR and high $Hit@10$. We tuned model parameters using validate datasets. We implement our framework using TransE, TransH, TransR and ComplEx as base models, and treat these base models as baselines. Furthermore, we also compare our method with the state-of-the-art results from Unstructured, SME, TransD, TEKE , Jointly (Xu et al., 2016), TransG and Mainifold, and we report the results from their original papers. The overall results are presented in Table 2.

Models		WN18		FB15K	
		MR	Hit10	MR	Hit10
Others	UnS	304	38.2	154	40.8
	SME	533	74.1	979	6.3
	TransD	212	92.2	91	77.3
	TransG	345	94.7	**50**	**88.2**
	Mainifold	-	**94.9**	-	88.1
Jointly	LSTM	**95**	91.6	90	69.7
	A-LSTM	123	90.9	73	75.5
TransE	TransE	251	89.2	125	47.1
	TEKE_E	127	93.8	79	67.6
	ATE_E	158	91.7	89	57.1
	AATE_E	123	94.1	76	76.1
TransH	TransH	303	86.7	84	58.5
	TEKE_H	128	93.6	75	70.4
	ATE_H	167	92.5	80	68.2
	AATE_H	132	94.0	73	74.6
TransR	TransR	219	91.7	78	65.5
	TEKE_R	203	92.3	79	68.5
	ATE_R	210	92.1	80	67.2
	AATE_R	185	93.7	77	69.4
ComplEx	ComplEx	219	94.7	78	84.0
	ATE_C	217	94.7	61	86.2
	AATE_C	**179**	**94.9**	**52**	**88.0**

Table 2: Evaluation results of link prediction.

From Table 2, we can see that both ATE and AATE models surpass all base models (TransE, TransH, TransR and ComplEx) on all metrics. This result verifies that the textual information is beneficial for structure-based knowledge graph representation learning models. Compared with the ATE models, the AATE models achieve better results on link prediction task, which verifies that the mutual attention between entity description and relation mention is effect for selecting meaningful words and enhancing the learning of knowledge graph representation.

For translation-based models, the proposed method achieves the best result based on TransE.

[5] http://openke.thunlp.org/

This is probably because TransH and TransR have tried to project the entity embeddings into the space of relation space, which may lead to the fact that the text information could not enhance the entity representation directly. In addition, our method implemented based on ComplEx has achieve better performances w.r.t TEKE (Wang et al., 2016) on all metrics, that verifies the importance of filtering out the noisy information.

4.2.1 Analysis on 1-to-N, N-to-1 and N-to-N Relations

To better analyse the effect of textual information for knowledge graph representation learning, this section presents the results of our model on different categories of relations including 1-N, N-1 and N-N on link prediction task. We present the results of our models based on TransE and of all baselines.

From Table 3, we can see that, both of our proposed methods have achieved higher performance over the base model on all types of relations (1-to-N, N-to-1 and N-to-N). In addition, our AATE model achieves better results than the Jointly(A-LSTM) model. Since both of AATE and Joint (A-LSTM) are implemented based on TransE, we verify that the triple-specific relation mention is valuable to improving the knowledge representation. Another reason why our proposed model achieves better results is that the attention from textual representation of relation and entity is more effective than the attention using structural representation for textual representation.

4.2.2 Fault Analysis

To gain more insight, we present a failure analysis to explore possible limitations and weaknesses of our model. In particular, several illustrative triples from the test set of FB15K are listed in Table 4. The tail entities of those triples are failed to be ranked in the top-10 candidates.

It can be seen from Table 4 that, the failures are mostly caused by the data sparsity problem, which results in relatively limited occurrences of entities and relations. All of "`Elementary school`", "`Abugida`", "`interests/collection_category /sub_categories`" and "`martial_arts/ martial_artist/martial_art`" appear less than 4 times in training data. It must also be mentioned that the triple "`(Abugida, language /language_writing_system/ languages, Khmer language)`" is included in the training data. Therefore, we can infer the first triple in Table 4 based on the above triple due to the general logic that "`language/human_language/writing _system`" and "`/language/language _writing_system/languages`" are a pair of inverse relations. Consequently, we believe it is important to incorporate the logic rules into knowledge embeddings, especially for the entities and relations with limited occurrences.

4.3 Triple Classification

In this section, we assess different models on the triple classification task. Triple classification aims to judge whether a given triple (h, r, t) is true fact or not, and it is usually modeled as a binary classification task (Socher et al., 2013; Bordes et al., 2013; Wang et al., 2016). Following Socher et al. (2013) we evaluate different systems on WN11 and FB13 datasets.

Given a triple (h, r, t) and all its accurate relation mentions and entity descriptions of this triple, In our experiments, a triple will be classified as a true fact if the score obtained by function f is below the relation-specific threshold δ_r, otherwise it will be classified as a false fact. The δ_r and the weight factor of α are optimized by maximizing classification accuracy on validation dataset, and different values of δ_r will be set for different relations. We use the same settings as link prediction task, all parameters are optimized on the validation datasets to obtain the best accuracies. We compare our method with all base models and the state-of-the-art performances from TransD, TEKE (Wang et al., 2016), TransG, Mainfold, and we report the best results from their original papers. The results are listed in Table 5.

From Table 5, we can see that: (1) The accurate textual information can consistently increase the accuracies on triple classification task. In all of the four base models, our model achieves significant improvements over TransE, TransH, TransR and ComplEx. This results verify that our method is a useful framework for exploiting textual information to enhance structure-based models; (2) Our method achieves better results on all datasets than TEKE. This result reveals that it is important to filter out the noisy data for knowledge graph representation learning. (3) Compared with the ATE model, our relation-sensitive attention

	Prediction Head (Hits@10)			Prediction Tail (Hits@10)		
Relation Category	1-to-N	N-to-1	N-to-N	1-to-N	N-to-1	N-to-N
#Triples in Test	2,078	6,084	109,526	2,078	6,084	109,526
Jointly(A-LSTM)	95.1	21.1	47.9	30.8	94.7	53.1
TransE	65.7	18.2	47.2	19.7	66.7	50.0
ATE_E	80.2	22.1	47.6	20.3	67.7	60.0
AATE_E	**96.1**	**35.2**	**49.1**	**32.2**	**98.3**	**60.3**

Table 3: Hit@10 of link prediction on different type of relations on FB15k dataset.

No	Head Entity (#)	Relation (#)	Tail Entity (#)
1	Upper Canada College (16)	education/educational_institution/school_type (728)	Elementary school (1)
2	Khmer language (9)	language/human_language/writing_system (41)	Abugida (1)
3	Film (255)	interests/collection_category/sub_categories (3)	Star Wars (31)
4	Jean-Claude Van Damme (28)	martial_arts/martial_artist/martial_art (1)	Taekwondo (155)

Table 4: The triples whose tail entities were failed to be ranked in top 10 candidates, # is the number of occurrences of the entity/relation in the training data.

Models		WN11	FB13	AVG.
Others	TransD	86.4	89.1	87.8
	TransG	87.4	**87.3**	87.4
	Mainfold	87.5	87.2	87.4
TransE	TransE	75.9	81.5	78.7
	TEKE_E	84.1	75.1	79.6
	ATE_E	84.3	75.4	79.9
	AATE_E	86.1	86.4	86.3
TransH	TransH	78.8	83.3	81.1
	TEKE_H	84.8	84.2	84.5
	ATE_H	85.1	83.9	84.5
	AATE_H	86.7	86.2	86.5
TransR	TransR	85.9	82.5	84.2
	TEKE_R	86.1	81.6	83.7
	ATE_R	86.2	84.4	85.3
	AATE_R	86.4	85.2	85.8
ComplEx	ComplEx	86.2	85.7	86.0
	ATE_C	87.2	87.1	87.2
	AATE_C	**88.0**	87.2	**87.6**

Table 5: Evaluation results of triple classification.

model improves the accuracies on all the datasets. We believe this is because mutual attention mechanism can better identify the relation-sensitive words from entity descriptions and extract entity-sensitive words from relation mention.

The results demonstrate that, our method has achieved the best performances on the triple classification task, which verifies that it is critical to filter out noisy text information to determine whether a triple should be added into knowledge graph or not.

5 Conclusions

In this paper, we propose an accurate text-enhanced knowledge graph representation framework, which can utilize accurate textual information enhance the knowledge representations of a triple, and can effectively handle the ambiguity of relations and entities through a mutual attention model between relation mentions and entity descriptions. Experiment results show that our method can achieve the state-of-the-art performance, and significantly outperforms previous text-enhanced knowledge representation models. And the mutual attention between relation mentions and entity descriptions can significantly improve the performance of knowledge representation. For future work, we want to further exploit entity types and logic rules as constraints to further improve knowledge representations.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grants no. 61433015, 61772505 and 61572477, and the Young Elite Scientists Sponsorship Program no. YESS20160177. Moreover, we sincerely thank the reviewers for their valuable comments.

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase:a collaboratively created graph database for structuring human knowledge. In *ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, Bc, Canada, June*. pages 1247–1250.

Antoine Bordes, Xavier Glorot, and Jason Weston. 2012. Joint learning of words and meaning representations for open-text semantic parsing. In *International Conference on Artificial Intelligence & Statistics*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko.

2013. Translating embeddings for modeling multi-relational data. *NIPS* pages 2787–2795.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, Usa, August*.

Miao Fan, Qiang Zhou, Emily Chang, and Thomas Fang Zheng. 2014. Transition-based knowledge graph embedding with relational mapping properties. In *PACLIC*. pages 328–337.

A Graves and J Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks* 18(56):602–610.

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

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*. pages 687–696.

Bhushan Kotnis and Vivi Nastase. 2017. Learning knowledge graph embeddings with type regularizer. *arXiv preprint arXiv:1706.09278* .

Phong Le and Willem Zuidema. 2015. Compositional distributional semantics with long short term memory. *arXiv preprint arXiv:1503.02510* .

Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015a. Modeling relation paths for representation learning of knowledge bases. *arXiv preprint arXiv:1506.00379* .

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015b. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*. pages 2181–2187.

Minh Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. *Computer Science* .

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the Acm* 38(11):39–41.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *HLT-NAACL*. pages 74–84.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45(11):2673–2681.

R. Socher, D. Chen, C. D. Manning, and A. Y. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *International Conference on Intelligent Control & Information Processing*. pages 464–469.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago:a core of semantic knowledge. In *International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May*. pages 697–706.

Ilya Sutskever. 2009. Modelling relational data using bayesian clustered tensor factorization. *Advances in Neural Information Processing Systems* pages 1821–1828.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *EMNLP*. volume 15, pages 1499–1509.

Kristina Toutanova, Xi Victoria Lin, Wen-tau Yih, Hoifung Poon, and Chris Quirk. 2016. Compositional learning of embeddings for relation paths in knowledge bases and text. In *ACL2016*. volume 1, pages 1434–1444.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*. pages 2071–2080.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* .

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*. pages 1112–1119.

Zhigang Wang, Juanzi Li, Zhiyuan LIU, and Jie TANG. 2016. Text-enhanced representation learning for knowledge graph. *To appear in IJCAI* 2016:04–17.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016a. From one point to a manifold: knowledge graph embedding for precise link prediction. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. AAAI Press, pages 1315–1321.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016b. Transg: A generative model for knowledge graph embedding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 2316–2325.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *AAAI*. pages 2659–2665.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. *arXiv preprint arXiv:1707.0669* .

Jiacheng Xu, Kan Chen, Xipeng Qiu, and Xuanjing Huang. 2016. Knowledge graph representation with jointly structural and textual encoding. *arXiv preprint arXiv:1611.08661* .

Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. 2016. Stacked attention networks for image question answering. In *IEEE Conference on Computer Vision and Pattern Recognition*. pages 21–29.

Wenpeng Yin, Hinrich Schtze, Bing Xiang, and Bowen Zhou. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Computer Science* .

Huaping Zhong, Jianwen Zhang, Zhen Wang, Hai Wan, and Zheng Chen. 2015. Aligning knowledge and text embeddings by entity descriptions. In *EMNLP*. pages 267–272.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *ACL*. pages 207–212.

Acquisition of Phrase Correspondences using Natural Deduction Proofs

Hitomi Yanaka[1]
hitomiyanaka@g.ecc.u-tokyo.ac.jp

Koji Mineshima[2]
mineshima.koji@ocha.ac.jp

Pascual Martínez-Gómez[3]
pascual.mg@aist.go.jp

Daisuke Bekki[2]
bekki@is.ocha.ac.jp

[1]The University of Tokyo
[2]Ochanomizu University
[3]Artificial Intelligence Research Center, AIST
Tokyo, Japan

Abstract

How to identify, extract, and use phrasal knowledge is a crucial problem for the task of Recognizing Textual Entailment (RTE). To solve this problem, we propose a method for detecting paraphrases via natural deduction proofs of semantic relations between sentence pairs. Our solution relies on a graph reformulation of partial variable unifications and an algorithm that induces subgraph alignments between meaning representations. Experiments show that our method can automatically detect various paraphrases that are absent from existing paraphrase databases. In addition, the detection of paraphrases using proof information improves the accuracy of RTE tasks.

1 Introduction

Recognizing Textual Entailment (RTE) is a challenging natural language processing task that aims to judge whether one text fragment logically follows from another text fragment (Dagan et al., 2013). Logic-based approaches have been successful in representing the meanings of complex sentences, ultimately having a positive impact on RTE (Bjerva et al., 2014; Beltagy et al., 2014; Mineshima et al., 2015, 2016; Abzianidze, 2015, 2016). Although logic-based approaches succeed in capturing the meanings of functional or logical words, it is difficult to capture the meanings of content words or phrases using genuine logical inference alone. This remains a crucial problem in accounting for lexical relations between content words or phrases via logical inference. To solve this problem, previous logic-based approaches use knowledge databases such as WordNet (Miller, 1995) to identify lexical relations within a sentence pair. While this solution has been successful in handling word-level paraphrases, its extension to phrase-level semantic relations is still an unsolved problem. There are three main difficulties that prevent an effective identification and use of phrasal linguistic knowledge.

The first difficulty is the presence of out-of-context phrase relations in popular databases such as the Paraphrase Database (PPDB) (Ganitkevitch et al., 2013). PPDB may suggest paraphrases that do not adhere to the context of our relevant text segments nor to their semantic structure, which might be problematic.

The second difficulty is finding semantic phrase correspondences between the relevant text segments. Typical approaches only rely on surface (Beltagy et al., 2013) or syntactic correspondences (Arase and Tsujii, 2017), often producing inaccurate alignments that significantly impact our inference capabilities. Instead, a mechanism to compute semantic phrase correspondences could potentially produce, if available, more coherent phrase pairs and solve the recurring issue of discontinuity.

The third difficulty is the intrinsic lack of coverage of databases for logical inference despite their large size. Whereas there is a relatively small number of possible word-to-word correspondences and thus their semantic relations can be enumerated, the same is not true for all phrase pairs that might be of interest. One alternative is to use functions of infinite domain (e.g., cosine similarity) between phrase representations (Tian et al., 2016), but these techniques are still under development, and we have not seen definitive successful applications when combined with logic systems.

In this study, we tackle these three problems. The contributions of this paper are summarized as follows: First, we propose a new method of detecting phrase correspondences through natu-

Proceedings of NAACL-HLT 2018, pages 756–766
New Orleans, Louisiana, June 1 - 6, 2018.

ral deduction proofs of semantic relations for a given sentence pair. Second, we show that our method automatically extracts various paraphrases that compensate for a shortage in previous paraphrase databases. Experiments show that extracted paraphrases using proof information improve the accuracy of RTE tasks.

2 Related Work

In this section, we review previous logical inference systems that are combined with lexical knowledge. The RTE system developed by Abzianidze (2016) uses WordNet as axioms and adds missing knowledge manually from the training dataset; however, this technique requires considerable human effort and is not extended to handle phrasal knowledge.

Martínez-Gómez et al. (2017) proposed an RTE system with an on-the-fly axiom injection mechanism guided by a natural deduction theorem prover. Pairs of unprovable sub-goals and plausible single premises are identified by means of a variable unification routine and then linguistic relations between their logical predicates are checked using lexical knowledge such as WordNet and VerbOcean (Chklovski and Pantel, 2004). However, this mechanism is limited to capturing word-to-word relations within a sentence pair.

Bjerva et al. (2014) proposes an RTE system where WordNet relations are used as axioms for word-to-word knowledge in theorem proving. For phrasal knowledge, PPDB is used to rephrase an input sentence pair instead of translating paraphrases into axioms. However, this solution ignores logical contexts that might be necessary when applying phrasal knowledge. Moreover, it does not apply to discontinuous phrases.

Beltagy et al. (2016) uses WordNet and PPDB as lexical knowledge in the RTE system. To increase their coverage of phrasal knowledge, the system combines a resolution strategy to align clauses and literals in a sentence pair and a statistical classifier to identify their semantic relation. However, this strategy only considers one possible set of alignments between fragments of a sentence pair, possibly causing inaccuracies when there are repetitions of content words and meta-predicates.

In our research, we propose an automatic phrase abduction mechanism to inject phrasal knowledge during the proof construction process. In addition, we consider multiple alignments by backtracking the decisions on variable and predicate unifications, which is a more flexible strategy. We represent logical formulas using graphs, since this is a general formalism that is easy to visualize and analyze. However, we use *natural deduction* (see Section 3.2) as a proof system instead of Markov Logic Networks for inference. Some research has investigated graph operations for semantic parsing (Reddy et al., 2014, 2016) and abstractive summarization (Liu et al., 2015); we contribute to these ideas by proposing a subgraph mapping algorithm that is useful for performing natural language inferences.

Considerable research efforts have been focused on the identification and extraction of paraphrases. One successful technique is associated with bilingual pivoting (Bannard and Callison-Burch, 2005; Zhao et al., 2008), in which alternative phrase translations are used as paraphrases at a certain probability. However, this technique requires large bilingual parallel corpora; moreover, word alignment errors likely cause noisy paraphrases. Another strategy is to extract phrase pairs from a monolingual paraphrase corpus using alignments between syntactic trees, guided by a linguistically motivated grammar (Arase and Tsujii, 2017). The main difference between these studies and ours is that they typically attempt alignment between words or syntactic trees, whereas we perform alignments between meaning representations, which enables the acquisition of more general paraphrases by distinguishing functional words from content words. This point is important in distinguishing among different semantic relations (e.g., antonyms and synonyms). In addition, word and syntactic alignments potentially ignore coreferences, making it difficult to find relations between many-to-many sentences. Semantic alignments enable this because coreferences must refer to the same variable as the original entity.

3 Logic-based Approach to RTE

3.1 Meaning representation

In logic-based approaches to RTE, a text T and a hypothesis H are mapped onto logical formulas T' and H'. To judge whether T entails H, we check whether $T' \Rightarrow H'$ is a theorem in a logical system.

For meaning representations, we use Neo-Davidsonian event semantics (Parsons, 1990). In this approach, a verb is analyzed as a one-place predicate over events. Both the arguments of a

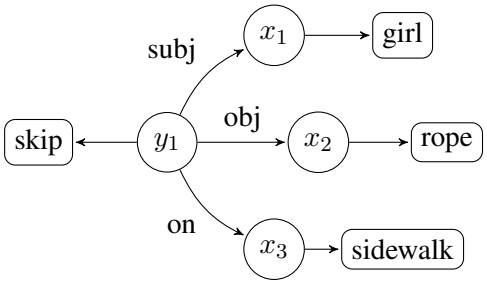


Figure 1: A graph for the basic formula (2).

verb and modifiers are linked to events by semantic roles, and the entire sentence is closed by existential quantification over events. For example, (1) is mapped onto (2).

(1) A girl is skipping rope on a sidewalk.

(2) $\exists x_1 \exists x_2 \exists x_3 \exists y_1 \, (\mathbf{girl}(x_1) \wedge \mathbf{rope}(x_2) \wedge \mathbf{sidewalk}(x_3) \wedge \mathbf{skip}(y_1) \wedge (\mathbf{subj}(y_1) = x_1) \wedge (\mathbf{obj}(y_1) = x_2) \wedge \mathbf{on}(y_1, x_3))$

We use x_i as a variable for entities and y_j for events. In this semantics, we represent all content words (e.g., *girl* and *skip*) as one-place predicates. Regarding functional words, we represent a preposition like *on* as a two-place predicate, e.g., $\mathbf{on}(y_1, x_3)$. We also use a small set of semantic roles such as $\mathbf{subj}$ and $\mathbf{obj}$ as a functional term and use equality ($=$) to connect an event and its participant, as in $\mathbf{subj}(y_1) = x_1$.

To be precise, the set of *atomic formulas* $\mathcal{A}$ in this event semantics is defined by the rule

$$\mathcal{A} ::= \mathbf{F}(t) \mid \mathbf{G}(t, u) \mid t = u$$

where $\mathbf{F}(t)$ is a one-place predicate (for content words), $\mathbf{G}(t, u)$ is a two-place predicate (for prepositions), t and u are a term. A term is defined as a constant, a variable, or a functional term of the form $f(t)$ where f is a semantic role and t is a term.

We call a formula constructed by conjunctions and existential quantifiers a *basic formula* in event semantics. Thus, a set of basic formulas φ in event semantics is defined as:

$$\varphi ::= \mathcal{A} \mid \varphi \wedge \varphi \mid \exists t \, \varphi$$

The formula in (2) is an instance of a basic formula, which captures the predicate-argument structure of a sentence.

On top of the system of basic formulas, we have a full language of event semantics with negation ($\neg$), disjunction ($\vee$), implication ($\rightarrow$), and a universal quantifier ($\forall$). These operators are used to represent additional logical features.

There is a natural correspondence between basic formulas and directed acyclic graphs (DAGs). Figure 1 shows an example[1]. In the graph representation, constants and variables correspond to vertices; both two-place predicates for prepositions (e.g., $\mathbf{on}(y_1, x_1)$) and functional terms for semantic roles (e.g., $\mathbf{subj}(y_1) = x_1$) are represented as edges. A one-place predicate $\mathbf{F}(t)$ in a logical formula can be represented as a functional relation $\mathbf{isa}(t, \mathbf{F})$, where $\mathbf{isa}$ is an expression relating a term t and a predicate $\mathbf{F}$ represented as a vertex. The $\mathbf{isa}$ edges are unlabeled for simplicity.

3.2 Natural deduction and word abduction

We use the system of natural deduction (Prawitz, 1965; Troelstra and Schwichtenberg, 2000) to capture phrase correspondences from a sentence pair (T, H), following the strategies for word axiom injection developed by Martínez-Gómez et al. (2017) and Yanaka et al. (2017). The sentence pair (T, H) is first mapped to a pair of formulas (T', H'). T' is initially set to the premise P, and H' is set to the goal G to be proved.

If formulas P and G are basic formulas, then the proving strategy is to decompose them into a set of atomic formulas using inference rules for conjunctions and existential quantifiers. The premise P is decomposed into a pool of premises $\mathcal{P} = \{\mathbf{p}_i(\theta_i) \mid i \in \{1, \ldots, m\}\}$, where each $\mathbf{p}_i(\theta_i)$ is an atomic formula and θ_i is a list of terms appearing in $\mathbf{p}_i(\theta_i)$. The goal G is also decomposed into a set of sub-goals $\mathcal{G} = \{\mathbf{g}_j(\theta'_j) \mid j \in \{1, \ldots, n\}\}$, where θ'_j is a list of terms appearing in $\mathbf{g}_j(\theta'_j)$.

The proof is performed by searching for a premise $\mathbf{p}_i(\theta_i)$ whose predicate matches that of a sub-goal $\mathbf{g}_j(\theta'_j)$. If such a premise is found, then variables in θ'_j are unified to those in θ_i and the sub-goal $\mathbf{g}_j(\theta'_j)$ can be removed from $\mathcal{G}$. If all the sub-goals can be removed, we prove $T' \rightarrow H'$. In the presence of two or more variables with the same predicate, there might be multiple possible variable unifications. Modern theorem provers explore these multiple possibilities in search of a configuration that proves a theorem.

Sub-goals may remain unproved when T logically does not entail H i.e., when there are no premise predicates $\mathbf{p}_i$ that are matched with $\mathbf{g}_j$. In this case, the system tries word axiom injection, called *word abduction*. More specifically, if there

[1] See Jones (2016) for some variants of graphical representations of logical formulas.

is a premise $\mathbf{p}_i(\theta_i)$ whose predicate has a linguistic relation (according to linguistic knowledge[2]) with that of a sub-goal $\mathbf{g}_j(\theta'_j)$, then variables in θ'_j are unified with those in θ_i and the sub-goal $\mathbf{g}_j(\theta'_j)$ can be removed from $\mathcal{G}$.

3.3 Graph illustration

Figure 2 shows an example to illustrate how the system works. To begin with, the input sentence pair (T, H) is mapped onto a pair of formulas, (T', H'). T' is initially placed to the premise P, and H' to the goal G. Note that these are basic formulas, and they are thus decomposed to the following sets of formulas $\mathcal{P}$ and $\mathcal{G}$, respectively:

$\mathcal{P} = \{\mathbf{lady}(x_1), \mathbf{meat}(x_2), \mathbf{cut}(y_1), \mathbf{up}(y_1),$
$\mathbf{precisely}(y_1), \mathbf{subj}(y_1)=x_1, \mathbf{obj}(y_1)=x_2\}$
$\mathcal{G} = \{\mathbf{woman}(x_3), \mathbf{meat}(x_4), \mathbf{cut}(y_2), \mathbf{piece}(x_5),$
$\mathbf{into}(y_2, x_5), \mathbf{subj}(y_2)=x_3, \mathbf{obj}(y_2)=x_4\}$

Steps 1 to 3 in Figure 2 demonstrate the variable unification routine and word axiom injection using graphs. Note that in step 1, all variables in formulas in $\mathcal{P}$ or $\mathcal{G}$ are initially different.

In step 2, we run a theorem proving mechanism that uses graph terminal vertices as anchors to unify variables between formulas in $\mathcal{P}$ and those in $\mathcal{G}$. The premise $\mathbf{meat}(x_2)$ in $\mathcal{P}$ matches the predicate $\mathbf{meat}$ of the sub-goal $\mathbf{meat}(x_4)$ in $\mathcal{G}$ and the variable unification $x_4 := x_2$ is applied (and similarly for the sub-goal $\mathbf{cut}(y_2)$ in $\mathcal{G}$ with the variable unification $y_2 := y_1$).

In step 3, we use the previous variable unification on y_1, the $\mathbf{subj}$ edge in $\mathcal{P}$ and $\mathcal{G}$ and the axiom $\forall x.\mathbf{lady}(x) \rightarrow \mathbf{woman}(x)$ from external knowledge to infer that $x_3 := x_1$.

4 Phrase Abduction

There is one critical reason that the word-to-word axiom injection described in Section 3.2 fails to detect phrase-to-phrase correspondences. That is, the natural deduction mechanism decomposes the goal G into atomic sub-goals that are then proved *one-by-one* (word-by-word), independently of each other except for the variable unification effect. This mechanism is particularly problematic when we attempt to prove phrases that resist decomposition, two-place predicates (e.g., $\mathbf{into}(x, y)$), or failures in variable unification (e.g., due to inaccurate semantics). Thus, we propose a method to detect phrase-to-phrase correspondence through natural deduction proofs.

[2]As given in a linguistic ontology or database such as WordNet or VerbOcean.

4.1 Phrase pair detection

We detect phrase-to-phrase entailing relations between T' and H' by finding alignments between the subgraphs of their meaning representations when $T' \Rightarrow H'$ or $T' \Rightarrow \neg H'$ hold. Finding subgraph alignments is a generalization of the subgraph isomorphism problem, which is NP-complete[3]. In this paper, we approximate a solution to this problem by using a combination of a backtracking variable unification and a deterministic graph search on the neighborhood of non-unified variables.

Using our running example in Figure 2, step 4 displays our proposed subgraph alignment. The variable x_5 in the graph of $\mathcal{G}$ cannot be unified with any variable in the graph of $\mathcal{P}$. This is a very common case in natural language inferences, as there might be concepts in H that are not directly supported by concepts in T. In this research, we propose spanning a subgraph starting at non-unified variables (e.g., x_5 in $\mathcal{G}$) whose boundaries are semantic roles (e.g., $\mathbf{subj}$, $\mathbf{obj}$). Its candidate semantics from $\mathcal{P}$ are then the attributes of its corresponding unified variables from $\mathcal{G}$ (e.g. *cut up precisely* $\rightarrow$ *cut into pieces*).

4.2 Graph alignments

To formalize this solution we introduce some graph notation. Let $V = \mathcal{V}^u \cup \mathcal{V}^{\bar{u}} \cup \mathcal{L}$ be the set of vertices, where $\mathcal{V}^u$ is the set of unified variables (e.g. x_1, x_2, y_1), $\mathcal{V}^{\bar{u}}$ is the set of non-unified variables (e.g. x_5), and $\mathcal{L}$ is a set of predicates (e.g., *lady*, *woman*). Let E be the set of labeled, directed edges $\langle v, l, v' \rangle$ where $v, v' \in V$ and l are labels that may represent a functional relation $\mathbf{isa}$, a preposition or a semantic role. We denote a set of two-place predicates for prepositions as PREP and a set of functional terms for semantic roles as ARGS; e.g., ARGS $= \{\mathbf{subj}, \mathbf{obj}\}$. A graph that represents $\mathcal{P}$ is then a tuple $G_\mathcal{P} = \langle V_\mathcal{P}, E_\mathcal{P} \rangle$, and similarly, for $\mathcal{G}$, $G_\mathcal{G} = \langle V_\mathcal{G}, E_\mathcal{G} \rangle$.

We can now define a function to span a subgraph in the neighborhood of non-unified variables $v \in \mathcal{V}^{\bar{u}}_\mathcal{G}$ in the graph of $\mathcal{G}$. We call a connected set of edges in which no semantic roles appear, i.e., $\{\langle v, l, v' \rangle \mid l \notin \text{ARGS}\}$, a *phrase set*. Let $E(x)$ be the phrase set in E such that each vertex is connected to x with an incoming or outgoing edge, that is, $E(x) = \{(v_i, l, v_k) \in E \mid (x = v_i \vee x = v_k) \wedge l \notin \text{ARGS}\}$.

[3]Emmert-Streib et al. (2016) gives a good overview.

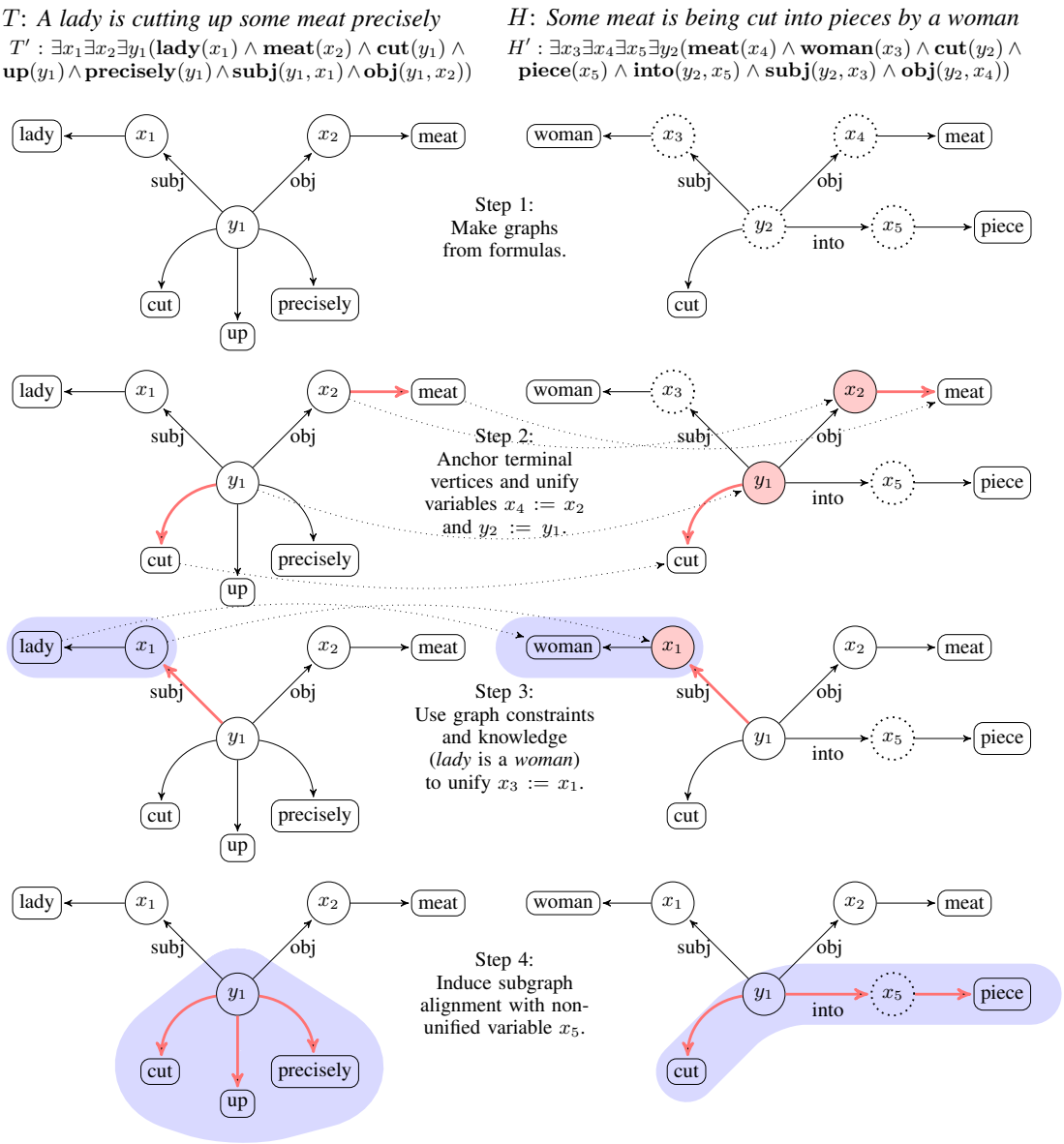


Figure 2: A graph representation of a theorem proving routine on basic formulas and variable unification. Dotted circles represent non-unified variables at each step, whereas edges without labels are attributes. The graph of the left side is the set of premises $\mathcal{P}$ and the graph of the right side is the set of sub-goals $\mathcal{G}$. Colored subgraphs represent a word or a phrase to which our axiom injection mechanism applies.

Note that $E(x)$ induces a subgraph in a given graph G and the condition $l \notin$ ARGS sets the boundaries of the subgraph by excluding the semantic roles of verb phrases. Given two phrase sets E and E', we say E' is reachable from E, written $E \sim E'$, if E and E' share at least one variable vertex. Let $\sim^*$ be the transitive closure of $\sim$. Given a set of edges $E_\mathcal{G}$ and a variable v, we define the *extended phrase set*, written Reach(v), as follows:

$$\text{Reach}(v) = \{e \in E \mid E_\mathcal{G}(v) \sim^* E\}$$

that is, the set of edges e that can be reached from v without crossing an edge with a semantic role label. This function defines a partition or equivalence class for non-unified variables $v \in \mathcal{V}_\mathcal{G}^u$, and each of these partitions induce a (possibly discontinuous) phrase in $\mathcal{G}$ that remains unproved.

The corresponding subgraph in $\mathcal{P}$ to each of these partitions is given by the vertices and edges connected with a path of length one to the unified variables that appear in Reach(v). That is,

$$\text{Corr}(v) = \{e \in E_\mathcal{P}(v'), v' \in V_\mathcal{G}^{[v]} \cap V_\mathcal{P}\}$$

where $V_\mathcal{G}^{[v]}$ denotes the vertices in the subgraph of $\mathcal{G}$ induced by the partition Reach(v).

A subgraph alignment between $\mathcal{P}$ and $\mathcal{G}$ is given by the pair of $\langle \text{Corr}(v), \text{Reach}(v) \rangle$ for all $v \in \mathcal{V}_{\mathcal{G}}^{\bar{u}}$, where the phrases can be read from the predicates in the vertices and edges labeled with prepositions.

We define a mapping $(\cdot)^\bullet$ from a labeled edge $\langle v, l, v' \rangle$ to an atomic formula as follows.

$$\langle v, l, v' \rangle^\bullet = \begin{cases} v'(v) & \text{if } l \text{ is } \mathbf{isa} \\ l(v, v') & \text{if } l \in \text{PREP} \\ l(v) = v' & \text{if } l \in \text{ARGS} \end{cases}$$

Let E be a set of labeled edges, and let $E^\bullet$ be $\{\langle v, l, v' \rangle^\bullet \mid \langle v, l, v' \rangle \in E\}$. The phrase axiom generated for each non-unified variable $v \in \mathcal{V}_{\mathcal{G}}^{\bar{u}}$ is defined as

$$\forall \theta_{\text{C}}. (\bigwedge \text{Corr}(v)^\bullet \rightarrow \exists \theta_{\text{R}}. (\bigwedge \text{Reach}(v)^\bullet)),$$

where θ_{C} is a set of free variables appearing in $\text{Corr}(v)^\bullet$ (which includes v) and θ_{R} is a set of free variables appearing in $\text{Reach}(v)^\bullet$ but not in $\text{Corr}(v)^\bullet$.

In Figure 2, the only non-unified variable in the sub-goal in step 4 is x_5, that is, $\mathcal{V}_{\mathcal{G}}^{\bar{u}} = \{x_5\}$. Then, starting from the variable x_5, $\text{Reach}(x_5)$ is

$$\{\langle y_1, \mathbf{into}, x_5 \rangle, \langle x_5, \mathbf{isa}, \mathbf{piece} \rangle\}.$$

Now $V_{\mathcal{G}}^{[x_5]} = \{y_1, x_5\}$, and thus $\text{Corr}(x_5)$ is

$$\{\langle y_1, \mathbf{isa}, \mathbf{cut} \rangle, \langle y_1, \mathbf{isa}, \mathbf{up} \rangle, \langle y_1, \mathbf{isa}, \mathbf{precisely} \rangle\}.$$

Finally, the following is the axiom generated from $\langle \text{Corr}(x_5), \text{Reach}(x_5) \rangle$[4].

$$\forall y_1 (\mathbf{cut}(y_1) \wedge \mathbf{up}(y_1) \wedge \mathbf{precisely}(y_1) \rightarrow \exists x_5 (\mathbf{into}(y_1, x_5) \wedge \mathbf{piece}(x_5))).$$

4.3 Non-basic formulas

If formulas P and G are not basic formulas (i.e., they contain logical operators other than $\wedge$ and $\exists$), they are decomposed according to inference rules of natural deduction. There are two types of inference rules: introduction rules decompose a goal formula into smaller sub-goals, and elimination rules decompose a formula in the pool of premises into smaller ones. Figure 3 shows introduction rules and elimination rules for decomposing non-basic formulas including negation, disjunction, implication, and a universal quantifier. By applying inference rules, a proof of non-basic formulas appearing in sub-goals can be decomposed to a set of subproofs that only have basic formulas in sub-goals. If a universal quantifier appears in premises, it is treated in the same way as other premises.

[4]Note that this axiom is logically equivalent to

$$\forall y_1 (\mathbf{cut}(y_1) \wedge \mathbf{up}(y_1) \wedge \mathbf{precisely}(y_1) \rightarrow \exists x_5 (\mathbf{cut}(y_1) \wedge \mathbf{into}(y_1, x_5) \wedge \mathbf{piece}(x_5)))$$

indicated in the colored subgraphs in step 4 of Figure 2.

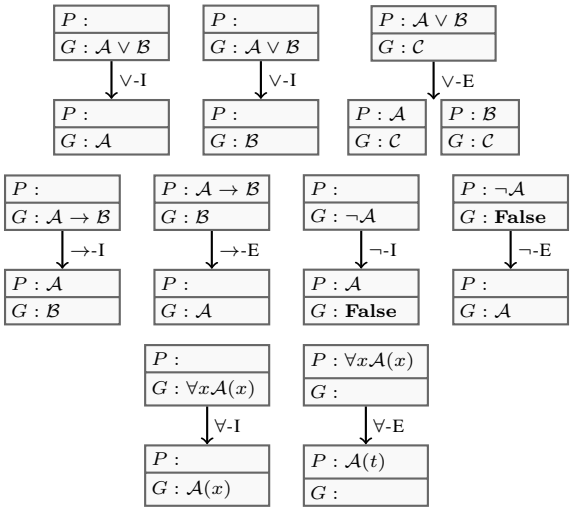


Figure 3: Inference rules used for decomposing non-basic formulas. P is a premise and G is a sub-goal. The initial formulas are at the top, with the formulas obtained by applying the inference rules shown below.

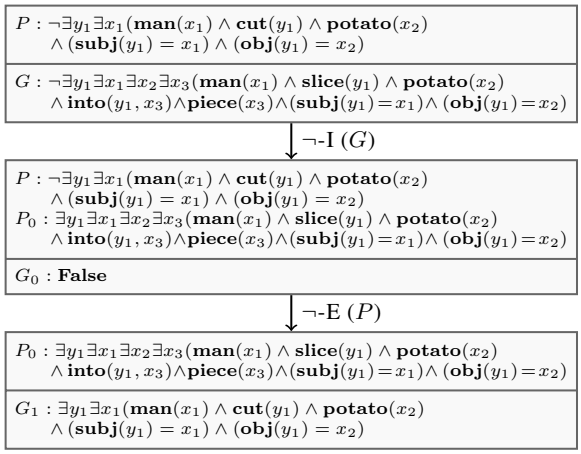


Figure 4: Proof process for the contradiction.

For example, consider the following sentence pair with the gold label "no" (contradiction):

T: *A man is not cutting a potato*

H: *A man is slicing a potato into pieces*

Figure 4 shows the proof process of $T' \Rightarrow \neg H'$. To prove the contradiction, the formulas T' and $\neg H'$ are set to P and G, respectively. Then, the negation in G is removed by applying the introduction rule $(\neg\text{-}I)$ to G. Here, **False** is the propositional constant denoting the contradiction. In the second stage of the proof, the goal is to prove **False** in G_0 from the two premises P and P_0. By applying $(\neg\text{-}E)$ to P, we can eliminate the negation from P, resulting in the new goal G_1.

As both the premise P_0 and the sub-goal G_1 are

basic formulas, the procedure described in the previous sections applies to the pair (P_0, G_1); these basic formulas are decomposed into atomic ones, and then the word-to-word abduction generates the desired axiom $\forall y_1(\mathbf{cut}(y_1) \rightarrow \mathbf{slice}(y_1))$. Finally, the graph alignment applies in the same way as described in Figure 2, which generates the phrase axiom:

$$\forall y_1(\mathbf{cut}(y_1) \rightarrow \exists x_5(\mathbf{into}(y_1, x_5) \wedge \mathbf{piece}(x_5)))$$

Using this axiom, one can complete the proof of the contradiction between T' and H'.

5 Experiments

5.1 Dataset selection

We use the SemEval-2014 version of the SICK dataset (Marelli et al., 2014) for evaluation. The SICK dataset is a dataset for semantic textual similarity (STS) as well as for RTE. It was originally designed for evaluating compositional distributional semantics, so it contains logically challenging problems involving quantifiers, negation, conjunction, and disjunction, as well as inferences with lexical and phrasal knowledge.

The SNLI dataset (Bowman et al., 2015) contains inference problems requiring phrasal knowledge. However, it is not concerned with logically challenging expressions; the semantic relationships between a premise and a hypothesis are often limited to synonym/hyponym lexical substitution, replacements of short phrases, or exact word matching. This is because hypotheses are often parallel to the premise in structures and vocabularies. The FraCaS dataset (Cooper et al., 1994) also contains logically complex problems. However, it is confined to purely logical inferences and thus does not contain problems requiring inferences with lexical and phrasal knowledge. For these reasons, we choose the SICK dataset to evaluate our method of using logical inference to extract phrasal knowledge.

The SICK dataset contains 9927 sentence pairs with a 5000/4927 training/test split. These sentence pairs are manually annotated with three types of labels *yes* (entailment), *no* (contradiction), or *unknown* (neutral) (see Table 1 for examples). In RTE tasks, we need to consider a directional relation between words such as hypernym and hyponym to prove entailment and contradiction. Hence, to extract phrasal knowledge for RTE tasks, we use the training data whose gold label is *entailment* or *contradiction*, excluding those with the *neutral* label.

5.2 Experimental setup

For the natural deduction proofs, we used ccg2lambda (Martínez-Gómez et al., 2016)[5], a higher-order automatic inference system, which converts CCG derivation trees into semantic representations and conducts natural deduction proofs automatically. We parsed the tokenized sentences of the premises and hypotheses using three wide-coverage CCG parsers: C&C (Clark and Curran, 2007), EasyCCG (Lewis and Steedman, 2014), and depccg (Yoshikawa et al., 2017). CCG derivation trees (parses) were converted into logical semantic representations based on Neo-Davidsonian event semantics (Section 3.1). The validation of semantic templates used for semantic representations was conducted exclusively on the trial split of the SICK dataset. We used Coq (Bertot and Castran, 2010), an interactive natural deduction theorem prover that we run fully automatically with a number of built-in theorem-proving routines called tactics, which include first-order logic.

We compare phrase abduction with different experimental conditions. **No axioms** is our system without axiom injection. **W2W** is the previous strategy of word abduction (Martínez-Gómez et al., 2017). **P2P** is our strategy of phrase abduction; **W2W+P2P** combines phrase abduction with word abduction. In addition, we compare our system with three purely logic-based (unsupervised) approaches: **The Meaning Factory** (Bjerva et al., 2014), **LangPro** (Abzianidze, 2015), and **UTexas** (Beltagy et al., 2014). We also compare our system with machine learning-based approaches: the current state-of-the-art deep learning model **GRU** (Yin and Schütze, 2017), a log-linear regression model **SemEval-2014 best** (Lai and Hockenmaier, 2014), and a hybrid approach combining a logistic regression model and probabilistic logic **PL+eclassif** (Beltagy et al., 2016).

5.3 Extracted paraphrases

We extracted 9445 axioms from the SICK training dataset. The proving time average to extract phrasal axioms was only 3.0 seconds for a one-sentence pair[6]. Table 2 shows some examples of

[5] Available at https://github.com/mynlp/ccg2lambda.

[6] Ours is a polynomial-time instance of the graph matching problem, where the vertex cover set (maximum number of variables in a phrase) is bounded to a small constant.

ID	Text	Hypothesis	Entailment
3941	*A boy is looking at a calendar*	*There is nobody checking a calendar*	No
5938	*Vegetables are being put into a pot by a man*	*Someone is pouring ingredients into a pot*	Yes
5930	*The man is not doing exercises*	*Two men are fighting*	Unknown

Table 1: Examples in the SICK dataset with different entailment labels and similarity scores.

Kind	Text	Hypothesis
noun phrase	***A blond woman*** *is sitting on the roof of a yellow vehicle, and two people are inside*	***A woman with blond hair*** *is sitting on the roof of a yellow vehicle, and two people are inside*
verb phrase	*The person is* ***setting fire to*** *the cameras*	*Some cameras are being* ***burned*** *by a person* ***with a blow torch***
verb phrase	*A man and a woman are* ***walking*** *together* ***through the woods***	*A man and a woman are hiking* ***through a wooded area***
prepositional phrase	*A child, who is small, is outdoors climbing steps outdoors* ***in an area full of grass***	*A small child is outdoors climbing steps* ***in a grassy area***
antonym	*A woman is* ***putting make-up on***	*The woman is* ***removing make-up***

Table 2: Examples of phrase alignments constructed by phrasal axiom injection.

	Prec.	Rec.	Acc.
GRU	–	–	87.1
PL+eclassif	–	–	85.1
SemEval2014 Best Score	81.6	81.9	84.6
The Meaning Factory	93.6	60.6	81.6
LangPro	98.0	58.1	81.4
UTexas	–	–	80.4
W2W+P2P	84.2	77.3	84.3
W2W	97.1	63.6	83.1
P2P	85.6	72.1	83.0
No axioms	98.9	46.5	76.7

Table 3: RTE results on the SICK dataset.

paraphrases we extracted from the natural deduction proof in the training set. In particular, the examples of verb phrases show our method has the potential to capture long paraphrases. Each paraphrase in Table 2 is not contained in WordNet and PPDB. There are many instances of non-contiguous phrases in the SICK dataset, in particular, verb-particle phrases. Shown in Table 2, our semantic alignment can detect non-contiguous phrases through the variable unification process, which is one of the main advantages over other shallow/syntactic methods. In addition, Table 2 shows our method is not limited to hypernym or hyponym relations, but it is also capable for detecting antonym phrases.

5.4 RTE evaluation results

Table 3 shows the experimental results. The results show that the combination of word abduction and phrase abduction improved the accuracy. When the **W2W+P2P** result is substituted for the **W2W** result, there is a 1.1% increase in accuracy (from 83.1% to 84.3%). The accuracy of **P2P** is almost equal to that of **W2W**. This is because the recall improves from 63.6% to 72.1% while the precision decreases from 97.1% to 85.6%. The increase in false positive cases caused this result; some details of false positive cases are described in the next subsection. **W2W+P2P** outperformed other purely logic-based systems. The machine learning-based approaches outperform **W2W+P2P**, but unlike these approaches, parameter estimation is not used in our method. This suggests that our method has the potential to increase the accuracy by using a classifier.

5.5 Positive examples and error analysis

Table 4 shows some positive and negative examples on RTE with the SICK dataset. For ID 9491, the sentence pair requires the paraphrase from *a field of brown grass* to *a grassy area*, not included in previous lexical knowledges. Our phrasal axiom injection can correctly generate this paraphrase from a natural deduction proof, and the system proves the entailment relation.

ID 2367 is also a positive example of phrasal axiom injection. The phrasal axiom between *set fire to cameras* and *burn cameras with a blow torch* was generated. This example shows that our semantic alignment succeeds in acquiring a general paraphrase by separating logical expressions such as *some* from content words and also by accounting for syntactic structures such as the passive-active alternation.

For ID 3628, the axiom shown in the table was extracted from the following sentence pair with

ID	Sentence Pair	Gold	Pred	Axiom
9491	*A group of four brown dogs are playing in a field of brown grass* *Four dogs are playing in a grassy area*	Yes	Yes	$\forall x_1(\mathbf{field}(x_1) \wedge \mathbf{brown}(x_1) \wedge \mathbf{grass}(x_1)$ $\rightarrow \mathbf{grassy}(x_1) \wedge \mathbf{area}(x_1))$
2367	*A person is burning some cameras with a blow torch* *The person is setting fire to the cameras*	Yes	Yes	$\forall x_1 \forall y_1(\mathbf{burn}(y_1) \wedge \mathbf{with}(y_1, x_1) \wedge \mathbf{blow\_torch}(x_1) \wedge \mathbf{camera}(\mathbf{obj}(y_1))$ $\rightarrow \mathbf{set}(y_1) \wedge \mathbf{fire}(\mathbf{obj}(y_1)) \wedge \mathbf{to}(y_1, \mathbf{obj}(y_1)) \wedge \mathbf{camera}(\mathbf{obj}(y_1)))$
3628	*A pan is being dropped over the meat* *The meat is being dropped into a pan*	Unk	Yes	$\forall y_1(\mathbf{pan}(\mathbf{obj}(y_1)) \rightarrow \mathbf{into}(y_1, \mathbf{obj}(y_1)))$
96	*A man is jumping into an empty pool* *There is no biker jumping in the air*	Unk	No	$\forall y_1(\mathbf{jump}(y_1) \rightarrow \exists x_1(\mathbf{in}(y_1, x_1) \wedge \mathbf{air}(x_1)))$ $\forall y_1(\mathbf{man}(y_1) \rightarrow \mathbf{biker}(y_1))$
408	*A group of explorers is walking through the grass* *Some people are walking*	Yes	Unk	

Table 4: Positive and negative examples on RTE from the SICK dataset.

their *entailment* label:

T_1: *A woman is putting meat in a pan*

H_1: *Someone is dropping the meat into a pan*

But the phrase *drop over* does not entail the phrase *drop into*, and a proof for the inference is over-generated in ID 3628. We extracted all possible phrasal axioms from the training dataset, so noisy axioms can be extracted as a consequence of multiple factors such as parsing errors or potential disambiguation in the training dataset. One possible solution for decreasing such noisy axioms would be to use additive composition models (Tian et al., 2016) and asymmetric learnable scoring functions to calculate the confidence on these asymmetric entailing relations between phrases.

ID 96 is also an example of over-generation of axioms. The first axiom, $\forall y_1(\mathbf{jump}(y_1) \rightarrow \exists x_1(\mathbf{in}(y_1, x_1) \wedge \mathbf{air}(x_1)))$ was extracted from the proof of $T_1 \Rightarrow H_1$:

T_1: *A child in a red outfit is jumping on a trampoline*

H_1: *A little boy in red clothes is jumping in the air*

The second axiom $\forall y_1(\mathbf{man}(y_1) \rightarrow \mathbf{biker}(y_1))$ was extracted from the proof of $T_2 \Rightarrow H_2$:

T_2: *A man on a yellow sport bike is doing a wheelie and a friend on a black bike is catching up*

H_2: *A biker on a yellow sport bike is doing a wheelie and a friend on a black bike is catching up*

Although these axioms play a role in the proofs of $T_1 \Rightarrow H_1$ and $T_2 \Rightarrow H_2$, the wrong axiom $\forall y_1(\mathbf{man}(y_1) \rightarrow \mathbf{biker}(y_1))$ causes the over-generation of a proof for the inference in ID 96. The correct one would rather be $\forall x_1 \forall y_1(\mathbf{man}(y_1) \wedge \mathbf{on}(y_1, x_1) \wedge \mathbf{bike}(x_1) \rightarrow \mathbf{biker}(y_1))$. In this case, it is necessary to bundle predicates in a noun-phrase by specifying the types of a variable (entity or event) when making phrase alignments.

For ID 408, the word *explorer* is not contained in the training entailment dataset and hence the relevant axiom $\forall x_1(\mathbf{explorer}(x_1) \rightarrow \mathbf{people}(x_1))$ was not generated. While our logic-based method enables detecting semantic phrase correspondences in a sentence pair in an unsupervised way, our next step is to predict unseen paraphrases of this type.

6 Conclusion

In this paper, we proposed a method of detecting phrase correspondences through natural deduction proofs of semantic relations between sentence pairs. The key idea is to attempt a proof with automatic phrasal axiom injection by the careful management of variable sharing during the proof construction process. Our method identifies semantic phrase alignments by monitoring the proof of a theorem and detecting unproved sub-goals and logical premises. The method of detecting semantic phrase alignments would be applicable to other semantic parsing formalisms and meaning representation languages such as abstract meaning representations (AMR) (Banarescu et al., 2013). Experiment results showed that our method detected various phrase alignments including non-contiguous phrases and antonym phrases. This result may contribute to previous phrase alignment approaches. The extracted phrasal axioms improved the accuracy of RTE tasks.

In future work, we shall enhance this methodology of phrasal axiom injection to predict unseen paraphrases. The pairs of premises and sub-goals that can be detected through the proof process conduct semantic alignments in a sentence pair. With the use of an additive composition model of distributional vectors, we can evaluate the validity of such semantic alignments. A combination of our phrasal axiom injection and additive composition model of distributional vectors has the potential to detect unseen paraphrases in a sentence pair.

Acknowledgments

We thank the three anonymous reviewers for their detailed comments. This work was supported by JST CREST Grant Number JPMJCR1301 and AIP Challenge Program, Japan.

References

Lasha Abzianidze. 2015. A tableau prover for natural logic and language. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 2492–2502.

Lasha Abzianidze. 2016. Natural solution to FraCaS entailment problems. In *Proceedings of the 5th Joint Conference on Lexical and Computational Semantics*. Association for Computational Linguistics, Berlin, Germany, pages 64–74.

Yuki Arase and Jun'ichi Tsujii. 2017. Monolingual phrase alignment on parse forests. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 1–11.

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

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 597–604.

Islam Beltagy, Cuong Chau, Gemma Boleda, Dan Garrette, Katrin Erk, and Raymond Mooney. 2013. Montague meets Markov: Deep semantics with probabilistic logical form. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics (*Sem-2013)*. Atlanta, GA, pages 11–21.

Islam Beltagy, Stephen Roller, Gemma Boleda, Katrin Erk, and Raymond J. Mooney. 2014. UTexas: Natural language semantics using distributional semantics and probabilistic logic. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval-2014)*. Association for Computational Linguistics and Dublin City University, Dublin, Ireland, pages 796–801.

Islam Beltagy, Stephen Roller, Pengxiang Cheng, Katrin Erk, and Raymond J. Mooney. 2016. Representing meaning with a combination of logical and distributional models. *Computational Linguistics* 42(4):763–808.

Yves Bertot and Pierre Castran. 2010. *Interactive Theorem Proving and Program Development: Coq'Art The Calculus of Inductive Constructions*. Springer Publishing Company, Incorporated, New York, USA.

Johannes Bjerva, Johan Bos, Rob van der Goot, and Malvina Nissim. 2014. The Meaning Factory: Formal semantics for recognizing textual entailment and determining semantic similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval-2014)*. Association for Computational Linguistics and Dublin City University, Dublin, Ireland, pages 642–646.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 632–642.

Timothy Chklovski and Patrick Pantel. 2004. Verbocean: Mining the web for fine-grained semantic verb relations. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Barcelona, Spain, pages 33–40.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics* 33(4):493–552.

Robin Cooper, Richard Crouch, Jan van Eijck, Chris Fox, Josef van Genabith, Jan Jaspers, Hans Kamp, Manfred Pinkal, Massimo Poesio, Stephen Pulman, et al. 1994. FraCaS–a framework for computational semantics. *Deliverable* D6.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. *Recognizing Textual Entailment: Models and Applications*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Frank Emmert-Streib, Matthias Dehmer, and Yongtang Shi. 2016. Fifty years of graph matching, network alignment and network comparison. *Information Sciences* 346-347(Supplement C):180 – 197.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologie*. Association for Computational Linguistics, Atlanta, Georgia, pages 758–764.

Bevan Keeley Jones. 2016. *Learning words and syntactic cues in highly ambiguous contexts*. Ph.D. thesis, The University of Edinburgh.

Alice Lai and Julia Hockenmaier. 2014. Illinois-lh: A denotational and distributional approach to semantics. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Association for Computational Linguistics and Dublin City University, Dublin, Ireland, pages 329–334.

Mike Lewis and Mark Steedman. 2014. A* CCG parsing with a supertag-factored model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Doha, Qatar, pages 990–1000.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman M. Sadeh, and Noah A. Smith. 2015. Toward abstractive summarization using semantic representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. The Association for Computational Linguistics, pages 1077–1086.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the 9th International Conference on Language Resources and Evaluation*. European Language Resources Association, Reykjavik, Iceland, pages 216–223.

Pascual Martínez-Gómez, Koji Mineshima, Yusuke Miyao, and Daisuke Bekki. 2016. ccg2lambda: A compositional semantics system. In *Proceedings of the 2016 System Demonstrations of the Association for Computational Linguistics*. Association for Computational Linguistics, Berlin, Germany, pages 85–90.

Pascual Martínez-Gómez, Koji Mineshima, Yusuke Miyao, and Daisuke Bekki. 2017. On-demand injection of lexical knowledge for recognising textual entailment. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Valencia, Spain, pages 710–720.

George A. Miller. 1995. WordNet: A lexical database for English. *Communications of the ACM* 38(11):39–41.

Koji Mineshima, Pascual Martínez-Gómez, Yusuke Miyao, and Daisuke Bekki. 2015. Higher-order logical inference with compositional semantics. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 2055–2061.

Koji Mineshima, Ribeka Tanaka, Pascual Martínez-Gómez, Yusuke Miyao, and Daisuke Bekki. 2016. Building compositional semantics and higher-order inference system for a wide-coverage japanese CCG parser. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 2236–2242.

Terence Parsons. 1990. *Events in The Semantics of English: a Study in Subatomic Semantics*. MIT Press, Cambridge, USA.

Dag Prawitz. 1965. *Natural Deduction – A Proof-Theoretical Study*. Almqvist & Wiksell, Stockholm, Sweden.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics* 2:377–392.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming Dependency Structures to Logical Forms for Semantic Parsing. *Transactions of the Association for Computational Linguistics* 4:127–140.

Ran Tian, Naoaki Okazaki, and Kentaro Inui. 2016. Learning semantically and additively compositional distributional representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Berlin, Germany, pages 1277–1287.

Anne Troelstra and Helmut Schwichtenberg. 2000. *Basic Proof Theory*. Cambridge University Press.

Hitomi Yanaka, Koji Mineshima, Pascual Martínez-Gómez, and Daisuke Bekki. 2017. Determining semantic textual similarity using natural deduction proofs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 692–702.

Wenpeng Yin and Hinrich Schütze. 2017. Task-specific attentive pooling of phrase alignments contributes to sentence matching. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, Valencia, Spain, pages 699–709.

Masashi Yoshikawa, Hiroshi Noji, and Yuji Matsumoto. 2017. A* CCG parsing with a supertag and dependency factored model. In *Proceedings of the 55nd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Vancouver, Canada, pages 277–287.

Shiqi Zhao, Cheng Niu, Ming Zhou, Ting Liu, and Sheng Li. 2008. Combining multiple resources to improve SMT-based paraphrasing model. In *Proceedings of the 46rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Columbus, Ohio, pages 1021–1029.

Automatic Stance Detection Using End-to-End Memory Networks

Mitra Mohtarami[1], **Ramy Baly**[1], **James Glass**[1], **Preslav Nakov**[2]
Lluís Màrquez[3]*, **Alessandro Moschitti**[3]*
[1]MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, USA
[2]Qatar Computing Research Institute, HBKU, Doha, Qatar; [3]Amazon
{mitra,baly,glass}@csail.mit.edu
pnakov@hbku.edu.qa; {lluismv,amosch}@amazon.com

Abstract

We present an effective end-to-end memory network model that jointly (*i*) predicts whether a given document can be considered as relevant evidence for a given claim, and (*ii*) extracts snippets of evidence that can be used to reason about the factuality of the target claim. Our model combines the advantages of convolutional and recurrent neural networks as part of a memory network. We further introduce a similarity matrix at the inference level of the memory network in order to extract snippets of evidence for input claims more accurately. Our experiments on a public benchmark dataset, FakeNewsChallenge, demonstrate the effectiveness of our approach.

1 Introduction

Recently, an unprecedented amount of false information has been flooding the Internet with aims ranging from affecting individual people's beliefs and decisions (Mihaylov et al., 2015; Mihaylov and Nakov, 2016) to influencing major events such as political elections (Vosoughi et al., 2018). Consequently, manual fact checking has emerged with the promise to support accurate and unbiased analysis of rumors spreading in social medias, as well as of claims made by public figures or news media.

As manual fact checking is a very tedious task, automatic fact checking has been proposed as a possible alternative. This is often broken into intermediate steps in order to alleviate the task complexity. One such step is *stance detection*, which is also useful for human experts as a stand-alone task. The task aims to identify the relative perspective of a piece of text with respect to a claim, typically modeled using labels such as *agree*, *disagree*, *discuss*, and *unrelated*; Figure 1 gives some examples.

Claim: Robert Plant Ripped up $800M Led Zeppelin Reunion Contract.	
Stance	**Snippet**
agree	Led Zeppelin's Robert Plant turned down £500m to reform supergroup...
disagree	Robert Plant's publicist has described as "rubbish" a Daily Mirror report that he rejected a £500m Led Zeppelin reunion...
discuss	Robert Plant reportedly tore up an $800 million Led Zeppelin reunion deal...
unrelated	Richard Branson's Virgin Galactic is set to launch SpaceShipTwo today...

Figure 1: Examples of snippets of text and their stance with respect to the target claim.

Here, we address the problem of stance detection using a novel model based on end-to-end memory networks (Sukhbaatar et al., 2015), which incorporates convolutional and recurrent neural networks, as well as a similarity matrix. Our model jointly addresses the problems of predicting the stance of a text with respect to a given claim, and of extracting relevant text snippets as support for the prediction of the model. We further introduce a similarity matrix, which we use at inference time in order to improve the extraction of relevant snippets.

The experimental results on the Fake News Challenge benchmark dataset show that our model, which is very feature-light, performs close to the state of the art. Our contributions can be summarized as follows: (*i*) We apply a novel memory network model enhanced with CNN and LSTM networks for stance detection. (*ii*) We further propose a novel extension of the general architecture based on a similarity matrix, which we use at inference time, and we show that this extension offers sizable performance gains. (*iii*) Finally, we show that our model is capable of extracting meaningful snippets from a given text document, which is useful not only for stance detection, but more importantly can support human experts who need to decide on the factuality of a given claim.

*Work conducted while these authors were at QCRI.

Proceedings of NAACL-HLT 2018, pages 767–776
New Orleans, Louisiana, June 1 - 6, 2018.

2 Stance Detection Memory Networks

Long-term memory is necessary to determine the stance of a long document with respect to a claim, as relevant parts of a document—paragraphs or text snippets—can indicate the perspective of a document with respect to a claim. Memory networks have been designed to remember past information (Sukhbaatar et al., 2015) and they can be particularly well-suited for stance detection since they can use a variety of inference strategies alongside their memory component.

In this section, we present a novel memory network for stance detection. It contains a new inference component that incorporates a similarity matrix to extract, with better accuracy, textual snippets that are relevant to the input claims.

2.1 Overview of the network

A memory network is a 5-tuple $\{M, I, G, O, R\}$, where the *memory* M is a sequence of objects or representations, the *input* I is a component that maps the input to its representation, the *generalization* component G (Sukhbaatar et al., 2015) updates the memory with respect to new input, the *output* O generates an output for each new input and the current memory state, and finally, the *response* R converts the output into a desired response format, e.g., a textual response or an action. These components can potentially use many different machine learning models.

Our new memory network for stance detection is a 6-tuple $\{M, I, F, G, O, R\}$, where F represents the new *inference* component. It takes an input document d as evidence and a textual statement s as a claim and converts them into their corresponding representations in the input I. Then, it passes them to the memory M. Next, the relevant parts of the input are identified in F, and afterwards they are used by G to update the memory. Finally, O generates an output from the updated memory, and converts it to a desired response format with R. The network architecture is depicted in Figure 2. We describe the components below.

2.2 Input Representation Component

The input to the stance detection algorithm is a document d as evidence and a textual statement s as a claim, (see lines 2 and 3 in Table 1). Each d is segmented into paragraphs x_j of varied lengths, where each x_j is considered as a piece of evidence for stance detection.

1	**Inputs:**
2	(1) A document (d) as a set of pieces of evidence $\{x_j\}$
3	(2) A textual statement containing a claim (s)
4	**Outputs:**
5	(1) predicting the relative perspective (or stance) of (d, s) to a claim as *agree*, *disagree*, *discuss*, *unrelated*.
6	*Inference outputs:*
7	(2) Top K evidence pieces x_j with their similarity scores
8	(3) Top K snippets of x_j with their similarity scores
9	**Memory Network Model:**
10	*1. Input memory representation (I):*
11	$d \rightarrow (X, W, E)$
12	$(X, W, E) \xrightarrow{TimeDistributed(LSTM)} \{m_1, ..., m_n\}$
13	$(X, W, E) \xrightarrow{TimeDistributed(CNN)} \{c_1, .., c_n\}$
14	$s \xrightarrow{LSTM,CNN} s_{lstm}, s_{cnn}$
15	*2. Memory (M), updating memory (G) and inference (F):*
16	$m_j = m_j \odot P^j_{tfidf}, \forall j$
17	$P^j_{lstm} = {s_{lstm}}^\intercal \times \mathbf{M} \times m_j, \forall j$
18	$c_j = c_j \odot P^j_{lstm}, \forall j$
19	$P^j_{cnn} = {s_{cnn}}^\intercal \times \mathbf{M}' \times c_j, \forall j$
20	*3. Output memory representation (O):* $o = \Big[\text{mean}(\{c_j\});$
21	$\big[\max(\{P^j_{cnn}\}); \text{mean}(\{P^j_{cnn}\})\big]; \big[\max(\{P^j_{lstm}\});$ $\text{mean}(\{P^j_{lstm}\})\big]; \big[\max(\{P^j_{tfidf}\}); \text{mean}(\{P^j_{tfidf}\})\big]\Big]$
22	*4. Generating the final prediction (R):*
23	$[o; s_{lstm}; s_{cnn}] \xrightarrow{MLP} \delta$
24	*5. Inference (F) outputs:*
25	$P^j_{cnn} \longrightarrow \{$*a set of evidence*$\} + \{$*similarity scores*$\}$
26	$M' \longrightarrow \{$*snippets*$\} + \{$*similarity scores*$\}$

Table 1: Summary of our memory network algorithm for stance detection.

Indeed, a paragraph usually presents a coherent argument, unified under one or more inter-related topics. The input component in our model converts each d into a set of pieces of evidence in a three dimensional (3D) tensor space as shown below (see line 11 in Table 1):

$$d = (X, W, E) \tag{1}$$

where $X = \{x_1, ..., x_n\}$ is a set of paragraphs considered as pieces of evidence; each x_j is represented by a set of words $W = \{w_1, ..., w_v\}$—drawn from a global vocabulary of size v—and a set of neural representations $E = \{e_1, ..., e_v\}$ for words in W. This 3D space is illustrated as a cube in Figure 2.

Each x_j is encoded from the 3D space into a semantic representation at the input component using a Long Short-Term Memory (LSTM) network. The lower left component in Figure 2 shows our LSTM network, which operates on our input as follows (see also line 12 in Table 1):

$$(X, W, E) \xrightarrow{TimeDistributed(LSTM)} \{m_1, ..., m_n\} \tag{2}$$

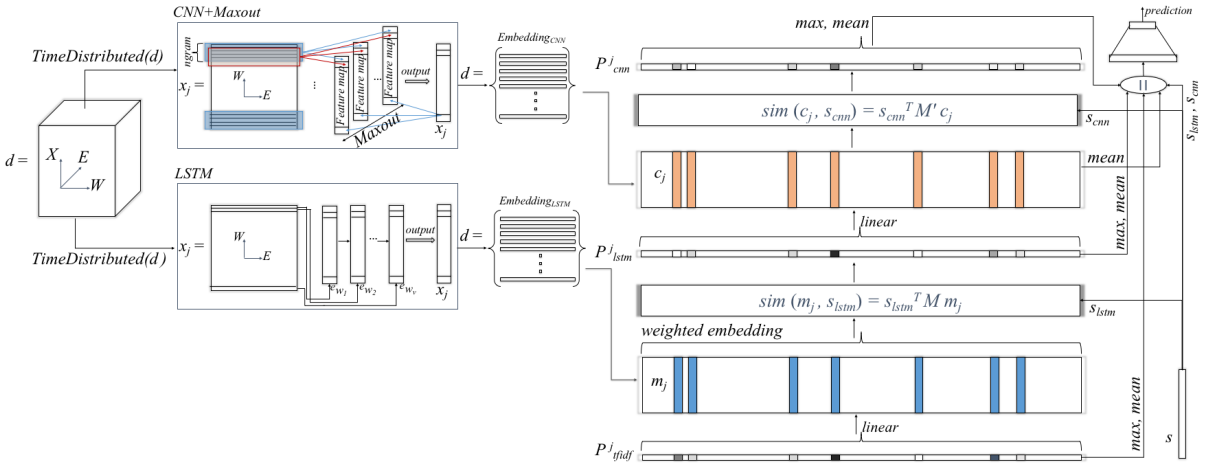


Figure 2: The architecture of our memory network model for stance detection.

where m_j is the LSTM representation of x_j, and *TimeDistributed()* indicates a wrapper that enables training the LSTM over all pieces of evidence by applying the same LSTM model to each time-step of a 3D input tensor, i.e., (X, W, E).

While LSTM networks were designed to effectively capture and memorize their inputs (Tan et al., 2016), Convolutional Neural Networks (CNNs) emphasize the local interaction between the individual words in the input word sequence, which is important for obtaining an effective representation. Here, we use a CNN in order to encode each x_j into its representation c_j as shown below (see line 13 in Table 1).

$$(X, W, E) \xrightarrow{TimeDistributed(CNN)} \{c_1, .., c_n\} \quad (3)$$

As shown in the left-top corner of Figure 2, this representation is passed as a new input to the component M of our memory network model. Moreover, we keep track of the computed n-grams from the CNN so that we can use them later in the inference and in the response components (see sections 2.3 and 2.6). For this purpose, we use a Maxout layer (Goodfellow et al., 2013) to take the maximum across k affine feature maps computed by the CNN, i.e., pooling across channels.

Previous work investigated the combination of convolutional and recurrent representations, which were fed to the other network as input (Tan et al., 2016; Donahue et al., 2015; Zuo et al., 2015; Sainath et al., 2015). In contrast, we feed individual outputs into our memory network separately, and we let it decide which representation better helps the target task. We demonstrate the effectiveness of this choice in our experiments.

Furthermore, we convert each input claim s into its representation using the corresponding LSTM and CNN networks as follows:

$$s \xrightarrow{LSTM,CNN} s_{lstm}, s_{cnn} \quad (4)$$

where s_{lstm} and s_{cnn} are the representations of s computed using $LSTM$ and CNN networks, respectively. Note that these are separate networks with different parameters from those used to encode the pieces of evidence.

Lines 10–14 of Table 1 describe the above steps in representing I in our memory network. This component encodes each input document d into a set of pieces of evidence $\{x_j\}\forall j$: it computes LSTM and CNN representations, m_j and c_j, respectively, for each x_j, and LSTM and CNN representations, s_{lstm} and s_{cnn}, for each claim s.

2.3 Inference Component

The resulting representations can serve to compute semantic similarity between claims and pieces of evidence. We define the similarity P^j_{lstm} between s and x_j as follows (see line 17 in Table 1):

$$P^j_{lstm} = s_{lstm}{}^{\intercal} \times \mathbf{M} \times m_j, \forall j \quad (5)$$

where $s_{lstm} \in \mathbb{R}^q$ and $m_j \in \mathbb{R}^d$ are the LSTM representations of s and x_j, respectively, and $M \in \mathbb{R}^{q \times d}$ is a similarity matrix capturing their similarity. For this purpose, M maps s and x_j into the same space as shown in Figure 3. M is a set of $q \times d$ parameters of the network, which are optimized during the training.

In a similar fashion, we compute the similarity P^j_{cnn} between x_j and s using the CNN representations as follows (see line 19 in Table 1):

$$P^j_{cnn} = s_{cnn}{}^{\intercal} \times \mathbf{M'} \times c_j, \forall j \quad (6)$$

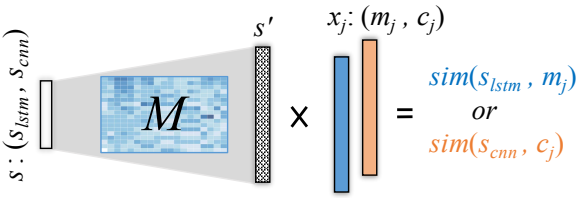


Figure 3: Matching a claim s and a piece of evidence x_j using a similarity matrix M. Here, s_{lstm} and s_{cnn} are the LSTM and CNN representations of s, whereas m_j and c_j are the LSTM and CNN representations of x_j.

where $s_{cnn} \in \mathbb{R}^{q'}$ and $c_j \in \mathbb{R}^{d'}$ are the representations of s and x_j obtained with CNN, respectively. The similarity matrix $M' \in \mathbb{R}^{q' \times d'}$ is a set of $q' \times d'$ parameters of the network and is optimized during the training. P^j_{lstm} and P^j_{cnn} indicate the claim-evidence similarity vectors computed based on the LSTM and on the CNN representations of s and x_j, respectively.

The rationale behind using the similarity matrix is that in our memory network model, as Figure 3 shows, we seek a transformation of the input claim such that $s' = M \times s$ in order to obtain the closest facts to the claim.

In fact, the relevant parts of the input document with respect to the input claim can be captured at a different level, e.g., using M' for the n-gram level or using the claim-evidence P^j_{lstm} or $P^j_{cnn}, \forall j$ at the paragraph level. We note that (*i*) P^j_{lstm} uses LSTM to take the word order and long-length dependencies into account, and (*ii*) P^j_{cnn} uses CNN to take n-grams and local dependencies into account, as explained in sections 2.2 and 2.3. Additionally, we compute another semantic similarity vector, P^j_{tfidf}, by applying a cosine similarity between the TF.IDF (Spärck Jones, 2004) representation of x_j and s. This is particularly useful for stance detection as it can help detect unrelated pieces of evidence.

2.4 Memory and Generalization Components

The information flow and updates in the memory is as follows: first, the representation vector $\{m_j\}\forall j$ is passed to the memory and updated using the claim-evidence similarity vector $\{P^j_{tfidf}\}$:

$$m_j = m_j \odot P^j_{tfidf}, \forall j \tag{7}$$

The reason for this weighting is to filter out most unrelated evidence with respect to the claim. The updated m_j in conjunction with s_{lstm} are used by the inference component–component F to compute $\{P^j_{lstm}\}$ as explained in Section 2.3.

Then, $\{P^j_{lstm}\}$ is used to update the new input set $\{c_j\}\forall j$ to the memory:

$$c_j = c_j \odot P^j_{lstm}, \forall j \tag{8}$$

Finally, the updated c_j in conjunction with s_{cnn} are used to compute P^j_{cnn} as explained in Sec. 2.3.

2.5 Output Representation Component

In memory networks, the memory output depends on the final goal, which, in our case, is to detect the relative perspective of a document to a claim. For this purpose, we apply the following equation:

$$o = \Big[\text{mean}(\{c_j\}); \big[\max(\{P^j_{cnn}\}); \text{mean}(\{P^j_{cnn}\})\big]; \big[\max(\{P^j_{lstm}\}); \text{mean}(\{P^j_{lstm}\})\big]; \big[\max(\{P^j_{tfidf}\}); \text{mean}(\{P^j_{tfidf}\})\big]\Big] \tag{9}$$

where $\text{mean}(\{c_j\})$ is the average vector of c_j representations. Furthermore, we compute the maximum and the average similarity between each piece of evidence and the claim using P^j_{tfidf}, P^j_{lstm} and P^j_{cnn}, which are computed for each evidence and claim in the inference component F. The maximum similarity identifies the part of document x_j that is most similar to the claim, while the average similarity measures the overall similarity between the document and the claim.

2.6 Response and Output Generation

This component computes the final stance of a document with respect to a claim. For this purpose, the concatenation of vectors o, s_{lstm} and s_{cnn} is fed into a Multi-Layer Perceptron (MLP), where a softmax predicts the stance of the document with respect to the claim, as shown below (see also lines 22–23 in Table 1):

$$[o; s_{lstm}; s_{cnn}] \xrightarrow{MLP} \delta \tag{10}$$

where δ is a softmax function. In addition to the resulting stance, we extract snippets from the input document that best indicate the perspective of the document with respect to the claim. For this purpose, we use P^j_{cnn} and M' as explained in Section 2.3 (see also lines 24–26 in Table 1).

The overall model is shown in Figure 2 and a summary of the model is presented in Table 1. All the model parameters, including those of (*i*) CNN and LSTM in I, (*ii*) the similarity matrices M and M' in F, and (*iii*) the MLP in R, are jointly learned during the training process.

3 Experiments and Evaluation

3.1 Data

We use the dataset provided by the Fake News Challenge,[1] where each example consists of a claim–document pair with the following possible relations between them: *agree* (the document agrees with the claim), *disagree* (the document disagrees with the claim), *discuss* (the document discusses the same topic as the claim, but does not take a stance with respect to the claim), *unrelated* (the document discusses a different topic than the topic of the claim). The data includes a total of 75.4K claim–document pairs, which link 2.5K unique articles with 2.5K unique claims, i.e., each claim is associated with 29.8 articles on average.

3.2 Settings

We use 100-dimensional word embeddings from GloVe (Pennington et al., 2014), which were trained on two billion tweets. We further use Adam as an optimizer and categorical cross-entropy as a loss. We use 100-dimensional units for the LSTM embeddings, and 100 feature maps with filter width of 5 for the CNN. We consider the first $p = 9$ paragraphs for each document, where p is the median of the number of paragraphs.

We optimize the hyper-parameters of the models using a validation dataset (20% of the training data). Finally, as the data is largely imbalanced towards the *unrelated* class, during training, we randomly select an equal number of instances from each class at each epoch.

3.3 Evaluation Measures

We use the following evaluation measures:

Accuracy: The number of correctly classified examples divided by the total number of examples. It is equivalent to micro-averaged F_1.

Macro-F1: We calculate F_1 for each class, and then we average across all classes.

Weighted Accuracy: This is a weighted, two-level scoring scheme, which is applied to each test example. First, if the example is from the *unrelated* class and the model correctly predicts it, the score is incremented by 0.25; otherwise, if the example is *related* and the model predicts *agree*, *disagree*, or *discuss*, the score is incremented by 0.25. Second, there is a further increment by 0.75 for each *related* example if the model predicts the correct label: *agree*, *disagree*, or *discuss*.

[1] Available at www.fakenewschallenge.org

Finally, the score is normalized by dividing it by the total number of test examples. The rationale behind this metric is that the binary *related/unrelated* classification task is expected to be much easier, while also being arguably less relevant to fake news detection than the stance detection task, which aims to further classify relevant instances as *agree*, *disagree*, or *discuss*. Therefore, the former task is given less weight and the latter task is given more weight through the weighted accuracy metric.

3.4 Baselines

Given the imbalanced nature of our data, we use two baselines in which we label all testing examples with the same label: (*i*) *unrelated* and (*ii*) *discuss*. The former is the majority class baseline, which is a reasonable baseline for *Accuracy* and *macro-F_1*, while the latter is a potentially better baseline for *Weighted Accuracy*.

We further use CNN and LSTM, and combinations thereof as baselines, since they form components of our model, and also because they yield state-of-the-art results for text, image, and video classification (Tan et al., 2016; Donahue et al., 2015; Zuo et al., 2015; Sainath et al., 2015).

Finally, we include the official baseline from the challenge, which is a Gradient Boosting classifier with word and n-gram overlap features, as well as indicators for refutation and polarity.

3.5 Our Models

sMemNN: This is our model presented in Figure 2. Note that unlike the CNN+LSTM and the LSTM+CNN baselines above, which feed the output of one network into the other one, the sMemNN model feeds the individual outputs of both the CNN and the LSTM networks into the memory network, and lets it decide how much to rely on each of them. This consideration also facilitates reasoning and explaining model predictions, as we will discuss in more detail below.

sMemNN (dotProduct): This is a version of sMemNN, where the similarity matrices are replaced by the dot product between the representation of the claims and of the evidence. For this purpose, we first project the claim representation to a dense layer that has the same size as the representation of each piece of evidence, and then we compute the dot product between the resulting representation and the representation of the evidence.

Methods	Total Parameters	Trainable Parameters	Weighted Accuracy	Macro-F1	Accuracy
1. All-*unrelated*	–	–	39.37	20.96	72.20
2. All-*discuss*	–	–	43.89	7.47	17.57
3. CNN	2.7M	188.7K	40.66	24.44	41.53
4. LSTM	2.8M	261.3K	57.23	37.23	60.21
5. CNN+LSTM	4.2M	361.5K	42.02	27.36	48.54
6. LSTM+CNN	2.8M	281.5K	60.21	40.33	65.36
7. Gradient Boosting	–	–	75.20	46.13	86.32
8. sMemNN (dotProduct)	5.4M	275.2K	75.13	50.21	83.85
9. sMemNN	5.5M	377.5K	78.97	56.75	87.27
10. sMemNN (with TF)	110M	105M	**81.23**	**56.88**	**88.57**

Table 2: Evaluation results on the test data.

sMemNN (with TF): Since our LSTM and CNN networks use a limited number of starting paragraphs[2] for an input document, we enrich our model with the BOW representation of documents and claims as well as their TF.IDF-based cosine similarity. We concatenate these vectors with the memory outputs (section 2.5) and pass them to the R component (section 2.6) of sMemNN. We expect these BOW vectors provide useful information that are not initially incorporated into the sMemNN model.

3.6 Results

Table 2 reports the performance of all models on the test dataset. The All-*unrelated* and the All-*discuss* baselines perform poorly across the evaluation measures, except for All-*unrelated*, which achieves high accuracy, which is due to *unrelated* being by far the dominant class in the dataset.

Next, we can see that the LSTM model consistently outperforms the CNN across all evaluation measures. Although the larger number of parameters of the LSTM can play a role, we believe that its superiority comes from it being able to remember previously observed relevant pieces of text.

Next, we see systematic improvements for the combinations of the CNN and the LSTM models: CNN+LSTM is better than CNN alone, and LSTM+CNN is better than LSTM alone. Better performance is achieved by the LSTM+CNN model, where claims and evidence are first processed by a LSTM and then fed into a CNN.

The Gradient Boosting model achieves sizable improvement over the above baseline neural models. However, we should note that these neural models do not have the rich hand-crafted features that were used in the Gradient Boosting model.

[2]Due to the long length of documents, it is impractical to consider all paragraphs when training LSTM and CNN.

Row 9 shows the results for our memory network model (sMemNN), which consistently outperforms all other baseline models across all evaluation metrics, achieving 10.62 and 3.77 points of absolute improvement in terms of Macro-F1 and Weighted Accuracy, respectively, over the best baseline (Gradient Boosting). We believe this is due to the memory network being able to capture good text snippets. As we will see below, these snippets are also useful for explaining the model's predictions. Comparing row 9 to row 8, we can see the importance of our proposed similarity matrix: replacing that matrix by a simple dot product hurts the performance of the model considerably across all evaluation measures, thus lowering it to the level of the Gradient Boosting model.

Finally, row 10 shows the results for our memory network model enriched by BOW representation. As we expected, it improves the performance of sMemNN - perhaps by capturing useful information from paragraphs beyond the starting few.

To put the results of sMemNN in perspective, we should mention that the best system at the Fake News Challenge (Baird et al., 2017) achieved a macro-F1 of 57.79, which is not significantly different from our results at the 0.05 significance level (p-value=0.53). Yet, they have an ensemble combining the feature-rich Gradient Boosting system with neural networks. In contrast, we only use raw text as input and no ensembles, and our main goal is to study a new memory network model and its explainability component.

Further analysis of the outputs (namely, the confusion matrices) of the different models we experimented with reveals the following general trends: (*i*) The *unrelated* examples are easy to detect, and most models show high performance for this class. (*ii*) The *agree* and the *disagree* examples are often misclassified as *discuss* by the baseline models.

Claim 1: man saved from bear attack - thanks to his justin bieber ringtone		
Evidence Id	$\mathbf{P^j_{cnn}}$	**Evidence Snippet**
2069-3	0.89	... fishing in the yakutia republic , russia , igor vorozhbitsyn is lucky to be alive after his justin bieber ringtone , baby , scared off a bear that was attacking him [0.41] ...
2069-7	1.0	... but as the bear clawed vorozhbitsyn ' s face and back his mobile phone rang , the ringtone selected was justin bieber ' s hit song baby . rightly startled [1.00] , the bear retreated back into [0.39] the forest ...
true label: *agree*; **predicted label:** *agree*		

Claim 2: 50ft crustacean , dubbed crabzilla , photographed lurking beneath the waters in whitstable		
Evidence Id	$\mathbf{P^j_{cnn}}$	**Evidence Snippet**
24835-1	0.0046	... a marine biologist has killed off claims [-0.0008] that a giant crab is [0.0033] living on the kent coast - insisting the image is probably a well - doctored hoax [0.0012] ...
24835-7	-0.0008	... i don ' t know what the currents are like around that harbour or what sort of they might produce in the sand , but i think it ' s more conceivable that someone is playing [0.0007] about with the photo ...
true label: *disagree*; **predicted label:** *disagree*		

Table 3: Examples of highly ranked snippets of evidence for an input claim, which are automatically extracted by our inference component. The P^j_{cnn} column and the values in the top-right corner of the highlighted snippets show the similarity between the claim and evidence, and between the claim and snippets of the evidence, respectively.

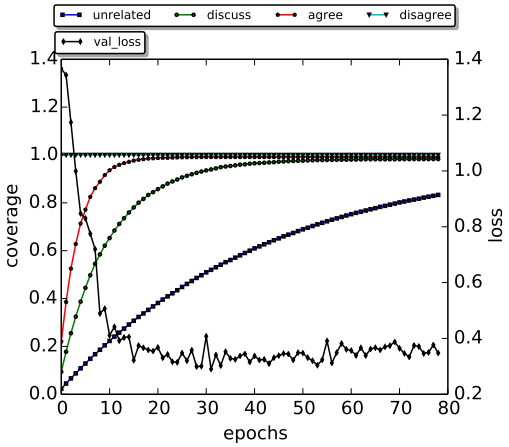


Figure 4: Effect of data coverage. The left y-axis shows the fraction of data observed during training (coverage), the right y-axis shows the loss during training.

This is mainly because the document that discusses a claim often shares the same topic with the claim, but then it does not take a stance with respect to that claim. (*iii*) The *disagree* examples are the most difficult ones for all models, probably because they represent by far the smallest class.

4 Discussion

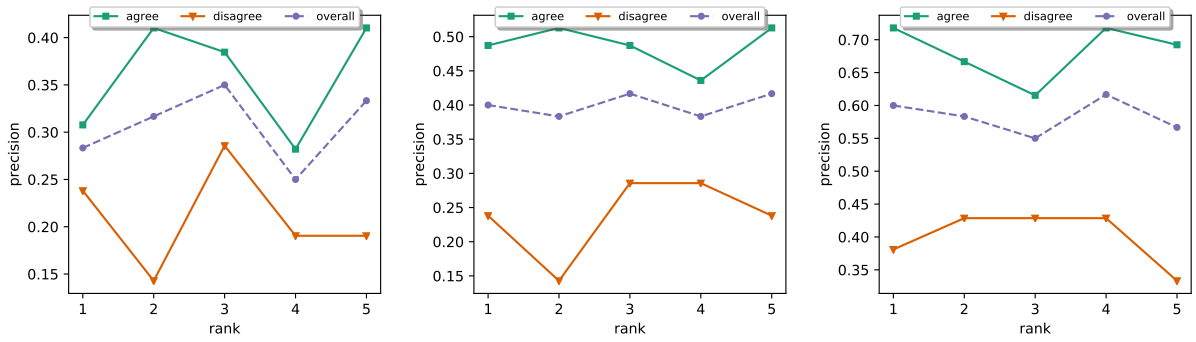
4.1 Training Data Coverage

As discussed previously, we balance the data at each training iteration by randomly selecting z instances from each of the four target classes, where z is the size of the class with the minimum number of training instances. Here, we investigate the amount of training data that gets actually used.

For this purpose, at each training iteration, we report the proportion of the training instances from each class that have been used for training so far, either at the current or at any of the previous iterations. As Figure 4 shows, our random data sampling procedure eventually covers almost all training instances. Since the *disagree* class is the smallest, its instances remain fully covered throughout the process. Moreover, almost all other related instances, i.e., *agree* and *discuss*, are observed during training, as well as a large fraction of the dominating *unrelated* examples. Note that the model achieves its best (lowest) loss on the validation dataset at iteration 31, when almost all *related* training instances are observed. This happens while the corresponding fraction for the *unrelated* pairs is around 50%, i.e., a considerable number of the *unrelated* instances are not required to be used for training.

4.2 Explainability

One of the main advantages of our memory network model, compared to the baselines and to related work in general, is that it has the capacity to explain its predictions by extracting snippets from each piece of evidence that supports its prediction. As we explained in Section 2.3, our inference component predicts the similarity between each piece of evidence x_j and the claim s at the n-grams level using the similarity matrix M' and the claim-evidence similarity vector P^j_{cnn}. Below, we explore our model's explainability in more detail.



(a) n-grams (b) consecutive n-grams (c) sentences

Figure 5: Prediction explainability. Sub-figures (a)-(c) show the precision of our model at explaining its predictions when the pieces of evidence are (a) fixed-length n-grams ($n = 5$), (b) combinations of several consecutive n-grams with similar scores, or (c) the entire sentence, if it includes at least one extracted n-gram snippet.

Table 3 shows examples of two claims and the snippets extracted as evidence. Column P^j_{cnn} shows the overall similarity between the evidence and the corresponding claim as computed by the inference component of our model. The highlighted texts are the snippets with the highest similarity to the claim as extracted by the same component. The values on the snippets' top-right show the claim-snippet similarity values obtained by the inference component.

Note that all snippets are fixed-length, namely 5-grams; however, in case there are several consecutive n-grams with similar scores, for better illustration, we combine them into a single snippet and we report their average values (see the snippet for evidence 2069-3). As these examples show, our model can accurately predict the stance of these pieces of evidence against their corresponding claims. Also, claim 2 and its corresponding evidence are shown at the second row of Table 3. As this example shows, the similarity values associated with snippets are either too small or negative, e.g., see the similarity value for the snippet "*biologist has killed off claims*." We can see that these help the model to make accurate predictions.

We conduct the following experiment to quantify the performance of our memory network at explaining its predictions: we randomly sample 100 *agree/disagree* claim–document examples from our gold data, and we manually evaluate the top five pieces of evidence that our model provides to support/oppose the corresponding claims. [3]

Figure 5(a) shows the precision of our memory network model at explaining its predictions when each supporting/opposing piece of evidence is an n-gram snippet of fixed length ($n = 5$) for the *agree* and the *disagree* classes, and their combinations at the top-k ranks, $k = \{1, \ldots, 5\}$. We can see in the figure that the model achieves precision of 0.28, 0.32, 0.35, 0.25, and 0.33 at ranks 1–5. Moreover, we find that it can accurately identify useful key phrases such as *officials declared the video*, *according to previous reports*, *believed will come*, *president in his tweets* as supporting pieces of evidence, and *proved a hoax*, *shot down a cnn report*, *would be skeptical* as opposing pieces of evidence.

Note that this relatively low precision of our memory network model at explaining its *agree/disagree* predictions is mainly due to the unsupervised nature of this task as no gold snippets justifying the document's gold stance with respect to the target claim are available in the Fake News Challenge dataset. [4]

Furthermore, our evaluation setup is at the n-gram level in Figure 5(a). However, if we conduct a more coarse-grained evaluation where we combine consecutive n-grams with similar scores into a single snippet, the precision for these new snippets will improve to 0.40, 0.38, 0.42, 0.38, and 0.42 at ranks 1–5, as Figure 5(b) shows. If we further extend the evaluation to the sentence level, the precision will jump to 0.60, 0.58, 0.55, 0.62, and 0.57 at ranks 1–5, as we can see in Figure 5(c).

[3] In 76 cases, our model correctly classified the *agree/disagree* examples when the evaluation was conducted, and it further provided arguably adequate snippets.

[4] Some other recent datasets, to be presented at this same HLT-NAACL'2018 conference, do have such gold evidence annotations (Baly et al., 2018; Thorne et al., 2018).

5 Related Work

While stance detection is an interesting task in its own right, e.g., for media monitoring, it is also an important component for fact checking and veracity inference.[5] Automatic fact checking was envisioned by Vlachos and Riedel (2014) as a multi-step process that (*i*) identifies check-worthy statements (Hassan et al., 2015; Gencheva et al., 2017; Jaradat et al., 2018), (*ii*) generates questions to be asked about these statements (Karadzhov et al., 2017), (*iii*) retrieves relevant information to create a knowledge base (Shiralkar et al., 2017), and (*iv*) infers the veracity of these statements, e.g., using text analysis (Castillo et al., 2011; Rashkin et al., 2017) or information from external sources (Mihaylova et al., 2018; Karadzhov et al., 2017; Popat et al., 2017).

There have been some nuances in the way researchers have defined the stance detection task. SemEval-2016 Task 6 (Mohammad et al., 2016) targets stances with respect to some target proposition, e.g., entities, concepts or events, as *in-favor*, *against*, or *neither*. The winning model in the task was based on transfer learning: a Recurrent Neural Network trained on a large Twitter corpus was used to predict task-relevant hashtags and to initialize a second recurrent neural network trained on the provided dataset for stance prediction (Zarrella and Marsh, 2016). Subsequently, Zubiaga et al. (2016) detected the stance of tweets toward rumors and hot topics using linear-chain conditional random fields (CRFs) and tree CRFs that analyze tweets based on their position in tree-like conversational threads.

Most commonly, stance detection is defined with respect to a *claim*, e.g., as in the 2017 Fake News Challenge. The best system in the challenge was an ensemble of gradient-boosted decision trees with rich features (e.g., *sentiment*, *word2vec*, *singular value decomposition* (*SVD*) and *TF.IDF* features, etc.) and a deep convolutional neural network to address the stance detection problem (Baird et al., 2017).

Unlike the above work, we use a feature-light memory network that jointly infers the stance and highlights relevant snippets of evidence with respect to a given claim.

[5] Yet, stance detection and fact checking are typically supported by separate datasets. Two notable upcoming exceptions, both appearing in this HLT-NAACL'2018, are (Thorne et al., 2018) for English and (Baly et al., 2018) for Arabic.

6 Conclusion

We studied the problem of stance detection, which aims to predict whether a given document supports, challenges, or just discusses a given claim. The nature of the task requires a machine learning model to focus on the relevant paragraphs of the evidence. Moreover, in order to understand whether a paragraph supports a claim, there is a need to refer to information in other paragraphs. CNNs or LSTMs are not well-suited for this task as they cannot model complex dependencies such as semantic relationships with respect to entire previous paragraphs. In contrast, memory networks are exactly designed to remember previous information. However, given the large size of documents and paragraphs, basic memory networks do not handle well irrelevant and noisy information, which we confirmed in our experiments.

Thus, we proposed a novel extension of general memory networks based on a similarity matrix and a stance filtering component, which we apply at the inference level, and we have shown that this extension offers sizable performance gains making memory networks competitive. Moreover, our model can extract meaningful snippets from documents that can explain the stance of a given claim.

In future work, we plan to extend the inference component to select an optimal set of explanations for each prediction, and to explain the model as a whole, not only at the instance level.

Acknowledgment

This research was carried out in collaboration between the MIT Computer Science and Artificial Intelligence Laboratory (CSAIL) and the HBKU Qatar Computing Research Institute (QCRI).

References

Sean Baird, Doug Sibley, and Yuxi Pan. 2017. Talos targets disinformation with fake news challenge victory. https://blog.talosintelligence.com/2017/06/talos-fake-news-challenge.html.

Ramy Baly, Mitra Mohtarami, James Glass, Lluís Màrquez, Alessandro Moschitti, and Preslav Nakov. 2018. Integrating stance detection and fact checking in a unified corpus. In *Proceedings of HLT-NAACL*. New Orleans, LA, USA.

Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information credibility on Twitter. In *Proceedings of WWW*. Hyderabad, India, pages 675–684.

Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Trevor Darrell, and Kate Saenko. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of CVPR*. Boston, MA, USA, pages 2625–2634.

Pepa Gencheva, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, and Ivan Koychev. 2017. A context-aware approach for detecting worth-checking claims in political debates. In *Proceedings of RANLP*. Varna, Bulgaria, pages 267–276.

Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Maxout networks. In *Proceedings of ICML*. Atlanta, GA, USA, pages 1319–1327.

Naeemul Hassan, Chengkai Li, and Mark Tremayne. 2015. Detecting check-worthy factual claims in presidential debates. In *Proceedings CIKM*. Melbourne, Australia, pages 1835–1838.

Israa Jaradat, Pepa Gencheva, Alberto Barrón-Cedeño, Lluís Màrquez, and Preslav Nakov. 2018. ClaimRank: Detecting check-worthy claims in Arabic and English. In *Proceedings of HLT-NAACL*. New Orleans, LA, USA.

Georgi Karadzhov, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, and Ivan Koychev. 2017. Fully automated fact checking using external sources. In *Proceedings of RANLP*. Varna, Bulgaria, pages 344–353.

Todor Mihaylov, Georgi Georgiev, and Preslav Nakov. 2015. Finding opinion manipulation trolls in news community forums. In *Proceedings of CoNLL*. Beijing, China, pages 310–314.

Todor Mihaylov and Preslav Nakov. 2016. Hunting for troll comments in news community forums. In *Proceedings of ACL*. Berlin, Germany.

Tsvetomila Mihaylova, Preslav Nakov, Lluis Marquez, Alberto Barron-Cedeno, Mitra Mohtarami, Georgi Karadzhov, and James Glass. 2018. Fact checking in community forums. In *Proceedings of AAAI*. New Orleans, LA, USA.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiao-Dan Zhu, and Colin Cherry. 2016. SemEval-2016 task 6: Detecting stance in tweets. In *Proceedings of SemEval*. Berlin, Germany, pages 31–41.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of EMNLP*. Doha, Qatar, pages 1532–1543.

Kashyap Popat, Subhabrata Mukherjee, Jannik Strötgen, and Gerhard Weikum. 2017. Where the truth lies: Explaining the credibility of emerging claims on the web and social media. In *Proceedings of WWW*. Perth, Australia, pages 1003–1012.

Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. 2017. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of EMNLP*. Copenhagen, Denmark, pages 2931–2937.

Tara N Sainath, Oriol Vinyals, Andrew Senior, and Haşim Sak. 2015. Convolutional, long short-term memory, fully connected deep neural networks. In *Proceedings of ICASSP*. Brisbane, Australia, pages 4580–4584.

Prashant Shiralkar, Alessandro Flammini, Filippo Menczer, and Giovanni Luca Ciampaglia. 2017. Finding streams in knowledge graphs to support fact checking. In *Proceedings of ICDM*. New Orleans, LA, USA, pages 859–864.

Karen Spärck Jones. 2004. IDF term weighting and IR research lessons. *Journal of documentation* 60(5):521–523.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Proceedings of NIPS*. Montreal, Canada, pages 2440–2448.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved representation learning for question answer matching. In *Proceedings of ACL*. Berlin, Germany, pages 464–473.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: A large-scale dataset for fact extraction and VERification. In *Proceedings of HLT-NAACL*. New Orleans, LA, USA.

Andreas Vlachos and Sebastian Riedel. 2014. Fact checking: Task definition and dataset construction. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*. Baltimore, MD, USA, pages 18–22.

Soroush Vosoughi, Deb Roy, and Sinan Aral. 2018. The spread of true and false news online. *Science* 359(6380):1146–1151.

Guido Zarrella and Amy Marsh. 2016. MITRE at SemEval-2016 Task 6: Transfer learning for stance detection. In *Proceedings of SemEval*. San Diego, CA, USA, pages 458–463.

Arkaitz Zubiaga, Elena Kochkina, Maria Liakata, Rob Procter, and Michal Lukasik. 2016. Stance classification in rumours as a sequential task exploiting the tree structure of social media conversations. In *Proceedings of COLING*. Osaka, Japan, pages 2438–2448.

Zhen Zuo, Bing Shuai, Gang Wang, Xiao Liu, Xingxing Wang, Bing Wang, and Yushi Chen. 2015. Convolutional recurrent neural networks: Learning spatial dependencies for image representation. In *Proceedings of CVPR*. Boston, MA, USA, pages 18–26.

Collective Entity Disambiguation with Structured Gradient Tree Boosting

Yi Yang **Ozan Irsoy** **Kazi Shefaet Rahman**
Bloomberg LP
New York, NY 10022
{yyang464+oirsoy+krahman7}@bloomberg.net

Abstract

We present a gradient-tree-boosting-based structured learning model for jointly disambiguating named entities in a document. Gradient tree boosting is a widely used machine learning algorithm that underlies many top-performing natural language processing systems. Surprisingly, most works limit the use of gradient tree boosting as a tool for regular classification or regression problems, despite the structured nature of language. To the best of our knowledge, our work is the first one that employs the structured gradient tree boosting (SGTB) algorithm for collective entity disambiguation. By defining global features over previous disambiguation decisions and jointly modeling them with local features, our system is able to produce globally optimized entity assignments for mentions in a document. Exact inference is prohibitively expensive for our globally normalized model. To solve this problem, we propose Bidirectional Beam Search with Gold path (BiBSG), an approximate inference algorithm that is a variant of the standard beam search algorithm. BiBSG makes use of global information from both past and future to perform better local search. Experiments on standard benchmark datasets show that SGTB significantly improves upon published results. Specifically, SGTB outperforms the previous state-of-the-art neural system by near 1% absolute accuracy on the popular AIDA-CoNLL dataset.[1]

1 Introduction

Entity disambiguation (ED) refers to the process of linking an entity mention in a document to its corresponding entity record in a reference knowledge base (e.g., Wikipedia or Freebase). As a core information extraction task, ED plays an important role in the language understanding pipeline, underlying a variety of downstream applications such as relation extraction (Mintz et al., 2009; Riedel et al., 2010), knowledge base population (Ji and Grishman, 2011; Dredze et al., 2010), and question answering (Berant et al., 2013; Yih et al., 2015). This task is challenging because of the inherent ambiguity between mentions and the referred entities. Consider, for example, the mention 'Washington', which can be linked to a city, a state, a person, an university, or a lake (Fig. 1).

Fortunately, simple and effective features have been proposed to capture the ambiguity that are designed to model the similarity between a mention (and its local context) and a candidate entity, as well as the relatedness between entities that co-occur in a single document. These are typically statistical features estimated from entity-linked corpora, and similarity features that are pre-computed using distance metrics such as cosine. For example, a key feature for ED is the *prior probability* of an entity given a specific mention, which is estimated from mention-entity co-occurrence statistics. This simple feature alone can yield 70% to 80% accuracy on both news and Twitter texts (Lazic et al., 2015; Guo et al., 2013).

To capture the non-linear relationships between the low-dimensional dense features like statistical features, sophisticated machine learning models such as neural networks and gradient tree boosting are preferred over linear models. In particular, gradient tree boosting has been shown to be highly competitive for ED in recent work (Yang and Chang, 2015; Yamada et al., 2016). However, although achieving appealing results, existing gradient-tree-boosting-based ED systems typically operate on each individual mention, without attempting to jointly resolve entity mentions in a document together. Joint entity disambiguation has been shown to significantly boost performance when used in conjunction with other machine learning techniques (Ratinov et al., 2011; Hoffart et al., 2011). However, how to train a

[1] When ready, the code will be published at https://github.com/bloomberg/sgtb.

Proceedings of NAACL-HLT 2018, pages 777–786
New Orleans, Louisiana, June 1 - 6, 2018.

global gradient tree boosting model that produces coherent entity assignments for all the mentions in a document is still an open question.

In this work, we present, to the best of our knowledge, the first structured gradient tree boosting (SGTB) model for collective entity disambiguation. Building on the general SGTB framework introduced by Yang and Chang (2015), we develop a globally normalized model for ED that employs a conditional random field (CRF) objective (Lafferty et al., 2001). The model permits the utilization of global features defined between the current entity candidate and the entire decision history for previous entity assignments, which enables the global optimization for all the entity mentions in a document. As discussed in prior work (Smith and Johnson, 2007; Andor et al., 2016), globally normalized models are more expressive than locally normalized models.

As in many other global models, our SGTB model suffers from the difficulty of computing the partition function (normalization term) for training and inference. We adopt beam search to address this problem, in which we keep track of multiple hypotheses and sum over the paths in the beam. In particular, we propose Bidirectional Beam Search with Gold path (BiBSG) technique that is specifically designed for SGTB model training. Compared to standard beam search strategies, BiBSG reduces model variance and also enjoys the advantage in its ability to consider both past and future information when predicting an output.

Our contributions are:

- We propose a SGTB model for collectively disambiguating entities in a document. By jointly modeling local decisions and global structure, SGTB is able to produce globally optimal entity assignments for all the mentions.
- We present BiBSG, an efficient algorithm for approximate bidirectional inference. The algorithm is tailored to SGTB models, which can reduce model variance by generating more point-wise functional gradients for estimating the auxiliary regression models.
- SGTB achieves state-of-the-art (SOTA) results on various popular ED datasets, and it outperforms the previous SOTA systems by 1-2% absolute accuracy on the AIDA-CoNLL (Hoffart et al., 2011) dataset.

2 Model

In this section, we present a SGTB model for collective entity disambiguation. We first formally define the task of ED, and then describe a structured learning formalization for producing globally coherent entity assignments for mentions in a document. Finally, we show how to optimize the model using functional gradient descent.

For an input document, assume that we are given all the mentions of named entities within it. Also assume that we are given a lexicon that maps each mention to a set of entity candidates in a given reference entity database (e.g., Wikipedia or Freebase). The ED system maps each mention in the document to an entry in the entity database. Since a mention is often ambiguous on its own (i.e., the lexicon maps the mention to multiple entity candidates), the ED system needs to leverage two types of contextual information for disambiguation: local information based on the entity mention and its surrounding words, and global information that exploits the document-level coherence of the predicted entities. Note that modeling entity-entity coherence is very challenging, as the long-range dependencies between entities correspond to exponentially large search space.

We formalize this task as a structured learning problem. Let $\mathbf{x}$ be a document with T target mentions, and $\mathbf{y} = \{y_t\}_{t=1}^{T}$ be the entity assignments of the mentions in the document. We use $S(\mathbf{x}, \mathbf{y})$ to denote the joint scoring function between the input document and the output structure. In traditional NLP tasks, such as part-of-speech tagging and named entity recognition, we often rely on low-order Markov assumptions to decompose the global scoring function into a summation of local functions. ED systems, however, are often required to model nonlocal phenomena, as any pair of entities is potentially interdependent. Therefore, we choose the following decomposition:

$$S(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{T} F(\mathbf{x}, y_t, \mathbf{y}_{1:t-1}), \qquad (1)$$

where $F(\mathbf{x}, y_t, \mathbf{y}_{1:t-1})$ is a factor scoring function. Specifically, a local prediction y_t depends on all the *previous decisions*, $\mathbf{y}_{1:t-1}$ in our model, which resembles recurrent neural network (RNN) models (Elman, 1990; Hochreiter and Schmidhuber, 1997).

We adopt a CRF loss objective, and define a

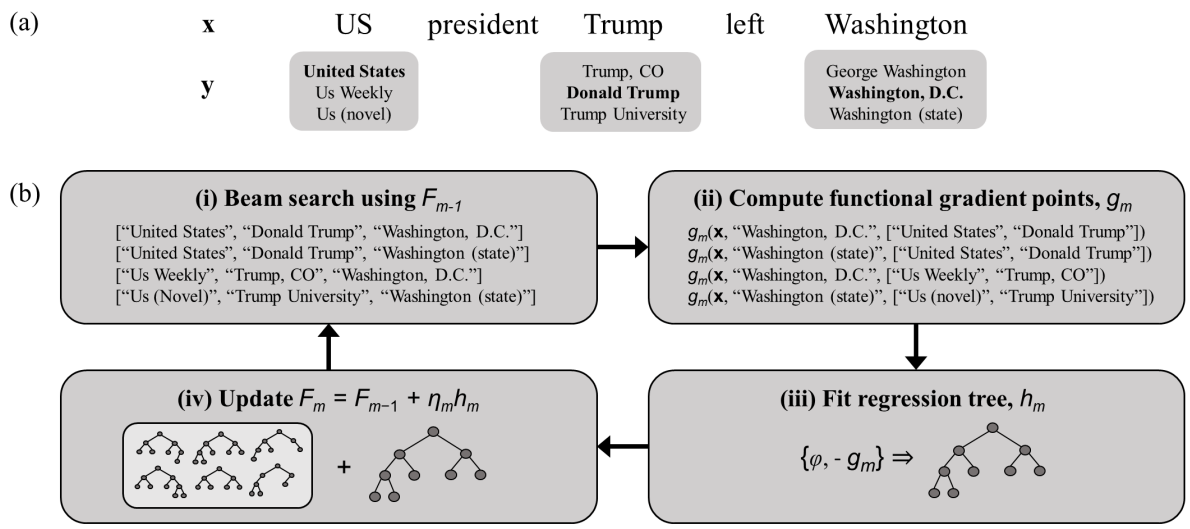


Figure 1: (a) Example document $\mathbf{x}$ with entity candidates for each mention (gold entities are in **bold**); (b) the m-th SGTB update iteration: (i) conduct beam search to sample candidate entity sequences (§ 3), (ii) compute point-wise functional gradients for each candidate sequence, (iii) fit a regression tree to the negative functional gradient points with input features, ϕ, (iv) update the factor scoring function, F, by adding the trained regression tree.

distribution over possible output structures as follows:

$$p(\mathbf{y}|\mathbf{x}) = \frac{\exp\{\sum_{t=1}^{T} F(\mathbf{x}, y_t, \mathbf{y}_{1:t-1})\}}{Z(\mathbf{x})}, \quad (2)$$

where

$$Z(\mathbf{x}) = \sum_{\mathbf{y}' \in Gen(\mathbf{x})} \exp\{\sum_{t=1}^{T} F(\mathbf{x}, y'_t, \mathbf{y}'_{1:t-1})\}$$

and $Gen(\mathbf{x})$ is the set of all possible sequences of entity assignments depending on the lexicon. $Z(\mathbf{x})$ is then a global normalization term. As shown in previous work, globally normalized models are very expressive, and also avoid the label bias problem (Lafferty et al., 2001; Andor et al., 2016). The inference problem is to find

$$\arg\max_{\mathbf{y} \in Gen(\mathbf{x})} p(\mathbf{y}|\mathbf{x}) = \arg\max_{\mathbf{y} \in Gen(\mathbf{x})} \sum_{t=1}^{T} F(\mathbf{x}, y_t, \mathbf{y}_{1:t-1}). \quad (3)$$

2.1 Structured gradient tree boosting

An overview of our SGTB model is shown in Fig. 1. The model minimizes the negative log-likelihood of the data,

$$\begin{aligned} L(\mathbf{y}^*, S(\mathbf{x}, \mathbf{y})) &= -\log p(\mathbf{y}^*|\mathbf{x}) \\ &= \log Z(\mathbf{x}) - S(\mathbf{x}, \mathbf{y}^*), \end{aligned} \quad (4)$$

where $\mathbf{y}^*$ is the gold output structure.

In a standard CRF, the factor scoring function is typically assumed to have this form: $F(\mathbf{x}, y_t, \mathbf{y}_{1:t-1}) = \theta^\top \phi(\mathbf{x}, y_t, \mathbf{y}_{1:t-1})$, where $\phi(\mathbf{x}, y_t, \mathbf{y}_{1:t-1})$ is the feature function and θ are the model parameters. The key idea of SGTB is that, instead of defining a parametric model and optimizing its parameters, we can directly optimize the factor scoring function $F(\cdot)$ iteratively by performing gradient descent in function space. In particular, suppose $F(\cdot) = F_{m-1}(\cdot)$ in the m-th iteration, we will update $F(\cdot)$ as follows:

$$\begin{aligned} F_m(\mathbf{x}, y_t, \mathbf{y}_{1:t-1}) = F_{m-1}(\mathbf{x}, y_t, \mathbf{y}_{1:t-1}) \\ - \eta_m g_m(\mathbf{x}, y_t, \mathbf{y}_{1:t-1}), \end{aligned} \quad (5)$$

where

$$\begin{aligned} g_m(\mathbf{x}, y_t, \mathbf{y}_{1:t-1}) &= \frac{\partial L(\mathbf{y}^*, S(\mathbf{x}, \mathbf{y}))}{\partial F(\mathbf{x}, y_t, \mathbf{y}_{1:t-1}))} \\ &= p(\mathbf{y}_{1:t}|\mathbf{x}) - \mathbf{1}[\mathbf{y}_{1:t} = \mathbf{y}^*_{1:t}] \end{aligned} \quad (6)$$

is the functional gradient, η_m is the learning rate, and $\mathbf{1}[\cdot]$ represents an indicator function, which returns 1 if the predicted sequence matches the gold one, and 0 otherwise. We initialize $F(\cdot)$ to 0 ($F_0(\cdot) = 0$).

We can approximate the negative functional gradient $-g_m(\cdot)$ with a regression tree model $h_m(\cdot)$ by fitting the training data $\{\phi(\mathbf{x}^{(i)}, y_t^{(i)}, \mathbf{y}_{1:t-1}^{(i)})\}$ to the point-wise negative functional gradients (also known as residuals) $\{-g_m(\mathbf{x}^{(i)}, y_t^{(i)}, \mathbf{y}_{1:t-1}^{(i)})\}$. Then the factor scoring

function can be obtained by

$$F(\mathbf{x}, y_t, \mathbf{y}_{1:t-1}) = \sum_{m=1}^{M} \eta_m h_m(\mathbf{x}, y_t, \mathbf{y}_{1:t-1}), \tag{7}$$

where $h_m(\mathbf{x}, y_t, \mathbf{y}_{1:t-1})$ is called a basis function. We set $\eta_m = 1$ in this work.

3 Training

Training the SGTB model requires computing the point-wise functional gradients with respect to training documents and candidate entity sequences. This is challenging, due to the exponential output structure search space. First, we are not able to enumerate all possible candidate entity sequences. Second, computing the conditional probabilities shown in Eq. 6 is intractable, as it is prohibitively expensive to compute the partition function $Z(\mathbf{x})$ in Eq. 2. Beam search can be used to address these problems. We can compute point-wise functional gradients for candidate entity sequences in the beam, and approximately compute the partition function by summing over the elements in the beam.

In this section, we present a bidirectional beam search training algorithm that always keeps the gold sequence in the beam. The algorithm is tailored to SGTB, and improves standard training methods in two aspects: (1) it reduces model variance by collecting more point-wise function gradients to train a regression tree; (2) it leverages information from both past and future to conduct better local search.

3.1 Beam search with gold path

The early update (Collins and Roark, 2004) and LaSO (Daumé III and Marcu, 2005; Xu and Fern, 2007) strategies are widely adopted with beam search for updating model parameters in previous work. Both methods keep track of the location of the gold path in the beam while decoding a training sequence. A gradient update step will be taken if the gold path falls out of the beam at a specific time step t or after the last step T. Adapting the strategies to SGTB training is straightforward. We will compute point-wise functional gradients for all candidate entity sequences after time step T or when the gold sequence falls out the beam. Both early update and LaSO are typically applied to online learning scenarios, in which model parameters are updated after passing one or a few training sequences.

SGTB training, however, fits the batch learning paradigm. In each training epoch, a SGTB model will be updated only once using the regression tree model fit on the point-wise negative functional gradients. The gradients are calculated with respect to the output sequences obtained from beam search. We propose a simple training strategy that computes and collects point-wise functional gradients at every step of a training sequence. In addition, instead of passively monitoring the gold path, we always keep the gold path in the beam to ensure that we have valid functional gradients at each time step. The new beam search training method, Beam Search with Gold path (BSG), generates much more point-wise functional gradients than early update or LaSO, which can reduce the variance of the auxiliary regression tree model. As a result, SGTB trained with BSG consistently outperforms early update or LaSO in our exploratory experiments, and it also requires fewer training epochs to converge.[2]

3.2 Bidirectional beam search

During beam search, if we consider a decision made at time step t, the joint probability $p(\mathbf{y}|\mathbf{x})$ can be factorized around t as follows:

$$\begin{aligned} p(\mathbf{y}|\mathbf{x}) = p(\mathbf{y}_{1:t-1}|\mathbf{x}) \cdot p(y_t|\mathbf{y}_{1:t-1}, \mathbf{x}) \\ \cdot p(\mathbf{y}_{t+1:T}|y_t, \mathbf{y}_{1:t-1}, \mathbf{x}). \end{aligned} \tag{8}$$

Traditional beam search performs inference in a unidirectional (left-to-right) fashion. Since the beam search at time step t considers only the beam sequences that were committed to so far, $\{\mathbf{y}_{1:t-1}\}$, it effectively approximates the above probability by assuming that all futures are equally likely, i.e. $p(\mathbf{y}_{t+1:T}|y_t, \mathbf{y}_{1:t-1}, \mathbf{x})$ is uniform. Therefore, at any given time, there is no information from the future when incorporating the global structure.

In this work, we adopt a Bidirectional Beam Search (BiBS) methodology that incorporates multiple beams to take future information into account (Sun et al., 2017). It makes two simplifying assumptions that better approximate the joint probability above while remaining tractable: (1) future predictions are independent of past predictions given y_t; (2) $p(y_t)$ is uniform. These yield the following approximation:

$$\begin{aligned} p(\mathbf{y}_{t+1:T}|y_t, \mathbf{y}_{1:t-1}, \mathbf{x}) = p(\mathbf{y}_{t+1:T}|y_t, \mathbf{x}) \\ \propto p(y_t|\mathbf{y}_{t+1:T}, \mathbf{x}) \cdot p(\mathbf{y}_{t+1:T}|\mathbf{x}). \end{aligned} \tag{9}$$

[2] Early update and LaSO perform similarly, thus we only report results for early update in § 5.

Substituting this back into Eq. 8 therefore yields:

$$p(\mathbf{y}|\mathbf{x}) \propto p(\mathbf{y}_{1:t-1}|\mathbf{x}) \cdot p(y_t|\mathbf{y}_{1:t-1}, \mathbf{x}) \cdot p(y_t|\mathbf{y}_{t+1:t}, \mathbf{x}) \cdot p(\mathbf{y}_{t+1:T}|\mathbf{x}), \quad (10)$$

which decomposes into multiplication of a forward probability and a backward probability. In (Sun et al., 2017), these are retrieved from forward and backward recurrent networks, whereas in our work we use the joint scores (log probabilities shown in Eq. 1) computed for partial sequences from forward and backward beams.

Algorithm 1: Bidirectional Beam Search with Gold path (BiBSG)

Input : input document $\mathbf{x}$, candidate sequences $\{\mathbf{y}\}$, joint scoring function $S(\mathbf{x}, \mathbf{y}_{t_1:t_2})$
Output: beam sequence set C
$C \leftarrow \emptyset$
while *not converged* **do**
  // forward beam search
  for $t = 1, \cdots, T$ **do**
    $C^{(F)} \leftarrow \text{top-B}_{\mathbf{y}_{1:t}}[S(\mathbf{x}, \mathbf{y}_{1:t}) + S(\mathbf{x}, \mathbf{y}_{T:t})]$
    // add gold subsequence
    $C^{(F)} \leftarrow C^{(F)} \cup \{\mathbf{y}^*_{1:t}\}$
    $C \leftarrow C \cup C^{(F)}$
  end
  // backward beam search
  for $t = T, \cdots, 1$ **do**
    $C^{(B)} \leftarrow \text{top-B}_{\mathbf{y}_{T:t}}[S(\mathbf{x}, \mathbf{y}_{T:t}) + S(\mathbf{x}, \mathbf{y}_{1:t})]$
    // add gold subsequence
    $C^{(B)} \leftarrow C^{(B)} \cup \{\mathbf{y}^*_{T:t}\}$
    $C \leftarrow C \cup C^{(B)}$
  end
end

The full inference algorithm, Bidirectional Beam Search with Gold path (BiBSG), is presented in Alg. 1. When performing the forward pass to update the forward beam, forward joint scores, $S(\mathbf{x}, \mathbf{y}_{1:t})$, are computed with respect to current forward beam, and backward joint scores, $S(\mathbf{x}, \mathbf{y}_{T:t})$, are computed with respect to previous backward beam. A similar procedure is used for the backward pass. The search converges very fast, and we use two rounds of bidirectional search as a good approximation. Finally, SGTB-BiBSG compares the conditional probabilities $p(\mathbf{y}^{(\cdot)}|\mathbf{x})$ of the best scoring output sequences $\mathbf{y}^{(\text{F})}$ and $\mathbf{y}^{(\text{B})}$ obtained from the forward and backward beams. The final prediction is the sequence with the higher conditional probability score.

4 Implementation

We provide implementation details of our SGTB systems, including entity candidate generation, adopted local and global features, and some efforts to make training and inference faster.

4.1 Candidate selection

We use a mention prior $\hat{p}(y|x)$ to select entity candidates for a mention x. Following Ganea and Hofmann (2017), the prior is computed by averaging mention prior probabilities built from mention-entity hyperlink statistics from Wikipedia[3] and a large Web corpus (Spitkovsky and Chang, 2012). Given a mention, we select the top 30 entity candidates according to $\hat{p}(y|x)$.

We also use a simple heuristic proposed by Ganea and Hofmann (2017) to improve candidate selection for persons: for a mention x, if there are mentions of persons that contain x as a continuous subsequence of words, then we consider the candidate set obtained from the longest mention for the mention x.

4.2 Features

The feature function $\phi(\mathbf{x}, y_t, \mathbf{y_{1:t-1}})$ can be decomposed into the summation of a local feature function $\phi_L(\mathbf{x}, y_t)$ and a global feature function $\phi_G(y_t, \mathbf{y}_{1:t-1})$.

Local features We consider standard local features that have been used in prior work, including mention priors $p(y|x)$ obtained from different resources; entity popularity features based on Wikipedia page view count statistics;[4] named entity recognition (NER) type features given by an in-house NER system trained on the CoNLL 2003 NER data (Tjong Kim Sang and De Meulder, 2003); entity type features based on Freebase type information; and three textual similarity features proposed by Yamada et al. (2016).[5]

Global features Three features are utilized to characterize entity-entity relationships: entity-entity co-occurrence counts obtained from Wikipedia, and two cosine similarity scores between entity vectors based on entity embeddings from (Ganea and Hofmann, 2017) and Freebase entity embeddings released by Google[6]

[3] We use a Wikipedia snapshot as of Feb. 2017.

[4] We obtain the statistics of Feb. 2017 and Dec. 2011 from https://dumps.wikimedia.org/other/pagecounts-ez/merged/ .

[5] We obtain embeddings jointly trained for words and entities from (Ganea and Hofmann, 2017).

[6] https://code.google.com/archive/p/word2vec/

respectively. We denote the entity-entity features between entities y_t and $y_{t'}$ as $\phi_E(y_t, y_{t'})$.

At step t of a training sequence, we quantify the coherence of y_t with respect to previous decisions $\mathbf{y}_{1:y-1}$ by first extracting entity-entity features between y_t and $y_{t'}$ where $1 \leq t' \leq t-1$, and then aggregating the information to have a global feature vector $\phi_G(y_t, \mathbf{y}_{1:t-1})$ of a fixed length:

$$\phi_G(y_t, \mathbf{y}_{1:t-1}) = \sum_{t'=1}^{t-1} \frac{\phi_E(y_t, y_{t'})}{t-1} \oplus \max_{t'=1}^{t-1} \phi_E(y_t, y_{t'}),$$

where $\oplus$ denotes concatenation of vectors.

4.3 Efficiency

Global models are powerful and effective, but often at a cost of efficiency. We discuss ways to speed up training and inference for SGTB models.

Many of the adopted features such as mention priors and entity-entity co-occurrences can be extracted once and retrieved later with just a hash map lookup. The most expensive features are the cosine similarity features based on word and entity embeddings. By normalizing the embeddings to have a unit norm, we can obtain the similarity features using dot products. We find this simple preprocessing makes feature extraction faster by two orders of magnitude.

SGTB training can be easily parallelized, as the computation of functional gradients are independent for different documents. During each training iteration, we randomly split training documents into different partitions, and then calculate the point-wise functional gradients for documents of different partitions in parallel.

5 Experiments

In this section, we evaluate SGTB on some of the most popular datasets for ED. After describing the experimental setup, we compare SGTB with previous state-of-the-art (SOTA) ED systems and present our main findings in § 5.3.

5.1 Data

We use six publicly available datasets to validate the effectiveness of SGTB. AIDA-CoNLL (Hoffart et al., 2011) is a widely adopted dataset for ED based on the CoNLL 2003 NER dataset (Tjong Kim Sang and De Meulder, 2003). It is further split into training (AIDA-train), development (AIDA-dev), and test (AIDA-test) sets.[7] AQUAINT (Milne and Witten, 2008), MSNBC (Cucerzan, 2007), and ACE (Ratinov et al., 2011) are three datasets for Wikification, which also contain Wikipedia concepts beyond named entities. These datasets were recently cleaned and updated by Guo and Barbosa (2016). WIKI and CWEB are automatically annotated datasets built from the ClueWeb and Wikipedia corpora by Guo and Barbosa (2016). The statistics of these datasets are available in Table 1.

Dataset	# mention	# doc	# mention per doc
AIDA-train	18,448	946	19.5
AIDA-dev	4,791	216	22.1
AIDA-test	4,485	231	19.4
AQUAINT	727	50	14.5
MSNBC	656	20	32.8
ACE	257	36	7.1
CWEB	11,154	320	34.8
WIKI	6,821	320	21.3

Table 1: Statistics of the ED datasets used in this work.

5.2 Experimental settings

Following previous work (Guo and Barbosa, 2016; Ganea and Hofmann, 2017), we evaluate our models on both *in-domain* and *cross-domain* testing settings. In particular, we train our models on AIDA-train set, tune hyperparameters on AIDA-dev set, and test on AIDA-test set (in-domain testing) and all other datasets (cross-domain testing). We follow prior work and report in-KB accuracies for AIDA-test and Bag-of-Title (BoT) F1 scores for the other test sets.

Two AIDA-CoNLL specific resources have been widely used in previous work. In order to have fair comparisons with these works, we also adopt them only for the AIDA datasets. First, we use a mention prior obtained from aliases to candidate entities released by Hoffart et al. (2011) along with the two priors described in § 4.1. Second, we also experiment with PPRforNED, an entity candidate selection system released by Pershina et al. (2015). It is unclear how candidates were pruned, but the entity candidates generated by this system have high recall and low ambiguity, and they contribute to some of the best results reported for AIDA-test (Yamada et al., 2016; Sil et al., 2018).

[7] AIDA-dev and AIDA-test are also referred as AIDA-a and AIDA-b datasets in previous work.

Competitive systems We implement four competitive ED systems, and three of them are based on variants of our proposed SGTB algorithm.[8] *Gradient tree boosting* is a local model that employs only local features to make independent decisions for every entity mention. Note that our local model is different from that presented by Yamada et al. (2016), where they treat ED as binary classification for each mention-entity pair. *SGTB-BS* is a Structured Gradient Tree Boosting model trained with Beam Search with early update strategy. *SGTB-BSG* uses Beam Search with Gold path training strategy presented in § 3.1. Finally, *SGTB-BiBSG* exploits Bidirectional Beam Search with Gold path to leverage information from both past and future for better local search.

In addition, we compare against best published results on all the datasets. To ensure fair comparisons, we group results according to candidate selection system that different ED systems adopted.

Parameter tuning We tune all the hyperparameters on the AIDA-dev set. We use recommended hyperparameter values from scikit-learn to train regression trees, except for the maximum depth of the tree, which we choose from $\{3, 5, 8\}$. After a set of preliminary experiments, we select the beam size from $\{3, 4, 5, 6\}$. The best values for the two hyperparameters are 3 and 4 respectively. As mentioned in § 2, the learning rate is set to 1. We train SGTB for at most 500 epochs (i.e., fit at most 500 regression trees). During training, we check the performance on the development set every 25 epochs to perform early stopping. Training takes 3 hours for SGTB-BS and SGTB-BSG, and takes 9 hours for SGTB-BiBSG on 16 threads.

5.3 Results

In-domain results In-domain evaluation results are presented in Table 2. As shown, SGTB achieves much better performance than all previously published results. Specifically, SGTB-BiBSG outperforms the previous SOTA system (Ganea and Hofmann, 2017) by 0.8% accuracy, and improves upon the best published results when employing the PPRforNED candidate selection system by 1.9% accuracy. Global information is clearly useful, as it helps to boost the performance by 2-4 points of accuracy, depending on the candidate generation system. In terms of beam search training strategies, BiBSG consistently outperforms BSG and beam search with early update. By employing more point-wise functional gradients to train the regression trees and leveraging global information from both past and future to carry on local search, BiBSG is able to find better global solutions than alternative training strategies.

[8] Our implementations are based on the scikit-learn package (Pedregosa et al., 2011).

System	PPRforNED	In-KB acc.
Published results		
Lazic et al. (2015)		86.4
Huang et al. (2015)		86.6
Chisholm and Hachey (2015)		88.7
Ganea et al. (2016)		87.6
Guo and Barbosa (2016)		89.0
Globerson et al. (2016)		91.0
Yamada et al. (2016)		91.5
Ganea and Hofmann (2017)		92.2
Our implementations		
Gradient tree boosting		88.4
SGTB-BS		91.7
SGTB-BSG		92.4
SGTB-BiBSG		**93.0**
Published results		
Pershina et al. (2015)	✓	91.8
Yamada et al. (2016)	✓	93.1
Sil et al. (2018)	✓	94.0
Our implementations		
Gradient tree boosting	✓	93.1
SGTB-BS	✓	95.1
SGTB-BSG	✓	95.5
SGTB-BiBSG	✓	**95.9**

Table 2: In-domain evaluation: in-KB accuracy results on the AIDA-test set. Checked PPRforNED indicates that the system uses PPRforNED (Pershina et al., 2015) to select candidate entities.The best results are in **bold**.

Cross-domain results As presented in Table 3, cross-domain experimental results are a little more mixed. SGTB-BS and SGTB-BSG perform quite competitively compared with SGTB-BiBSG. In a cross-domain evaluation setting, the test data is drawn from a different distribution as the training data. Therefore, less expressive models may be preferred as they may learn more abstract representations that will generalize better to out-of-domain data. Nevertheless, our SGTB models achieve better performance than best published results on three of the five popular ED datasets. Specifically, SGTB-BS outperforms the prior SOTA system by absolute 4% F1 on the CWEB dataset, and SGTB-BiBSG performs consistently well across different datasets.

System	AQUAINT	MSNBC	ACE	CWEB	WIKI
Published results					
Fang et al. (2016)	88.8	81.2	85.3	-	-
Ganea et al. (2016)	89.2	91.0	88.7	-	-
Milne and Witten (2008)	85.0	78.0	81.0	64.1	81.7
Hoffart et al. (2011)	56.0	79.0	80.0	58.6	63.0
Ratinov et al. (2011)	83.0	75.0	82.0	56.2	67.2
Cheng and Roth (2013)	90.0	90.0	86.0	67.5	73.4
Guo and Barbosa (2016)	87.0	92.0	88.0	77.0	**84.5**
Ganea and Hofmann (2017)	88.5	**93.7**	88.5	77.9	77.5
Our implementations					
Gradient tree boosting	90.3	91.1	**89.2**	78.8	75.0
SGTB-BS	**90.5**	92.4	88.9	81.7	76.4
SGTB-BSG	89.4	92.5	88.6	81.7	78.4
SGTB-BiBSG	89.9	92.6	88.5	**81.8**	79.2

Table 3: Cross-domain evaluation: Bag-of-Title (BoT) F1 results on ED datasets. The best results are in **bold**.

6 Related work

Entity disambiguation Most ED systems consist of a local component that models relatedness between a mention and a candidate entity, as well as a global component that produces coherent entity assignments for all mentions within a document. Recent research has largely focused on joint resolution of entities, which is usually performed by maximizing the global topical coherence between entities. As discussed above, directly optimizing the coherence objective is computationally intractable, and several heuristics and approximations have been proposed to address the problem. Hoffart et al. (2011) use an iterative heuristic to remove unpromising mention-entity edges. Yamada et al. (2016) employ a two-stage approach, in which global information is incorporated in the second stage based on local decisions from the first stage. Approximate inference techniques have been widely adopted for ED. Cheng and Roth (2013) use an integer linear program (ILP) solver. Belief propagation (BP) and its variant loopy belief propagation (LBP) have been used by Ganea et al. (2016) and Ganea and Hofmann (2017) respectively. We employ another standard approximate inference algorithm, beam search, in this work. To make beam search a better fit for SGTB training, we propose BiBSG that improves beam search training on stability and effectiveness.

Structured gradient tree boosting Gradient tree boosting has been used in some of the most accurate systems for a variety of classification and regression problems (Babenko et al., 2011; Wu et al., 2010; Yamada et al., 2016). However, gradient tree boosting is seldom studied in the context of structured learning, with only a few exceptions. Dietterich et al. (2004) propose TreeCRF that replaces the linear scoring function of a CRF with a scoring function given by a gradient tree boosting model. TreeCRF achieves comparable or better results than CRF on some linear chain structured prediction problems. Bagnell et al. (2007) extend the Maximum Margin Planning (MMP; Ratliff et al., 2006) algorithm to structured prediction problems by learning new features using gradient boosting machines. Yang and Chang (2015) present a general SGTB framework that is flexible in the choice of loss functions and specific structures. They also apply SGTB to the task of tweet entity linking with a special non-overlapping structure. By decomposing the structures into local substructures, exact inference is tractable in all the aforementioned works. Our work shows that we can train SGTB models efficiently and effectively even with approximate inference. This extends the utility of SGTB models to a wider range of interesting structured prediction problems.

7 Conclusion and future work

In this paper, we present a structured gradient tree boosting model for entity disambiguation. Entity coherence modeling is challenging, as exact inference is prohibitively expensive due to the pairwise entity relatedness terms in the objective function. We propose an approximate inference algorithm, BiBSG, that is designed specifically for SGTB to solve this problem. Experiments on benchmark ED datasets suggest that the expressive SGTB models are extremely good at dealing with the task of ED. SGTB significantly outperforms all previous systems on the AIDA-CoNLL dataset,

and it also achieves SOTA results on many other ED datasets even in the cross-domain evaluation setting. SGTB is a family of structured learning algorithms that can be potentially applied to other core NLP tasks. In the future, we would like to investigate the effectiveness of SGTB on other information extraction tasks, such as relation extraction and coreference resolution.

8 Acknowledgments

We thank Prabhanjan Kambadur and other people in the Bloomberg AI team for their valuable comments on earlier version of this paper. We also thank the NAACL reviewers for their helpful feedback. This work also benefitted from discussions with Mark Dredze and Karl Stratos.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. 2011. Robust object tracking with online multiple instance learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* .

JA Bagnell, Joel Chestnutt, David M Bradley, and Nathan D Ratliff. 2007. Boosting structured prediction for imitation learning. In *Neural Information Processing Systems (NIPS)*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*.

Xiao Cheng and Dan Roth. 2013. Relational inference for wikification. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*.

Andrew Chisholm and Ben Hachey. 2015. Entity disambiguation with web links. *Transactions of the Association for Computational Linguistics* 3.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*.

Hal Daumé III and Daniel Marcu. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Thomas G Dietterich, Adam Ashenfelter, and Yaroslav Bulatov. 2004. Training conditional random fields via gradient tree boosting. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Mark Dredze, Paul McNamee, Delip Rao, Adam Gerber, and Tim Finin. 2010. Entity disambiguation for knowledge base population. In *Proceedings of the Association for Computational Linguistics (ACL)*.

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

Wei Fang, Jianwen Zhang, Dilin Wang, Zheng Chen, and Ming Li. 2016. Entity disambiguation by knowledge and text jointly embedding. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*.

Octavian-Eugen Ganea, Marina Ganea, Aurelien Lucchi, Carsten Eickhoff, and Thomas Hofmann. 2016. Probabilistic bag-of-hyperlinks model for entity linking. In *Proceedings of the International Conference on World Wide Web (WWW)*.

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep joint entity disambiguation with local neural attention. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*.

Amir Globerson, Nevena Lazic, Soumen Chakrabarti, Amarnag Subramanya, Michael Ringaard, and Fernando Pereira. 2016. Collective entity resolution with multi-focal attention. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Stephen Guo, Ming-Wei Chang, and Emre Kiciman. 2013. To link or not to link? a study on end-to-end tweet entity linking. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Zhaochen Guo and Denilson Barbosa. 2016. Robust named entity disambiguation with random walks. *Semantic Web* .

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

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*.

Hongzhao Huang, Larry Heck, and Heng Ji. 2015. Leveraging deep neural networks and knowledge graphs for entity disambiguation. *arXiv preprint arXiv:1504.07678* .

Heng Ji and Ralph Grishman. 2011. Knowledge base population: Successful approaches and challenges. In *Proceedings of the Association for Computational Linguistics (ACL)*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Nevena Lazic, Amarnag Subramanya, Michael Ringgaard, and Fernando Pereira. 2015. Plato: A selective context model for entity resolution. *Transactions of the Association for Computational Linguistics* .

David Milne and Ian H Witten. 2008. Learning to link with wikipedia. In *Proceedings of the ACM conference on Information and knowledge management (CIKM)*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research* .

Maria Pershina, Yifan He, and Ralph Grishman. 2015. Personalized page rank for named entity disambiguation. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. 2006. Maximum margin planning. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. *Machine learning and knowledge discovery in databases* .

Avirup Sil, Gourab Kundu, Radu Florian, and Wael Hamza. 2018. Neural cross-lingual entity linking. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

Noah A Smith and Mark Johnson. 2007. Weighted and probabilistic context-free grammars are equally expressive. *Computational Linguistics* .

Valentin I Spitkovsky and Angel X Chang. 2012. A cross-lingual dictionary for english wikipedia concepts. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*.

Qing Sun, Stefan Lee, and Dhruv Batra. 2017. Bidirectional beam search: Forward-backward inference in neural sequence models for fill-in-the-blank image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Qiang Wu, Christopher JC Burges, Krysta M Svore, and Jianfeng Gao. 2010. Adapting boosting for information retrieval measures. *Information Retrieval* .

Yuehua Xu and Alan Fern. 2007. On learning linear ranking functions for beam search. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*.

Yi Yang and Ming-Wei Chang. 2015. S-mart: Novel tree-based structured learning algorithms applied to tweet entity linking. In *Proceedings of the Association for Computational Linguistics (ACL)*.

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

DeepAlignment: Unsupervised Ontology Matching With Refined Word Vectors

Prodromos Kolyvakis[1], **Alexandros Kalousis**[2], **Dimitris Kiritsis**[1]
[1]École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
{prodromos.kolyvakis,dimitris.kiritsis}@epfl.ch
[2]Business Informatics Department, University of Applied Sciences,
Western Switzerland Carouge, HES-SO, Switzerland
alexandros.kalousis@hesge.ch

Abstract

Ontologies compartmentalize types and relations in a target domain and provide the semantic backbone needed for a plethora of practical applications. Very often different ontologies are developed independently for the same domain. Such "parallel" ontologies raise the need for a process that will establish alignments between their entities in order to unify and extend the existing knowledge. In this work, we present a novel entity alignment method which we dub *DeepAlignment*. DeepAlignment refines pre-trained word vectors aiming at deriving ontological entity descriptions which are tailored to the ontology matching task. The absence of explicit information relevant to the ontology matching task during the refinement process makes DeepAlignment completely unsupervised. We empirically evaluate our method using standard ontology matching benchmarks. We present significant performance improvements over the current state-of-the-art, demonstrating the advantages that representation learning techniques bring to ontology matching.

1 Introduction

Translation across heterogeneous conceptual systems is an important challenge for cognitive science (Goldstone and Rogosky, 2002; Stolk et al., 2016). Ontology Matching constitutes the task of establishing correspondences between semantically related entities (i.e. classes and properties) from different ontologies, as illustrated in Figure 1. Similarly, ontology matching is crucial for accomplishing a mutual understanding across heterogeneous artificial cognitive agents (Taylor, 2015). However, despite the many proposed solutions, it is widely accepted that there is no solution robust enough to deal with the high ontological linguistic variability (Shvaiko and Euzenat, 2008, 2013); hampering, thus, the discovery of shared meanings.

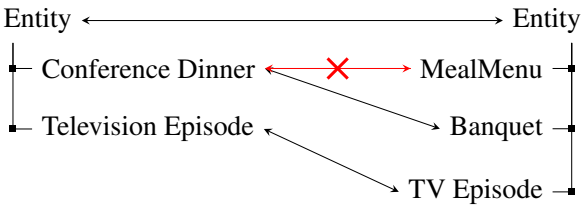


Figure 1: Example of alignments (black lines) and misalignments (red crossed lines) between ontologies.

Research in automatic ontology matching has focused on engineering features from terminological, structural, extensional (ontology instances) and semantic model information extracted from the ontological model. These features are then used to compute ontological entity similarities that will guide the ontology matching. Deriving such features for a given problem is an extremely time consuming task. To make matters worse, these features do not transfer in other domains. As Cheatham and Hitzler (2013) have recently shown, the performance of ontology matching based on different textual features varies greatly with the type of ontologies under consideration.

At the same time, machine learning research is characterised by a shift from feature engineering based approaches to feature and representation learning as a result of the performance improvements brought by deep learning methods. A by now classical example is the unsupervised learning of semantic word representations based on the *distributional hypothesis* (Harris, 1954), i.e. the assumption that semantically similar or related words appear in similar contexts (Deerwester et al., 1990; Bengio et al., 2003; Mikolov et al., 2013a,c; Pennington et al., 2014). Word vectors have the potential to bring significant value to on-

Proceedings of NAACL-HLT 2018, pages 787–798
New Orleans, Louisiana, June 1 - 6, 2018.

tology matching given the fact that a great deal of ontological information comes in textual form.

One drawback of these semantic word embeddings is that they tend to coalesce the notions of *semantic similarity* and *conceptual association* (Hill et al., 2016b). For instance, the word "harness" is highly related to the word "horse", as they share strong associations, i.e. a harness is often used on horses (Lofi, 2016). From an ontological point of view, however, these types should not be similar. Moreover, as unsupervised learning requires even larger text corpora, the learned vectors tend to bring closer words with similar frequency instead of similar meaning (Faruqui et al., 2016). Clearly, word representations that reflect frequency instead of meaning is an undesired feature if we seek to exploit word vectors for ontology matching; alignment based on such representations will reflect similar frequency instead of similar meaning.

A number of lightweight vector space representation refining techniques were introduced recently in an effort to correct these biases (Faruqui et al., 2015; Mrkšić et al., 2016). They use synonymy and antonymy constraints extracted from semantic lexicons to refine the learned word representations and make them better suited for semantic similarity tasks. Such methods are a way to inject domain-specific knowledge to tailor the learned word representations to a given task. As a result, we can exploit the synonymy/antonymy constraints to learn semantic word representations that are better candidates for ontology matching.

In this paper we learn representations of ontological entities instead of feature engineering them. We use the learned representations to compute the entities' semantic distances and to subsequently perform the ontology matching task. In order to represent the ontological entities, we exploit the textual information that accompanies them. We represent words by learning their representations using synonymy and antonymy constraints extracted from general lexical resources and information captured implicitly in ontologies. We cast the problem of ontology matching as an instance of the Stable Marriage problem (Gale and Shapley, 1962) using the entities semantic distances.

Our approach has a number of advantages. The word embeddings we establish are tailored to the domains and ontologies we want to match. The method relies on a generic unsupervised representation learning solution which is important given the small size of training sets in ontology matching problems. We evaluate our approach on the Conference dataset provided by the Ontology Alignment Evaluation Initiative (OAEI) campaign and on a real world alignment scenario between the Schema.org and the DBpedia Ontologies. We compare our method to state-of-the-art ontology matching systems and show significant performance gains on both benchmarks. Our approach demonstrates the advantages that representation learning can bring to the task of ontology matching and shows a novel way to study the problem in the setting of recent advances in NLP.

2 Related Work

2.1 Selecting Features for Ontology Matching

The vast majority of ontology matching research follows the feature engineering approach (Wang and Xu, 2008; Cruz et al., 2009; Khadir et al., 2011; Jiménez-Ruiz and Grau, 2011; Fahad et al., 2012; Ngo and Bellahsene, 2012; Gulić et al., 2016). Features are generated using a broad range of techniques (Anam et al., 2015; Harispe et al., 2015), ranging from the exploitation of terminological information, including structural similarities and logical constraints, such as datatype properties, cardinality constraints, etc.

Ontology matching is done by acting on the aforementioned features in different ways. Heuristic methods that rely on aggregation functions, such as *max, min, average, weighted sum*, etc., to fuse the information found in these features are quite popular (Anam et al., 2015). Other approaches use first order logic and cast ontology matching as a satisfiability problem (Giunchiglia et al., 2004; Jiménez-Ruiz and Grau, 2011).

Several works exploit supervised machine learning for Ontology Matching. Mao et al. (2011) cast ontology mapping as a binary classification problem. They generate various domain independent features to describe the characteristics of the entities and train an SVM classifier on a set which provides positive and negative examples of entity alignments. In general, the number of real alignments is orders of magnitude smaller than the number of possible alignments which introduces a serious class imbalance problem (Mao et al., 2008) hindering learning. Since we only use supervision to refine the word vector representations we avoid altogether the class imbalance problem.

2.2 Deep Learning for Ontology Matching

Deep learning has so far limited impact on ontology matching. To the best of our knowledge, only two approaches, (Zhang et al., 2014; Xiang et al., 2015), have explored the use of unsupervised deep learning techniques. Zhang et al. (2014) are considered to be the first ones that use word vectors in ontology matching. They train *word2vec* (Mikolov et al., 2013a) vectors on Wikipedia. They use the semantic transformations to complement the lexical information, i.e. names, labels and comments, describing entities. Their entity matching strategy is based on maximum similarity; for every entity e in the source ontology O, the algorithm finds the most similar entity e' in the target ontology O'. Their experiments on the OAEI benchmarks show that their techniques, even when combined with classical NLP techniques, could not outperform the state-of-the-art. In contrast, we refine pre-trained word embeddings with the intention of leveraging a new word vector set that is tailored to the ontology matching task.

Xiang et al. (2015) propose an entity representation learning algorithm based on Stacked Auto-Encoders (Bengio et al., 2007). To describe an entity they use a combination of its class ID, labels, comments, properties descriptions and its instances' descriptions. The entities' similarity is computed with a fixed point algorithm. They perform the entity matching using the Stable Marriage algorithm. Training such powerful models with so small training sets is problematic. We overcome this by using a transfer learning approach, known to reduce learning sample complexity (Pentina and Ben-David, 2015), to adapt pre-trained word vectors to a given ontological domain.

3 DeepAlignment

We present an ontology matching approach that uses information from ontologies and additional knowledge sources to extract synonymy/antonymy relations which we use to refine pre-trained word vectors so that they are better suited for the ontology matching task. We represent each ontological entity as the bag of words of its textual description, which we complement with the refined word embeddings. We match the entities of two different ontologies using the Stable Marriage algorithm over the entities' pairwise distances. We compute the aforementioned distances using a variant of a document similarity metric.

3.1 Preliminaries

Before we proceed with the presentation of the method, we will provide a formal definition of what an entity correspondence is. Given two ontologies O and O', we define the correspondence between two entities $e \in O$ and $e' \in O'$ as the five-element tuple:

$$cor_{e,e'} =< id, e, e', r, n > \quad (1)$$

where r is a matching relation between e and e' (e.g., equivalence, subsumption) and $n \in [0, 1]$ is the degree of confidence of the matching relation between e and e' (Euzenat and Shvaiko, 2013). The id holds the unique identifier of the mapping. Unlike the majority of ontology alignment systems which discover one-to-one equivalence mappings (Anam et al., 2015), we focus on discovering many-to-many mappings. We will also introduce some additional notation used in the paper. Let $u_1, u_2 \in \mathbb{R}^d$ be two d-dimensional vectors, we compute their cosine distance as follows: $d(u_1, u_2) = 1 - cos(u_1, u_2)$. For $x \in \mathbb{R}$, we define the *rectifier* activation function as: $\tau(x) = \max(x, 0)$.

3.2 Learning Domain Specific Word Vectors

The *counter-fitting* method (Mrkšić et al., 2016) uses synonymy and antonymy relations extracted from semantic lexicons to refine and adapt pre-trained word embeddings for given semantic similarity tasks. We broaden the concept of antonymy relations and allow for a larger class of ontology relations to define antonymies. This allows us to inject domain knowledge encoded in ontologies and produce more appropriate word vectors for the ontology matching task. In the rest of the section we revise the main elements of the counter-fitting method and describe how we can exploit it for learning domain specific word embeddings.

Let $V = \{v_1, v_2, \dots v_N\}$ be an indexed set of word vectors of size N. The counter-fitting method transforms a pretrained vector set V into a new one $V' = \{v'_1, v'_2, \dots v'_N\}$, based on a set of synonymy and antonymy constraints S and A, respectively. This is done by solving the following non-convex optimization problem:

$$\min_{V'} \kappa_1 AR(V') + \kappa_2 SA(V') + \kappa_3 VSP(V, V')$$

The $AR(V')$ function defined as:

$$AR(V') = \sum_{(u,w)\in A} \tau(1 - d(v'_u, v'_w))$$

is called *antonym repel* and pushes the refined word vectors of "antonymous" words to be away from each other. As we already mentioned, we extend the notion of antonymy relations with respect to its more narrow traditional linguistic definition. We consider that two entities in a given ontology are "antonymous" if they have not been explicitly stated as equivalent, in the sense of a logical assertion or a synonymy relation found in a semantic lexicon.

The $SA(V')$ function defined as:

$$SA(V') = \sum_{(u,w)\in S} d(v'_u, v'_w)$$

is called *synonym attract* and brings closer the transformed word vectors of synonyms. In order to extract synonymy information we search for paraphrases in semantic lexicons. Concretely, let $\omega_1 = \{word_1^1, word_2^1, \ldots, word_m^1\}$, $\omega_2 = \{word_1^2, word_2^2, \ldots, word_n^2\}$ be the textual information of two entities from different ontologies. If the combination $\{word_i^1, word_j^2\}$ or $\{word_j^2, word_i^1\}$ for some $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, n\}$ appears as a paraphrase in any semantic lexicon then we add the synonymy information (u, w) in the set S of synonymy constraints.

The $VSP(V, V')$ function defined as:

$$VSP(V, V') = \sum_{i=1}^{N} \sum_{j\in N(i)} \tau(d(v'_i, v'_j) - d(v_i, v_j))$$

forces the refined vector space to reflect the original word-vector distances. $N(i)$ is the set of words that lie within ρ distance from the i-th word vector in the original vector-space. The experiments show that the value of ρ does not affect significantly the performance of the whole algorithm, so for computational efficiency we fix it to $\rho = 0.05$. We minimize the objective function with stochastic gradient descent (SGD). We use as a convergence criterion the norm of the gradient. We continue updating the model until this is smaller than 10^{-5}. In our experiments we typically observe convergence with less than 25 iterations.

3.3 Semantic Distance Between Entities

As before, let V' be the refined word vectors and $\omega_1 = \{word_1^1, word_2^1, \ldots, word_m^1\}$, $\omega_2 = \{word_1^2, word_2^2, \ldots, word_n^2\}$ be the textual information that describes two entities from different ontologies. The textual information of an entity can be extracted from different sources, such as the entity's name, label, comments, etc. We replace the appearance of a word with its refined word vector. Hence, we end up with two sets of word vectors Q and S, respectively. In order to do the matching of the entities of two ontologies we use a semantic distance over the entities' representations, here the set of word vectors associated with each entity.

There have been many ways to compute the semantic similarity of two word sets, such as the *Word Moving Distance* (Kusner et al., 2015) and the *Dual Embedding Space Model* (DESM) (Nalisnick et al., 2016). We will base our semantic distance δ on a slight variation of the DESM similarity metric. Our metric δ computes the distance of two sets of word vectors Q and S as follows:

$$\delta(Q, S) = \frac{1}{|Q|} \sum_{q_i \in Q} d(q_i, \bar{S}) \qquad (2)$$

where $\bar{S} = \frac{1}{|S|} \sum_{s_j \in S} \frac{s_j}{\|s_j\|}$ is the normalised average of the word embeddings that constitute the set of words S.

Hence, one of the word vectors' sets is represented by the centroid of its normalized vectors. The overall set-to-set distance δ is the normalized average of the cosine distance d between the computed centroid and the other's set word vectors. A first observation is that the introduced distance is not symmetric. Ideally, we would expect the semantic distance of two word sets to be irrelevant of the order of the inputs. To make it symmetric, we redefine the distance between two sets of word vectors as:

$$dis(\omega_1, \omega_2) = \max(\delta(Q, S), \delta(S, Q)) \qquad (3)$$

It is important to note that $dis(\omega_1, \omega_2)$ is not a proper distance metric as it does not satisfy the triangle inequality property. Despite this fact, it has proved to work extremely well on all the ontology matching scenarios.

3.4 Ontology Matching

Similar to the work in (Xiang et al., 2015) we use the extension of the Stable Marriage Assignment

problem to unequal sets (Gale and Shapley, 1962; McVitie and Wilson, 1970). The stable marriage algorithm computes one-to-one mappings based on a preference $m \times n$ matrix, where m and n is the number of entities in ontologies O and O', respectively. Note that the violation of the triangle inequality by our semantic distance (equation 3) is not an impediment to the Stable Marriage algorithm (Gale and Shapley, 1962).

The majority of the ontology matching systems produce equivalence mappings with cardinality one-to-one. Hence, one entity e in ontology O can be mapped to at most one entity in e' in O' and vice versa. According to a recent review (Anam et al., 2015) only two out of almost twenty ontology matching systems provide solutions to detect many-to-many mappings. However, ontology designers focus on different degrees of granularity, so it is expected that one entity from some ontology can correspond to more than one entities in another ontology and vice-verca.

To address this problem, we present an algorithm that extends the one-to-one mappings of the previous step to many-to-many. The basic idea is that some alignments that were omitted by the Stable Marriage solution were very close to the optimal alignment and they should also be included in the final alignment set. However, despite the use of refined word vectors, we cannot completely avoid the problems that come from the semantic similarity and conceptual association coalescence. The solution of this problem comes from the observation that we can add the constraint that the mapping should be extended only in the case that the new entity that will be added will share a subsumption relation with the existing one. Below we give a more formal definition of what we will call an ϵ-optimal mapping between two entities e and e' that belong to two different ontologies O and O' respectively.

Definition 1 *Let $e \rightarrow e'$ be the optimal mapping - produced by the Stable Marriage Solution - from the entity $e \in O$ to the entity $e' \in O'$, where O and O' are two different ontologies. Let $e \rightarrow e''$ be another mapping, where $e'' \in O'$. Given an $\epsilon > 0$, we call the mapping $e \rightarrow e''$ ϵ-optimal with respect to the mapping $e \rightarrow e'$ if and only if the following two hold:*

- $|dis(\omega_1, \omega_2) - dis(\omega_1, \omega_3)| < \epsilon$, *where ω_1, ω_2, ω_3 is the textual information of entities e, e' and e'', respectively.*

Algorithm 1 extendMap(e, h, $\mathcal{O}'$, P_e, $i_{e'}$, n, ϵ, r)

Require: source entity: e
hash function from integers to entities: h
subsumption's transitive closure: $\mathcal{O}'$
sorted (increasingly) preference matrix: P_e
index of optimal solution: $i_{e'}$
number of target's ontology entities: n
$\epsilon-$optimality value: ϵ
number of relatives: r

Ensure: sequence of the $\epsilon-$optimal mappings

```
Initialization: list = ∅
opt = P_e[i_{e'}]
e' = h(i_{e'})
for i = min(i_{e'} + 1, n) to min(i_{e'} + r, n) do
    tmp = P_e[i]
    if abs(opt − tmp) < ε then
        e_i = h(i)
        if (e_i, e') ∈ O' or (e', e_i) ∈ O' then
            list.append(e → e_i)
        end if
    end if
end for
```

- *e' and e'' should be logically related with a subsumption relation. Equivalently, there must be either a logical assertion that e' is subclass of e'' or e'' is subclass of e'.*

The *subsumption restriction* requires that the extended alignments share a taxonomic relation, in order to avoid matchings between entities that are conceptually associated. We iteratively search for ϵ-optimal mappings according to the algorithm 1 to extend the established one-to-one mappings to many-to-many. For efficiency reasons, we do not check all the entities, but only the r closest entities according to the $dis(\omega_1, \omega_2)$ distance. As a final step, we iteratively pass through all the produced alignments and we discard those with $dis(\omega_1, \omega_2)$ greater than a hyperparameter value $thres$.

4 Experiments

In this section, we present the experiments we performed on the OAEI conference dataset and in one real word alignment scenario between the Schema.org and DBpedia ontologies. One of the main problems that we have encountered with the comparative evaluation of our algorithm is that even though numerous ontology matching algorithms exist, for only a very small portion of them either the respective software or the system's out-

put is publicly available. To the best of our knowledge, among all the systems tested in the conference dataset only AML (Cruz et al., 2009) and LogMap (Jiménez-Ruiz and Grau, 2011) are publicly available. As it happens these are two of the state-of-the-art systems. Moreover, AML offers solutions to detect many-to-many alignments (Faria et al., 2015) and, thus, constitutes a competitive baseline against which we will compare the performance of extendMap which also provides many-to-many alignments.

When training to refine the vector representations an unbalanced proportion of synonymy and antonymy constraints sets can cause problems; the set with the lower cardinality will have limited impact on the final word representations. To overcome this problem, we run an additional step of the counter-fitting procedure, using only a small random subset of the supernumerary constraints and all constraints of the minority set. We randomly undersample the larger set and reduce its cardinality to that of the smaller set. We call this additional step the *recounter-fitting* process. To demonstrate the importance of the recounter-fitting process and test the behavior of the pre-trained word vectors in the absence of synonymy and/or antonymy relations, we have conducted additional experiments which we also present.

In all of our experiments we have applied the counter-fitting process upon the Paragram-SL999 word vectors provided by Wieting et al. (2015). With respect to the textual information extracted for each entity, we have only used the entity's ID (rdf:ID). To estimate the precision, recall and $F1$ measure of all the systems, that we consider for testing, and check for the statistical significance of the results we use an approximate randomization test with 1048576 shuffles, as described in Yeh (2000).

4.1 Semantic Lexicons

Let $\omega_1 = \{word_1^1, word_2^1, \ldots, word_m^1\}$, $\omega_2 = \{word_1^2, word_2^2, \ldots, word_n^2\}$ be the textual information that accompanies two entities from different ontologies. We extracted the synonymy and antonymy constraints that we used in the experiments from the following semantic lexicons:

WordNet: a well known lexical database for the English language (Miller, 1995). In our experiments we did not use WordNet synonyms. Instead, we have included WordNet antonymy pairs together with the "antonymy" relations extracted by the ontologies. The strategy that we have followed in order to create the WordNet's antonymy pairs is that every two words with antonymous word senses, we have considered them as antonyms.

PPDB 2.0: the latest release of the Paraphrase Database (Pavlick et al., 2015). We have used this database in two different ways. We have used the largest available single-token terms (XXXL version) in the database and we have extracted the *Equivalence* relations as synonyms, and the *Exclusion* relations as antonyms. Additionally, we have searched the whole XXXL version of PPDB for paraphrases based on the words appeared in two entities from different ontologies. Namely, our strategy was the following: If the pair $(word_i^1, word_j^2)$ or the pair $(word_j^2, word_i^1)$ appeared on the PPDB and their type of relation was not *Exclusion*, we considered it as synonym.

WikiSynonyms: a semantic lexicon which is built by exploiting the Wikipedia redirects to discover terms that are mostly synonymous (Dakka and Ipeirotis, 2008). In our experiments we have used it only on the Schema.org[1] - DBpedia[2] scenario. Our strategy was the following: we search if there exist synonyms in the WikiSynonyms for the ω_1 and ω_2. If this is the case, we extract them and we stop there. In the opposite case we extract the synonyms for each $word_i^1$ and $word_j^2$.

4.2 Hyperparameter Tuning

We tuned the hyperparameters on a set of 100 alignments which we generated by randomly sampling the synonyms and antonyms extracted from WordNet and PPDB. We chose the vocabulary of the 100 alignments so that it is disjoint to the vocabulary that we used in the alignment experiments, described in the evaluation benchmarks, in order to avoid any information leakage from training to testing. We tuned to maximize the $F1$ measure. In particular, we did a coarse grid search over a parameter space for $\kappa_1, \kappa_2, \kappa_3$, r, ϵ and $thres$. We considered $\kappa_1, \kappa_2 \in [0.35, 0.45]$ and $\kappa_3 \in [0.1, 0.2]$ with common step 0.01, $r \in [1, 10]$ with step 1, $\epsilon \in [0.01, 0.1]$ with step 0.01 and $thres \in [0.3, 0.7]$ with step 0.05. We trained for 25 epochs for each hyperparameter using SGD.

[1] https://github.com/schemaorg/schemaorg/blob/sdo-callisto/data/releases/3.2/schema.ttl

[2] http://downloads.dbpedia.org/2014/dbpedia_2014.owl.bz2

The best values were the following: $\kappa_1 = 0.4$, $\kappa_2 = 0.4$, $\kappa_3 = 0.1$, $r = 8$, $\epsilon = 0.07$ and $thres = 0.5$. We used the selected configuration on all the alignment scenarios described below.

4.3 Evaluation Benchmarks

One of our evaluation benchmarks comes from the Ontology Alignment Evaluation Initiative (OAEI), which organizes annual campaigns for evaluating ontology matching systems. The external to OAEI evaluation benchmark comes from the provided alignments between the Schema.org and the DBpedia ontologies. We provide some further details for each dataset below:

OAEI Conference Dataset: It contains 7 ontologies addressing the same domain, namely the conference organization. These ontologies are suitable for ontology matching task because of their heterogeneous character of origin. The overall performance (micro-precision, micro-recall, micro-F1) of the systems is tested upon 21 different test cases. Specifically, we summed up the individual true positives, false positives and false negatives based on the system results for the different ontology matching tasks and, in the next step, we computed the performance metrics. The original reference alignment is not closed under the alignment relation, so the transitive closure should be computed before proceeding on the evaluation of the systems.

Schema.org - DBpedia Alignment: It corresponds to the incomplete mapping of the Schema.org and DBpedia ontologies. Schema.org is a collaborative, community activity with a mission to create, maintain, and promote schemas for structured data on the Internet, on web pages, in email messages, and beyond. On the other hand, DBpedia is a crowd-sourced community effort to extract structured information from Wikipedia and make this information available on the Web. This alignment corresponds to a real case scenario between two of the most widely used ontologies in the web today.

4.4 Experimental Results

All the systems presented in the Conference dataset experiments (Table 1) fall into the category of feature engineering. CroMatch (Gulić et al., 2016), AML (Cruz et al., 2009), XMap (Djeddi and Khadir, 2010) perform ontology matching based on heuristic methods that rely on aggregation functions. LogMap and LogMapBio (Jiménez-Ruiz and Grau, 2011) use logic-based reasoning over the extracted features and cast the ontology matching to a satisfiability problem.

4.4.1 OAEI Conference Dataset

Table 1 shows the performance of our algorithm compared to the five top performing systems on the Conference 2016 benchmark, according to the results published in OAEI[3]. DeepAligment achieves the highest micro-$F1$ measure and the highest recall. We were able to perform statistical significance test only for the two systems that were publicly available. DeepAlignment is significantly better than both of them with a $p\text{-}value \leq 0.05$. In order to explore the performance effect of the many-to-many mappings that DeepAlignment produces we also did experiments where our extendMap algorithm was not used, thus generating only one-to-one alignments. We give these results under the DeepAlignment$_*$ listing. It can be seen that DeepAlignment$_*$ achieves the same level of recall as the state-of-the-art systems and this with no feature engineering. When we compare the performance of DeepAlignment$_*$ and DeepAlignment we see that the use of extendMap generates correct many-to-many alignments and thus it does not produce large numbers of false positives. In any case, however, we retain a small precision which indicates a semantic similarity and conceptual association coalescence.

System	Precision	Recall	Micro-F1
DeepAlignment	0.71	**0.80**	**0.75**
CroMatch	0.76	0.69	0.72
AML	0.79	0.65	0.71
DeepAlignment$_*$	0.68	0.68	0.68
XMap	0.81	0.58	0.67
LogMap	0.79	0.58	0.66
LogMapBio	0.75	0.58	0.65
StringEquiv	0.83	0.50	0.62

Table 1: Results on Conference OAEI dataset. StringEquiv corresponds to ontology matching by simple string equivalence check.

We perform additional experiments to investigate the importance of the counter-fitting step, which are summarized in Table 2. In all of these experiments, we have applied the extendMap algorithm. The last row of Table 2, corresponds to the best result reported in Table 1. The first row

[3] http://oaei.ontologymatching.org/2016/

gives the results of executing the algorithm without the counter-fitting process, just by providing the Paragram-SL999 word vectors.

Parameters		Precision	Recall	Micro-F1
Synonyms	**Antonyms**			
No	No	0.63	0.55	0.59
No	Yes	0.67	0.51	0.58
Yes	No	0.69	0.72	0.71
Yes	Restricted	0.65	0.78	0.71
Yes	Yes	**0.71**	**0.80**	**0.75**

Table 2: Experiments on Conference OAEI dataset.

The results support the importance of the counter-fitting process, which succeeds in tailoring the word embeddings to the ontology matching task. By injecting only antonymy information (second row), we observe an increase in precision, but a decrease in recall. This behavior is due to the fact that the antonym repel factor imposes an orthogonality constraint to the word vectors, leading to higher values of the dis distance. In absence of synonymy information, the majority of words tend to become "antonymous". The third row of Table 2 gives the performance when we also include synonyms extracted from PPDB but no antonymy information. We can see that this leads to a large increase of all the recorded performance metrics. Finally, we also include antonymy information only from the Cmt and the Conference ontologies found in the Conference dataset. This has two effects: an increase in recall, but a decrease in precision. This can be explained by the fact that even though all ontologies describe the same domain the description granularity provided by each of them is not capable of giving all the antonymy relations needed to provide more refined alignments.

4.4.2 Schema.org - DBpedia Alignment

Table 3 summarizes the obtained results from the matching of the Schema.org and DBpedia ontologies. The fact that the alignment is incomplete restricts us on testing the performance only on the recall. To make the comparison as fair as possible, we did not apply the extendMap algorithm. We should highlight that we have applied the recounter-fitting process because the synonyms that we have extracted from the PPDB and WikiSynonyms were very few compared to the constructed "antonyms". The results of the LogMap system show a quite similar behavior with the experiments conducted in the conference dataset. However the recall of AML is zero. It

System	Recall
DeepAlignment$_*$	0.82
LogMap	0.5
AML	0

Table 3: Results on aligning Schema.org and DBpedia ontologies.

discovers none of the available alignments even though it manages to recall other quite reasonable matchings, which, however, are not included in the ground truth. According to our understanding, this might be an indication of the absence of domain transferability of the extracted features as well as of the implemented metrics. We summarize in Table 4 the results of the experiments we did on the two domains to study the effect of counter-fitting and recounter-fitting. As we can see, even without the counter-fitting, the semantic embeddings show quite good results. This provides evidence on the importance of using representation learning techniques instead of the classical feature engineering choice. By injecting only antonymy information (second row), we observe a different behavior in the recall metric compared to the one presented in Table 2. This can be explained by the fact that while the antonym repel factor imposes an orthogonality constraint, its effect is by no means universal to the whole word vector space. Therefore, a misalignment can be pushed far away leaving the space open for a true alignment to be detected. With the addition of the extracted synonyms, we observe an increase of 0.13 in the recall. However, the insertion of the extracted "antonyms" leads to lower performance. This shows practically the importance of applying the recounter-fitting process that allows both the synonym attract and the antonym repel factors to affect the word vectors.

Parameters			Recall
Recounter-fitting	**Synonyms**	**Antonyms**	
No	No	No	0.71
No	No	Yes	0.76
No	Yes	No	0.84
No	Yes	Yes	0.76
Yes	Yes	Restricted	**0.82**

Table 4: Experiments on aligning Schema.org and DBpedia ontologies. Restricted indicates that we choose only a small random subset of the antonymy constraints.

4.5 Further Analysis

DeepAlignment vs. initial word vectors. To investigate the impact of the initial pre-trained word vectors on DeepAlignment's performance, we carried out two additional experiments, this time using a set of word2vec vectors (Mikolov et al., 2013b), trained on the Google news dataset[4]. We report and compare the obtained results to the ones produced by the use of Paragram-SL999 vectors in Table 5. In the absence of counter-fitting,

Counter fitting	Word Vectors	Conference			Schema.org DBpedia
		P	R	Micro-F1	R
No	word2vec	0.64	0.52	0.58	0.74
No	Paragram	0.63	0.55	0.59	0.71
Yes	word2vec	0.67	0.75	0.71	0.75
Yes	Paragram	0.71	0.80	0.75	0.76

Table 5: Dependency of DeepAlignment's performance on the choice of the initial word vectors[6].

the word2vec vectors achieve better results on the Schema.org - DBpedia scenario, however, they exhibit lower performance on the conference dataset. This observation is in accordance with recent studies (Hill et al., 2016a) which show that different word vectors optimization objectives yield representations tailored to different applications and domains. After the application of the counter-fitting process, the use of Paragram-SL999 vectors leads to a better performance. This fact provides additional evidence that word vectors which reflect semantic similarity are better candidates for being further tailored to the ontology matching task.

DeepAlignment vs. resources' coverage. The choice and coverage of the different lexical resources may have a determining factor on the performance of DeepAlignment. For that reason, we present in Table 6 a set of experiments where we exclude a part of the synonymy/antonymy relations from the various semantic lexicons. For both the matching scenarios, we experimented with excluding all the antonyms from PPDB and WikiSynonyms. For the conference dataset, we additionaly experimented with including only a subset of PPDB synonyms (50% coverage). Finally, we carried out one experiment where we excluded all the synonymy information extracted from WikiSynonyms for the Schema.org - DBpedia scenario. The resulted performance is presented in the rows 1, 4, 2, 5 of Table 6, respectively. The reported results provide evidence that the greater the coverage of synonyms and antonyms, the greater the performance of DeepAlignment will be.

Dataset	Experiment Setting	P	R	Micro-F1
Conference	With no antonyms from PPDB & WikiSynonyms	0.67	0.76	0.71
	With only a subset of the PPDB synonyms	0.67	0.76	0.71
	With all the available synonyms/antonyms	0.71	0.80	0.75
Schema.org DBpedia	With no antonyms from PPDB & WikiSynonyms	-	0.76	-
	With no synonyms from WikiSynonyms	-	0.73	-
	With all the available synonyms & antonyms	-	0.76	-

Table 6: Dependency of DeepAlignment's performance on the external resources' coverage[6].

5 Conclusion

In this paper, we propose the refinement of pre-trained word vectors with the purpose of deriving ontological entity descriptions which are tailored to the ontology matching task. The refined word representations are learned so that they incorporate domain knowledge encoded in ontologies as well as knowledge extracted from semantic lexicons. The refinement procedure does not use any explicit information relevant to the ontology matching task making the entity representation task completely unsupervised. We perform ontology matching by applying the Stable Marriage algorithm over the entities' pairwise distances. Our experimental results demonstrate significant performance gains over the state-of-the-art and show a novel way to study the problem of ontology matching under the setting of NLP.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments on the paper. This project was supported by the Swiss State Secretariat for Education, Research and Innovation SERI (SERI; contract number 15.0303) through the European Union's Horizon 2020 research and innovation programme (grant agreement No 688203; bIoTope). This paper reflects the authors' view only, and the EU as well as the Swiss Government is not responsible for any use that may be made of the information it contains.

[4] https://code.google.com/p/word2vec

[6] For the Schema.org - DBpedia scenario's experiments, the recounter-fitting process has not been applied.

References

Sarawat Anam, Yang Sok Kim, Byeong Ho Kang, and Qing Liu. 2015. Review of ontology matching approaches and challenges. *International journal of Computer Science and Network Solutions* 3(3):1–27.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research* 3(Feb):1137–1155.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al. 2007. Greedy layer-wise training of deep networks. *Advances in neural information processing systems* 19:153.

Michelle Cheatham and Pascal Hitzler. 2013. String similarity metrics for ontology alignment. In *International Semantic Web Conference*. Springer, pages 294–309.

Isabel F Cruz, Flavio Palandri Antonelli, and Cosmin Stroe. 2009. Agreementmaker: efficient matching for large real-world schemas and ontologies. *Proceedings of the VLDB Endowment* 2(2):1586–1589.

Wisam Dakka and Panagiotis G Ipeirotis. 2008. Automatic extraction of useful facet hierarchies from text databases. In *2008 IEEE 24th International Conference on Data Engineering*. IEEE, pages 466–475.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science* 41(6):391.

Warith Eddine Djeddi and Mohammed Tarek Khadir. 2010. Xmap: a novel structural approach for alignment of owl-full ontologies. In *Machine and Web Intelligence (ICMWI), 2010 International Conference on*. IEEE, pages 368–373.

Jérôme Euzenat and Pavel Shvaiko. 2013. *Ontology matching*. Springer-Verlag, Heidelberg (DE), 2nd edition.

Muhammad Fahad, Nejib Moalla, and Abdelaziz Bouras. 2012. Detection and resolution of semantic inconsistency and redundancy in an automatic ontology merging system. *Journal of Intelligent Information Systems* 39(2):535–557.

Daniel Faria, Catarina Martins, Amruta Nanavaty, Daniela Oliveira, Booma Sowkarthiga, Aynaz Taheri, Catia Pesquita, Francisco M Couto, and Isabel F Cruz. 2015. Aml results for oaei 2015. In *OM*. pages 116–123.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Denver, Colorado, pages 1606–1615. http://www.aclweb.org/anthology/N15-1184.

Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. 2016. Problems with evaluation of word embeddings using word similarity tasks. In *Proc. of the 1st Workshop on Evaluating Vector Space Representations for NLP*. http://aclweb.org/anthology/W/W16/W16-2506.pdf.

David Gale and Lloyd S Shapley. 1962. College admissions and the stability of marriage. *The American Mathematical Monthly* 69(1):9–15.

Fausto Giunchiglia, Pavel Shvaiko, and Mikalai Yatskevich. 2004. S-match: an algorithm and an implementation of semantic matching. In *European semantic web symposium*. Springer, pages 61–75.

Robert L Goldstone and Brian J Rogosky. 2002. Using relations within conceptual systems to translate across conceptual systems. *Cognition* 84(3):295–320.

Marko Gulić, Boris Vrdoljak, and Marko Banek. 2016. Cromatcher: An ontology matching system based on automated weighted aggregation and iterative final alignment. *Web Semantics: Science, Services and Agents on the World Wide Web* 41:50–71.

Sebastien Harispe, Sylvie Ranwez, Stefan Janaqi, and Jacky Montmain. 2015. Semantic similarity from natural language and ontology analysis. *Synthesis Lectures on Human Language Technologies* 8(1):1–254.

Zellig S Harris. 1954. Distributional structure. *Word* 10(2-3):146–162.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016a. Learning distributed representations of sentences from unlabelled data. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*. pages 1367–1377. http://aclweb.org/anthology/N/N16/N16-1162.pdf.

Felix Hill, Roi Reichart, and Anna Korhonen. 2016b. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics* .

Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau. 2011. Logmap: Logic-based and scalable ontology matching. In *International Semantic Web Conference*. Springer, pages 273–288.

MT Khadir, A Djeddai, and W Djeddi. 2011. Xmap++: A novel semantic approach for alignment of owl-full ontologies based on semantic relationship using wordnet. In *Innovation in Information & Communication Technology (ISIICT), 2011 Fourth International Symposium on*. IEEE, pages 13–18.

Matt J Kusner, Yu Sun, Nicholas I Kolkin, and Kilian Q Weinberger. 2015. From word embeddings to document distances. In *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*. pages 957–966.

Christoph Lofi. 2016. Measuring semantic similarity and relatedness with distributional and knowledge-based approaches. *Database Society of Japan (DBSJ) Journal* 14(1):1–9.

M. Mao, Y. Peng, and M. Spring. 2008. Ontology mapping: As a binary classification problem. In *2008 Fourth International Conference on Semantics, Knowledge and Grid*. pages 20–25. https://doi.org/10.1109/SKG.2008.101.

Ming Mao, Yefei Peng, and Michael Spring. 2011. Ontology mapping: as a binary classification problem. *Concurrency and Computation: Practice and Experience* 23(9):1010–1025.

DG McVitie and Leslie B Wilson. 1970. Stable marriage assignment for unequal sets. *BIT Numerical Mathematics* 10(3):295–309.

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

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013c. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 142–148. http://www.aclweb.org/anthology/N16-1018.

Eric Nalisnick, Bhaskar Mitra, Nick Craswell, and Rich Caruana. 2016. Improving document ranking with dual word embeddings. In *Proceedings of the 25th International Conference Companion on World Wide Web*. International World Wide Web Conferences Steering Committee, pages 83–84.

Duy Hoa Ngo and Zohra Bellahsene. 2012. Yam++:(not) yet another matcher for ontology matching task. In *BDA'2012: 28e journées Bases de Données Avancées*. pages N–A.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. Ppdb 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Association for Computational Linguistics, Beijing, China, pages 425–430. http://www.aclweb.org/anthology/P15-2070.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Anastasia Pentina and Shai Ben-David. 2015. Multi-task and lifelong learning of kernels. In *International Conference on Algorithmic Learning Theory*. Springer, pages 194–208.

Pavel Shvaiko and Jérôme Euzenat. 2008. Ten challenges for ontology matching. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer, pages 1164–1182.

Pavel Shvaiko and Jérôme Euzenat. 2013. Ontology matching: state of the art and future challenges. *IEEE Transactions on knowledge and data engineering* 25(1):158–176.

Arjen Stolk, Lennart Verhagen, and Ivan Toni. 2016. Conceptual alignment: How brains achieve mutual understanding. *Trends in cognitive sciences* 20(3):180–191.

Julia M Taylor. 2015. Mapping human understanding to robotic perception. *Procedia Computer Science* 56:514–519.

Peng Wang and Baowen Xu. 2008. Lily: Ontology alignment results for oaei 2008. In *Proceedings of the 3rd International Conference on Ontology Matching-Volume 431*. CEUR-WS. org, pages 167–175.

John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Dan Roth. 2015. From paraphrase database to compositional paraphrase model and back. *Transactions of the Association for Computational Linguistics* 3:345–358.

Chuncheng Xiang, Tingsong Jiang, Baobao Chang, and Zhifang Sui. 2015. Ersom: A structural ontology

matching approach using automatically learned entity representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 2419–2429. http://aclweb.org/anthology/D15-1289.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th conference on Computational linguistics-Volume 2*. Association for Computational Linguistics, pages 947–953.

Yuanzhe Zhang, Xuepeng Wang, Siwei Lai, Shizhu He, Kang Liu, Jun Zhao, and Xueqiang Lv. 2014. Ontology matching with word embeddings. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, Springer, pages 34–45.

Efficient Sequence Learning with Group Recurrent Networks

Fei Gao[1,2], **Lijun Wu**[3], **Li Zhao**[2], **Tao Qin**[2], **Xueqi Cheng**[1] and **Tie-Yan Liu**[2]
[1]Institute of Computing Technology, Chinese Academy of Sciences
[2]Microsoft Research
[3]School of Data and Computer Science, Sun Yat-sen University
{feiga,lizo,taoqin,tie-yan.liu}@microsoft.com;
wulijun3@mail2.sysu.edu.cn; cxq@ict.ac.cn

Abstract

Recurrent neural networks have achieved state-of-the-art results in many artificial intelligence tasks, such as language modeling, neural machine translation, speech recognition and so on. One of the key factors to these successes is big models. However, training such big models usually takes days or even weeks of time even if using tens of GPU cards. In this paper, we propose an efficient architecture to improve the efficiency of such RNN model training, which adopts the group strategy for recurrent layers, while exploiting the representation rearrangement strategy between layers as well as time steps. To demonstrate the advantages of our models, we conduct experiments on several datasets and tasks. The results show that our architecture achieves comparable or better accuracy comparing with baselines, with a much smaller number of parameters and at a much lower computational cost.

1 Introduction

Recurrent Neural Networks (RNNs) have been widely used for sequence learning, and achieved state-of-the-art results in many artificial intelligence tasks in recent years, including language modeling (Zaremba et al., 2014; Merity et al., 2017), neural machine translation (Sutskever et al., 2014; Bahdanau et al., 2014), and speech recognition (Graves et al., 2013).

To get better accuracy, recent state-of-the-art RNN models are designed toward big scale, include going deep (stacking multiple recurrent layers) (Pascanu et al., 2013a) and/or going wide (increasing dimensions of hidden states). For example, an RNN based commercial Neural Machine Translation (NMT) system would employ tens of layers in total, resulting in a large model with hundreds of millions of parameters (Wu et al., 2016). However, when the model size increases, the computational cost, as well as the memory needed for the training, increases dramatically. The training cost of aforementioned NMT model reaches as high as 10^{19} FLOPs, and the training procedure spends several days with even 96 GPU cards (Wu et al., 2016) – such complexity is prohibitively expensive.

While above models benefit from big neural networks, it is observed that such networks often have redundancy of parameters (Kim and Rush, 2016), motivating us to improve parameter efficiency and design more compact architectures that are more efficient in training while keeping good performance. Recently, many efficient architectures for convolution neural networks (CNNs) have been proposed to reduce training cost in computer vision domain. Among them, the group convolution is one of the most widely used and successful attempts (Szegedy et al., 2015; Chollet, 2016; Zhang et al., 2017b), which splits the channels into groups and conducts convolution separately for each group. It's essentially a diagonal sparse operation to the convolutional layer, which reduces the number of parameters as well as the computation complexity linearly w.r.t. the group size. Empirical results for such group convolution optimization show great speed up with small degradation on accuracy. In contrast, there are very limited attempts for designing better architectures for RNNs.

Inspired by those works on CNNs, in this paper, we generalize the group idea to RNNs to conduct recurrent learning in the group level. Different from CNNs, there are two kinds of parameter redundancy in RNNs: (1) the weight matrices transforming a low-level feature representation to a high-level one may contain redundancy, and (2) the recurrent weight matrices transferring the hidden state of the current step to the hidden state of the next step may also contain redundancy. Therefore, when designing efficient RNNs, we need to consider both the kinds of redundancy.

Proceedings of NAACL-HLT 2018, pages 799–808
New Orleans, Louisiana, June 1 - 6, 2018.

We present a simple architecture for efficient sequence learning which consists of *group recurrent* layers and *representation rearrangement* layers. First, in a recurrent layer, we split both the input of the sequence and the hidden states into disjoint groups, and do recurrent learning separately for each group. This operation clearly reduces the model complexity, and can learn *intra-group* features efficiently. However, it fails to capture dependency cross different groups. To recover the *inter-groups* correlation, we further introduce a representation rearrangement layer between any two consecutive recurrent layers, as well as any two time steps. With these two operations, we explicitly factorize a recurrent temporal learning into *intra-group* temporal learning and *inter-group* temporal learning with a much smaller number of parameters.

The group recurrent layer we proposed is equivalent to the standard recurrent layer with a block-diagonal sparse weight matrix. That is, our model employs a uniform sparse structure which can be computed very efficiently. To show the advantages of our model, we analyze the computation cost and memory usage comparing with standard recurrent networks. The efficiency improvement is linear to the number of groups. We conduct experiments on language modeling, neural machine translation and abstractive summarization by using a state-of-the-art RNN architecture as baseline. The results show that our model can achieve comparable or better accuracy, with a much smaller number of parameters and in a shorter training time.

The remainder of this paper is organized as follows. We first present our newly proposed architecture and conduct in depth analysis on its efficiency improvement. Then we show a series of empirical study to verify the effectiveness of our methods. Finally, to better position our work, we introduce some related work and then conclude our work.

2 Architecture

In this section, we introduce our proposed architecture for RNNs. Before getting into the details of the group recurrent layer and representation rearrangement layer in our architecture, we first revisit the vanilla RNNs.

An RNN is a neural network with recurrent layers that capture temporal dynamics of a sequence with arbitrary length. It recursively applies a transition function to its internal hidden state for each symbol of input sequence. The hidden state at time step t is computed as a function f of the current input symbol x_t and the previous hidden state h_{t-1} in a recurrent form:

$$h_t = f\left(x_t, h_{t-1}\right). \tag{1}$$

For vanilla RNN, the commonly used state-to-state transition function is,

$$h_t = tanh\left(Wx_t + Uh_{t-1}\right), \tag{2}$$

where W is the input-to-hidden weight matrix, U is the state-to-state recurrent weight matrix, and $tanh$ is the hyperbolic tangent function. Our work is independent to the choices of the recurrent function (f in Equation 1). For simplicity, in the following, we take the vanilla RNN as an example to introduce and analyze our new architecture.

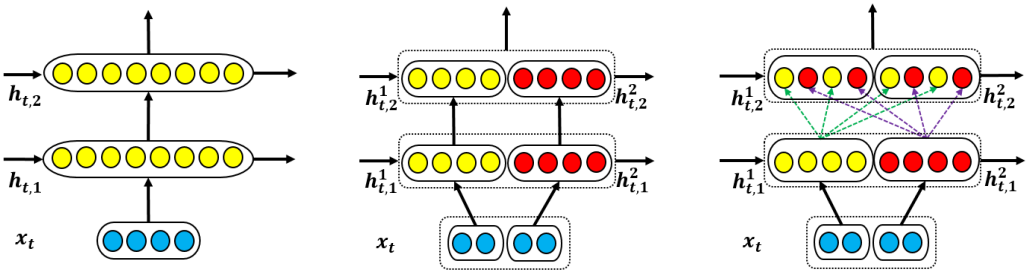
We aim to design an efficient RNN architecture by reducing the parameter redundancy while keeping accuracy at the same time. Inspired by the success of group convolution in CNN, our architecture employs the group strategy to achieve a sparsely connected structure between neurons of recurrent layers, and employs the representation rearrangement to recover the correlation that may destroyed by the sparsity. At a high level, we explicitly factorize the recurrent learning as inter-group recurrent learning and intra-group recurrent learning. In the following, we will describe our RNN architecture in detail, which consists of a group recurrent layer for intra-group correlation and a representation rearrangement layer for inter-group correlation.

2.1 Group recurrent layer for intra-group correlation

For standard recurrent layer, the model complexity increases quadratically with the dimension of hidden state. Suppose the input x is with dimension M, while the hidden state is with dimension N. Then, for standard vanilla RNN cell, according to Equation 2, the number of parameters, as well as the computation cost is

$$N^2 + N * M. \tag{3}$$

It's obvious that the hidden state dimension largely determines the model complexity. Optimization on reducing computation w.r.t the hidden state is the key to improve the overall efficiency. Accordingly, we present a group recurrent



(a) (b) (c)

Figure 1: Illustration of group recurrent network architecture. $h^k_{t,i}$ represents the hidden state of k-th group in i-th layer for time step t (a) The standard recurrent neural networks. (b) The group recurrent neural networks without representation rearrangement. This is efficient but the output only depends on the input in corresponding feature group. (c) Our proposed group recurrent neural network architecture, constituted with group recurrent layer and representation rearrangement layer.

layer which adopts a group strategy to approximate the standard recurrent layer. Specifically, we consider to split both the input x_t and hidden state h_t into K disjoint groups as $\{x^1_t, x^2_t, ..., x^K_t\}$ and $\{h^1_t, h^2_t, ..., h^K_t\}$ respectively, where x^i_t, h^i_t represent the input and hidden state for i-th group at time step t. Based on this split, we then perform recurrent computation in every group independently. This will captures the *intra-group* temporal correlation within the sequence. Formally, in the group recurrent layer, we first compute the hidden state of each group h^i_t as

$$h^i_t = f_i(x^i_t, h^i_{t-1}), i = 1, 2, ..., K. \quad (4)$$

Then, concatenating all the hidden states from each group together,

$$h_t = concat(h^1_t, h^2_t, ..., h^K_t) \quad (5)$$

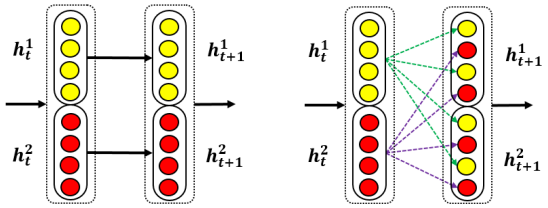
we get the output of the group recurrent layer. The group recurrent layer is illustrated as Figure 2(a) and Figure 1(b).

Obviously, by splitting the features and hidden states into K groups, the number of parameters and the computation cost of recurrent layer reduce to

$$K * ((\frac{N}{K})^2 + \frac{N}{K} * \frac{M}{K}) = \frac{N^2 + N * M}{K} \quad (6)$$

Comparing Equation 3 with Equation 6, the group recurrent is K times more efficient than the standard recurrent layer, in terms of both computational cost and number of parameters.

Although the theoretical computational cost is attractive, the speedup ratio also depends on the implementation details. A naive implementation of Equation 4 would introduce a *for* loop, which is not efficient since the additional overhead and poor parallelism. In order to really achieve linear speed up, we employ a *batch matrix multiplication* to assemble the computation of different groups in a single round of matrix multiplication. This operation is critical especially when each group isn't big enough to fully utilize the entire GPU computation power.



(a) (b)

Figure 2: Illustration of group recurrent network along the temporal direction. (a) The group recurrent neural network without representation rearrangement. (b) Our proposed group recurrent neural network with representation rearrangement.

2.2 Representation rearrangement for inter-group correlation

Group recurrent layer is K times more efficient comparing with the standard recurrent layer. But, it only captures the temporal correlation inside a single feature group and fails to learn dependency across features from different groups. more specifically, the internal state of RNN only contains history from corresponding group (Figure

1(b)). Similar problem also exists in the vertical direction of group recurrent layers (Figure 2(a)). Consider a network with multiple stacked group recurrent layers, the output of the specific group are only get from the corresponding input group. Obviously, there will be a significant drop of representation power since many feature correlations are cut off by this architecture.

To recover the *inter-group* correlations, one simple way is adding a projection layer to transform the hidden state outputted by the group recurrent layer, like the 1×1 convolution used in depthwise separable convolutional (Chollet, 2016). However, such method would bring additional N^2 computation complexity and model parameters.

Inspired by the idea of permuting channels between convolutional layers in recent CNN architectures (Zhang et al., 2017a,b), we propose to add representation rearrangement layer between consecutive group recurrent layers (Figure 1(c)), as well as the time steps within a group recurrent layer (Figure 2(b)). The representation rearrangement aims to rearrange the hidden representation, to make sure the subsequent layers, or time steps, can see features from all input groups.

The representation rearrangement layer is parameter-free and simple. We leverage the same implementation in (Zhang et al., 2017b) to conduct the rearrangement. It's finished with basic tensor operations *reshape*, and *transpose*, which brings (almost) no runtime overhead in our experiments. Consider the immediate representation $h_t \in R^N$ outputted by group recurrent layer with group number K. First, we reshape the representation to add an additional group dimension, resulting in a tensor with new shape $(K, N/K)$. Second, we transpose the two dimensions of the temporary tensor, changing the tensor shape to $(N/K, K)$. Finally, we reshape the tensor along the first axis to restore the representation to its original shape (a vector of size N). Figure 3 illustrates the operations with a simple example whose representation is with size 8 and group number is 2.

Combining the group recurrent layer and representation rearrangement layer, we rebuild the recurrent layer into an efficient and effective layer. We note that, different from convolutional neural networks that are only deep in space, the stack RNNs are deep in both space and time. Figure 1 illustrates our architecture along the spatial direction, and Figure 2 illustrates our architecture along the temporal direction. By applying group operation and representation rearrangement in both space and time, we build a new recurrent neural network with high efficiency.

3 Discussion

In this section, we analyze the relation between group recurrent layer and standard recurrent layer, and discuss the advantages of group recurrent networks.

3.1 Relation to standard recurrent layer

The group recurrent layer in Equation 4 and 5 can be re-formulated as

$$h_t = \tanh\Big(\begin{bmatrix} W^1 & 0 & \cdots & 0 \\ 0 & W^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & W^K \end{bmatrix} \begin{bmatrix} x_t^1 \\ x_t^2 \\ \vdots \\ x_t^K \end{bmatrix} + \begin{bmatrix} U^1 & 0 & \cdots & 0 \\ 0 & U^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & U^K \end{bmatrix} \begin{bmatrix} h_t^1 \\ h_t^2 \\ \vdots \\ h_t^K \end{bmatrix} \Big) \tag{7}$$

From the reformulation, we can see group recurrent layer is equivalent to standard recurrent layer with block-diagonal sparse weight matrix. Our method employs a group level sparsity in recurrent computation, leading to a uniform sparse structure. This uniform sparse structure can enjoy the efficient computing of dense matrix, as we discussed in Section 2.1. This reformulation also shows that there is no connection across neurons in different groups. Increasing the group number will lead to higher sparse rate. This sparse structure may limit the representation ability of our model. In order to recover the correlation across different groups, we add representation rearrangement to make up for representation ability.

3.2 Model capacity

We have shown that with same width of recurrent layer, our architecture with group number K achieves a compact model, which has K times less number of parameters than the standard recurrent network. Therefore with same number of parameters, group recurrent networks can provide more possibility to try more complex model without any

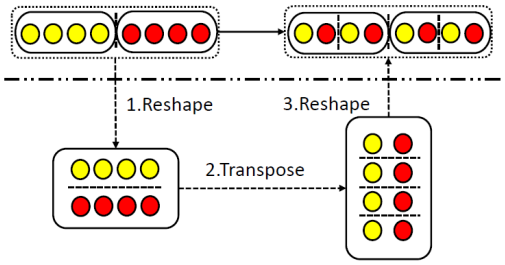


Figure 3: Illustration of the implementation of representation rearrangement with basic tensor operation.

additional computation and parameter overhead. Given a standard recurrent neural network, we can construct a corresponding group recurrent neural network with same number of parameters, but with K times wider, or with K times deeper. A factor smaller than K would make our networks still effective than standard recurrent network, but with wider and/or deeper recurrent layers. This could somehow compensate the potential performance drop due to the aggressive sparsity when group number is too large. Therefore, our architecture provides large model space to find a better trade-off between parameter and performance given a fixed resource budget. And our model is a more effective RNN architecture when the network goes deeper and wider.

At last, we note that our architecture focuses on improving the efficiency of recurrent layers. Thus the whole parameter and computational cost reduction depend on the ratio of recurrent layer in the entire network. Consider a text classification task, a often used RNN model would introduce an embedding layer for the input tokens and a softmax layer for the output, making the parameter reduction and speedup for the whole network is not strictly linear with the group number. However, we argue that for deeper and/or wider RNN whose recurrent layers dominate the parameter and computational cost, our method would enjoy more efficiency improvement.

4 Experiments

In this section, we present results on three sequence learning tasks to show the effectiveness of our method: 1). language modeling; 2). neural machine translation; 3). abstractive summarization.

4.1 Language modeling

For evaluating the effectiveness of our approach, we perform language modeling over Penn Treebak (PTB) dataset (Marcus et al., 1993). We use the data preprocessed by (Mikolov et al., 2010) [1], which consists of $929K$ training words, $73K$ validation words, and $82K$ test words. It has $10K$ words in its vocabulary. We compare our method (named Group LSTM) with the standard LSTM baseline (Zaremba et al., 2014) and its two variants with Bayesian dropout (named LSTM + BD) (Gal and Ghahramani, 2016) and with word tying (named LSTM + WT) (Press and Wolf, 2017). Following the big model settings in (Zaremba et al., 2014; Gal and Ghahramani, 2016; Inan et al., 2016) , all experiments use a two-layer LSTM with $1,500$ hidden units and an embedding of size $1,500$. We set group number 2 in this experiment since PTB is a relative simple dataset. We use Stochastic Gradient Descent (SGD) to train all models.

Results We compare the word level perplexity obtained by the standard LSTM baseline models and our group variants, in which we replace the standard LSTM layer with our group LSTM layer. As shown in Table 1, Group LSTM achieves comparable performance with the standard LSTM baseline, but with a 27% parameter reduction. A variant using Bayesian dropout (BD) is proposed by (Gal and Ghahramani, 2016) to prevent overfitting and improve performance. We test our model with LSTM + BD, achieving similar results with above comparison. Finally, we compare our model with the recently proposed word tying (WT) technology, which ties input embedding and output embedding with same weights. Our model achieves even better perplexity than the results reported by (Press and Wolf, 2017). Since word tying reduces the number of parameters of embedding and softmax layers, thus improving the ratio of LSTM layer parameter. Our method achieves a 35% parameter reduction.

4.2 Neural machine translation

We then study our model in neural machine translation. We conduct experiments on two translation tasks, German-English task (De-En for short) and English-German task (En-De for short). For De-En translation, we use data from the De-En ma-

[1] http://www.fit.vutbr.cz/~imikolov/rnnlm/simple-examples.tgz

Model	Parameters	Validation Set	Test Set
LSTM (Zaremba et al., 2014)	66M	82.2	78.4
2 Group LSTM	**48M**	**82.0**	**78.6**
LSTM + BD (Gal and Ghahramani, 2016)	66M	77.9	75.2
2 Group LSTM + BD	**48M**	**79.9**	**75.8**
LSTM + WT (Press and Wolf, 2017)	51M	77.4	74.3
2 Group LSTM + WT	**33M**	**76.8**	**73.3**
LSTM + BD + WT (Press and Wolf, 2017)	51M	75.8	73.2
2 Group LSTM + BD + WT	**33M**	**75.6**	**71.8**

Table 1: Single model complexity on validation and test sets for the Penn Treebank language modeling task. BD is Bayesian dropout. WT is word tying.

chine translation track of the IWSLT 2014 evaluation campaign (Cettolo et al., 2014). We follow the pre-processing described in previous works (Wu et al., 2017). The training data comprises about $153K$ sentence pairs. The size of validation data set is $6,969$, and the test set is $6,750$. For En-De translation, we use a widely adopted dataset (Jean et al., 2015; Wu et al., 2016). Specifically, part of data in WMT'14 is used as the training data, which consists of 4.5M sentences pairs. *newstest2012* and *newstest2013* are concatenated as the validation set and *newstest2014* acts as test set. These two datasets are preprocessed by byte pair encoding (BPE) with vocabulary of $25K$ and $30K$ for De-En and En-De respectively, and the max length of sub-word sentence is 64.

Our model is based on RNNSearch model (Bahdanau et al., 2014), but replacing the standard LSTM layer with our group LSTM layer. Therefore, we name our model as Group RNNSearch model. The model is constructed by LSTM encoder and decoder with attention, where the first layer of encoder is bidirectional LSTM. For De-En, we use two layers for both encoder and decoder. The embedding size is 256, which is same as the hidden size for all LSTM layers. As for En-De, we use four layers for encoder and decoder [2]. The embedding size is 512 and the hidden size is 1024 [3]. All the models are trained by Adadelta (Zeiler, 2012) with initial learning rate 1.0. The gradient is clipped with threshold 2.5. The mini-batch size is 32 for De-En and 128 for En-De. We use dropout (Srivastava et al., 2014) with rate 0.1 for all layers except the layer before softmax with 0.5. We halve the learning rate according to the validation performance.

[2]For easy to implement, we still keep the first layer with attention computation in the decoder as original LSTM layer.

[3]In our implementation, suppose the hidden size is d, after the first bi-directional LSTM layer in the encoder, the hidden size of the above LSTM layers in the encoder should be $2 \times d$.

Model	Params	BLEU
NPMT (Huang et al., 2017)	Unclear	30.08
RNNSearch	6.0M	31.03
2 Group RNNSearch	**4.3M**	**31.08**
4 Group RNNSearch	3.4M	30.96
8 Group RNNSearch	3.0M	30.73
16 Group RNNSearch	2.7M	30.35

Table 2: BLEU scores on IWSLT 2014 De-En test set. We report BLEU score results together with number of parameters of recurrent layers.

Model	Params	BLEU
DeepLAU (Wang et al., 2017)	Unclear	23.80
GNMT (Wu et al., 2016)	160M‡	24.61
2 Group RNNSearch	111M	23.93
4 Group RNNSearch	78M	23.61

Table 3: BLEU scores on WMT'14 En-De test set. We report BLEU score results together with number of parameters of recurrent layers. Numbers with ‡ are approximately calculated by ourselves according to the settings described in the paper.

Results We compute tokenized case-sensitive BLEU (Papineni et al., 2002) [4] score as evaluation metric. For decoding, we use beam search (Sutskever et al., 2014) with beam size 5.

From Table 2, we can observe that on De-En task, Group RNNSearch models achieve comparable or better BLEU score compared with the RNNSearch but with much less number of parameters. Specifically, with group number 2 and 4, we achieve about 28% and 43% parameter reduction of recurrent layers respectively. Note that our results also outperform the state-of-the-art result reported in NPMT (Huang et al., 2017).

The En-De translation results are shown in Table 3. We compare our Group RNNSearch models with Google's GNMT system (Wu et al., 2016) and DeepLAU (Wang et al., 2017). Our 4

[4]https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl

Group RNNSearch model achieves 23.61, which is comparable to DeepLAU (23.80). Our 2 Group RNNSearch model achieves a BLEU score of 23.93, slightly less than GNMT (24.61), but outperforms the DeepLAU. More importantly, our Group RNNSearch models decrease more than 30% and 50% RNN parameters with 2 groups and 4 groups respectively compared with GNMT.

4.3 Abstractive summarization

At last, we valid our approach on abstractive summarization task. We train on the Gigaword corpus (Graff and Cieri, 2003) and pre-process it identically to (Rush et al., 2015; Shen et al., 2016), resulting in $3.8M$ training article-headline pairs, $190K$ for validation and $2,000$ for test. Similar to (Shen et al., 2016), we use a source and target vocabulary consisting of $30K$ words.

The model is almost same as the one used in De-En machine translation, which is a two layers RNNSearch model, except that the embedding size is 512, and the LSTM hidden size in both encoder and decoder is 512. The initial values of all weight parameters are uniformly sampled between $(-0.05, 0.05)$. We train our Group RNNSearch model by Adadelta (Zeiler, 2012) with learning rate 1.0 and gradient clipping threshold 1.5 (Pascanu et al., 2013b). The mini-batch size is 64.

Results We evaluate the summarization task by commonly used ROUGE (Lin, 2004) F1 score. During decoding, we use beam search with beam size 10. The results are shown in Table 4.

From Table 4, we can observe that the performance is consistent with machine translation task. Our Group RNNSearch model achieves comparable results with RNNSearch, and our 2 Group RNNSearch model even outperforms RNNSearch baseline. Besides, we compare with several other widely adopted methods, our models also show strong performance. Therefore, we can keep the good performance even though we reduce the parameters of the recurrent layers by nearly 50%, which greatly proves the effectiveness of our method.

4.4 Ablation analysis

In addition to showing that group RNN can achieve competing or better performance with much less number of parameters, we further study the effect of group number to training speed and convergence, and the effect of representation rearrangement to performance. Due to space limitation, we only report results for language modeling on PTB dataset; for other tasks we have similar results.

Model	Params	R-1	R-2	R-L
(Rush et al., 2015)	-	29.8	11.9	26.9
(Luong et al., 2015)	-	33.1	14.4	30.7
(Chopra et al., 2016)	-	33.8	15.9	31.1
RNNSearch	24.1M	34.4	15.8	31.8
2 Group RNNSearch	**17.0M**	**34.8**	**15.9**	**32.1**
4 Group RNNSearch	13.5M	34.3	15.7	31.6
8 Group RNNSearch	11.8M	34.3	15.6	31.6
16 Group RNNSearch	10.9M	33.8	15.3	31.2

Table 4: ROUGE F1 scores on abstractive summarization test set. RG-N stands for N-gram based ROUGE F1 score, RG-L stands for longest common subsequence based ROUGE F1 score. Params stands for the parameters of the recurrent layers.

Group	without	with (improvement)
2	82.5	**78.6 (+4.7%)**
4	86.6	**82.6 (+4.6%)**

Table 5: The effect of representation rearrangement to model performance.

In Figure 4, the left one shows that how number of parameters and training speed vary when group number ranging from 1 to 16. We can see that the number of parameters (of recurrent layers) is reduced linearly when increasing number of groups. In the meantime, we also achieves substantial speed up about throughput when increasing group number. We note that the speedup is sub-linear instead of linear since our method focuses on the speedup on recurrent layers, as discussed in Section 3.2. Besides, we also compare the convergence curve in the right of Figure 4, which shows that our method (almost) doesn't slow down the convergence in terms of epoch number. Considering the throughput speedup of our method, we can accelerate training by a large margin.

At last, we study the role that representation rearrangement layer plays in our architecture. We compare Group LSTM with and without representation rearrangement between layers and time steps, with the group number 2 and 4 respectively. From Table 5, we can see that the models with representation rearrangement consistently outperforms the ones without representation rearrangement. This shows the representation rearrangement is critical for group RNN.

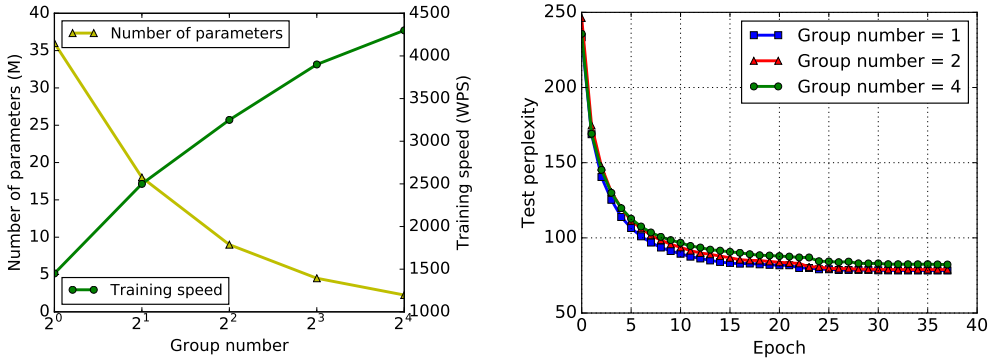


Figure 4: Illustration of group recurrent network analysis. Left: The number of parameters and the training speed (word per second, WPS) on different group numbers. Right: The test perplexity (convergence curve) along the training epochs.

5 Related Work

Improving RNN efficiency for sequence learning is a hot topic in recent deep learning research. For parameter and computation reduction, LightRNN (Li et al., 2016) is proposed to solve big vocabulary problem with a 2-component shared embedding, while our work addresses the parameter redundancy caused by recurrent layers. To speed up RNN, Persistent RNN (Diamos et al., 2016) is proposed to improve the RNN computation throughput by mapping deep RNN efficiently onto GPUs, which exploits GPU's inverted memory hierarchy to reuse network weights over multiple time steps. (Neil et al., 2017) proposes delta networks for optimizing the matrix-vector multiplications in RNN computation by considering the temporal properties of the data. Quasi-RNN (Bradbury et al., 2016) and SRU (Lei and Zhang, 2017) are proposed for speeding up RNN computation by designing novel recurrent units which relax dependency between time steps. Different from these works, we optimize RNN from the perspective of network architecture innovation by adopting a group strategy.

There is a long history about the group idea in deep learning, especially in convolutional neural networks, aiming to improve the computation efficiency and parameter efficiency. Such works can date back at least to AlexNet (Krizhevsky et al., 2012), which splits the convolutional layers into 2 independent groups for the ease of model-parallelism. The Inception (Szegedy et al., 2015) architecture proposes a module that employs uniform sparsity to improve the parameter efficiency. Going to the extreme of Inception, the Xception (Chollet, 2016) adopts a depthwise separable convolution, where each spatial convolution only works on a single channel. MobileNet (Howard et al., 2017) uses the same idea for efficient mobile model. IGCNet (Zhang et al., 2017a) and ShuffleNet (Zhang et al., 2017b) also adopt the group convolution idea, and further permute the features across consecutive layers. Similar to these works, we also exploit the group strategy. But we focus on efficient sequence learning with RNN, which, different from CNN, contains an internal memory and an additional temporal direction. In the RNN literature, there is only one paper (Kuchaiev and Ginsburg, 2017), to our best knowledge, exploiting the group strategy. However, this work assumes the features are group independent, thus failing to capturing the inter-group correlation. Our work employs a representational rearrangement mechanism, which avoids the assumption and improves the performance, as shown in our empirical experiments.

6 Conclusion

We have presented an efficient RNN architecture for sequence learning. Our architecture employs a group recurrent layer to learn intra-group correlation efficiently, and representation rearrangement layer to recover inter-group correlation for keeping representation ability. We demonstrate our model is more efficient in terms of parameters and computational cost. We conduct extensive experiments on language modeling, neural machine translations and abstractive summarization, showing that our method achieves competing performance with much less computing resource.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2016. Quasi-recurrent neural networks. *arXiv preprint arXiv:1611.01576* .

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th iwslt evaluation campaign, iwslt 2014.

François Chollet. 2016. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint arXiv:1610.02357* .

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *NAACL*. pages 93–98.

Greg Diamos, Shubho Sengupta, Bryan Catanzaro, Mike Chrzanowski, Adam Coates, Erich Elsen, Jesse Engel, Awni Hannun, and Sanjeev Satheesh. 2016. Persistent rnns: Stashing recurrent weights on-chip. In *International Conference on Machine Learning*. pages 2024–2033.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*. pages 1019–1027.

David Graff and C Cieri. 2003. English gigaword, linguistic data consortium.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, pages 6645–6649.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* .

Po-Sen Huang, Chong Wang, Dengyong Zhou, and Li Deng. 2017. Neural phrase-based machine translation. *CoRR* abs/1706.05565.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2016. Tying word vectors and word classifiers: A loss framework for language modeling. *arXiv preprint arXiv:1611.01462* .

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *ACL*.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1317–1327.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. pages 1097–1105.

Oleksii Kuchaiev and Boris Ginsburg. 2017. Factorization tricks for lstm networks. *arXiv preprint arXiv:1703.10722* .

Tao Lei and Yu Zhang. 2017. Training rnns as fast as cnns. *arXiv preprint arXiv:1709.02755* .

Xiang Li, Tao Qin, Jian Yang, Xiaolin Hu, and Tieyan Liu. 2016. Lightrnn: Memory and computation-efficient recurrent neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, Curran Associates, Inc., pages 4385–4393.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *ACL-04 workshop*. Barcelona, Spain.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

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

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and Optimizing LSTM Language Models. *arXiv preprint arXiv:1708.02182* .

Tomas Mikolov, Martin Karafit, Lukas Burget, Jan Cernock, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September*. pages 1045–1048.

Daniel Neil, Jun Haeng Lee, Tobi Delbruck, and Shih-Chii Liu. 2017. Delta networks for optimized recurrent network computation. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*. PMLR, International Convention Centre, Sydney, Australia, volume 70 of *Proceedings of Machine Learning Research*, pages 2584–2593.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*. Association for Computational Linguistics.

Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2013a. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026* .

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013b. On the difficulty of training recurrent neural networks. In *ICML*. pages 1310–1318.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, Valencia, Spain, pages 157–163.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685* .

Shiqi Shen, Yu Zhao, Zhiyuan Liu, Maosong Sun, et al. 2016. Neural headline generation with sentence-wise optimization. *arXiv preprint arXiv:1604.01904* .

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research* 15(1):1929–1958.

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

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 1–9.

Mingxuan Wang, Zhengdong Lu, Jie Zhou, and Qun Liu. 2017. Deep neural machine translation with linear associative unit. *arXiv preprint arXiv:1705.00861* .

Lijun Wu, Li Zhao, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2017. Sequence prediction with unlabeled data by reward function learning. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*. pages 3098–3104.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* .

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329* .

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

Ting Zhang, Guo-Jun Qi, Bin Xiao, and Jingdong Wang. 2017a. Primal-dual group convolutions for deep neural networks. *arXiv preprint arXiv:1707.02725* .

Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. 2017b. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *arXiv preprint arXiv:1707.01083* .

FEVER: a large-scale dataset for Fact Extraction and VERification

James Thorne[1], **Andreas Vlachos**[1], **Christos Christodoulopoulos**[2], and **Arpit Mittal**[2]

[1]Department of Computer Science, University of Sheffield
[2]Amazon Research Cambridge
{j.thorne, a.vlachos}@sheffield.ac.uk
{chrchrs, mitarpit}@amazon.co.uk

Abstract

In this paper we introduce a new publicly available dataset for verification against textual sources, FEVER: Fact Extraction and VERification. It consists of 185,445 claims generated by altering sentences extracted from Wikipedia and subsequently verified without knowledge of the sentence they were derived from. The claims are classified as SUPPORTED, REFUTED or NOTENOUGHINFO by annotators achieving 0.6841 in Fleiss κ. For the first two classes, the annotators also recorded the sentence(s) forming the necessary evidence for their judgment. To characterize the challenge of the dataset presented, we develop a pipeline approach and compare it to suitably designed oracles. The best accuracy we achieve on labeling a claim accompanied by the correct evidence is 31.87%, while if we ignore the evidence we achieve 50.91%. Thus we believe that FEVER is a challenging testbed that will help stimulate progress on claim verification against textual sources.

1 Introduction

The ever-increasing amounts of textual information available combined with the ease in sharing it through the web has increased the demand for verification, also referred to as fact checking. While it has received a lot of attention in the context of journalism, verification is important for other domains, e.g. information in scientific publications, product reviews, etc.

In this paper we focus on verification of textual claims against textual sources. When compared to textual entailment (TE)/natural language inference (Dagan et al., 2009; Bowman et al., 2015), the key difference is that in these tasks the passage to verify each claim is given, and in recent years it typically consists a single sentence, while in verification systems it is retrieved from a large set of documents in order to form the evidence. Another related task is question answering (QA), for which approaches have recently been extended to handle large-scale resources such as Wikipedia (Chen et al., 2017). However, questions typically provide the information needed to identify the answer, while information missing from a claim can often be crucial in retrieving refuting evidence. For example, a claim stating "Fiji's largest island is Kauai." can be refuted by retrieving "Kauai is the oldest Hawaiian Island." as evidence.

Progress on the aforementioned tasks has benefited from the availability of large-scale datasets (Bowman et al., 2015; Rajpurkar et al., 2016). However, despite the rising interest in verification and fact checking among researchers, the datasets currently used for this task are limited to a few hundred claims. Indicatively, the recently conducted Fake News Challenge (Pomerleau and Rao, 2017) with 50 participating teams used a dataset consisting of 300 claims verified against 2,595 associated news articles which is orders of magnitude smaller than those used for TE and QA.

In this paper we present a new dataset for claim verification, FEVER: Fact Extraction and VERification. It consists of 185,445 claims manually verified against the introductory sections of Wikipedia pages and classified as SUPPORTED, REFUTED or NOTENOUGHINFO. For the first two classes, systems and annotators need to also return the combination of sentences forming the necessary evidence supporting or refuting the claim (see Figure 1). The claims were generated by human annotators extracting claims from Wikipedia and mutating them in a variety of ways, some of which were meaning-altering. The verification of each

Proceedings of NAACL-HLT 2018, pages 809–819
New Orleans, Louisiana, June 1 - 6, 2018.

claim was conducted in a separate annotation process by annotators who were aware of the page but not the sentence from which original claim was extracted and thus in 31.75% of the claims more than one sentence was considered appropriate evidence. Claims require composition of evidence from multiple sentences in 16.82% of cases. Furthermore, in 12.15% of the claims, this evidence was taken from multiple pages.

To ensure annotation consistency, we developed suitable guidelines and user interfaces, resulting in inter-annotator agreement of 0.6841 in Fleiss κ (Fleiss, 1971) in claim verification classification, and 95.42% precision and 72.36% recall in evidence retrieval.

To characterize the challenges posed by FEVER we develop a pipeline approach which, given a claim, first identifies relevant documents, then selects sentences forming the evidence from the documents and finally classifies the claim w.r.t. evidence. The best performing version achieves 31.87% accuracy in verification when requiring correct evidence to be retrieved for claims SUPPORTED or REFUTED, and 50.91% if the correctness of the evidence is ignored, both indicating the difficulty but also the feasibility of the task. We also conducted oracle experiments in which components of the pipeline were replaced by the gold standard annotations, and observed that the most challenging part of the task is selecting the sentences containing the evidence. In addition to publishing the data via our website[1], we also publish the annotation interfaces[2] and the baseline system[3] to stimulate further research on verification.

2 Related Works

Vlachos and Riedel (2014) constructed a dataset for claim verification consisting of 106 claims, selecting data from fact-checking websites such as PolitiFact, taking advantage of the labelled claims available there. However, in order to develop claim verification components we typically require the justification for each verdict, including the sources used. While this information is usually available in justifications provided by the journalists, they are not in a machine-readable form. Thus, also considering the small number of claims, the task defined by the dataset proposed remains too challenging for the ML/NLP methods currently available. Wang (2017) extended this approach by including all 12.8K claims available by Politifact via its API, however the justification and the evidence contained in it was ignored in the experiments as it was not machine-readable. Instead, the claims were classified considering only the text and the metadata related to the person making the claim. While this rendered the task amenable to current NLP/ML methods, it does not allow for verification against any sources and no evidence needs to be returned to justify the verdicts.

Claim: The Rodney King riots took place in the most populous county in the USA.

[wiki/Los_Angeles_Riots]
The 1992 Los Angeles riots, also known as the Rodney King riots were a series of riots, lootings, arsons, and civil disturbances that occurred in Los Angeles County, California in April and May 1992.

[wiki/Los_Angeles_County]
Los Angeles County, officially the County of Los Angeles, is the most populous county in the USA.

Verdict: Supported

Figure 1: Manually verified claim requiring evidence from multiple Wikipedia pages.

The Fake News challenge (Pomerleau and Rao, 2017) modelled verification as stance classification: given a claim and an article, predict whether the article supports, refutes, observes (neutrally states the claim) or is irrelevant to the claim. It consists of 50K labelled claim-article pairs, combining 300 claims with 2,582 articles. The claims and the articles were curated and labeled by journalists in the context of the Emergent Project (Silverman, 2015), and the dataset was first proposed by Ferreira and Vlachos (2016), who only classified the claim w.r.t. the article headline instead of the whole article. Similar to recognizing textual entailment (RTE) (Dagan et al., 2009), the systems were provided with the sources to verify against, instead of having to retrieve them.

A differently motivated but closely related dataset is the one developed by Angeli and Manning (2014) to evaluate natural logic inference for common sense reasoning, as it evaluated sim-

[1] http://fever.ai
[2] https://github.com/awslabs/fever
[3] https://github.com/sheffieldnlp/fever-baselines

ple claims such as "not all birds can fly" against textual sources — including Wikipedia — which were processed with an Open Information Extraction system (Mausam et al., 2012). However, the claims were small in number (1,378) and limited in variety as they were derived from eight binary ConceptNet relations (Tandon et al., 2011).

Claim verification is also related to the multilingual Answer Validation Exercise (Rodrigo et al., 2009) conducted in the context of the TREC shared tasks. Apart from the difference in dataset size (1,000 instances per language), the key difference is that the claims being validated were answers returned to questions by QA systems. The questions and the QA systems themselves provide additional context to the claim, while in our task definition the claims are outside any particular context. In the same vein, Kobayashi et al. (2017) collected a dataset of 412 statements in context from high-school student exams that were validated against Wikipedia and history textbooks.

3 Fact extraction and verification dataset

The dataset was constructed in two stages[4] :

Claim Generation Extracting information from Wikipedia and generating claims from it.

Claim Labeling Classifying whether a claim is supported or refuted by Wikipedia and selecting the evidence for it, or deciding there's not enough information to make a decision.

3.1 Task 1 - Claim Generation

The objective of this task was to generate claims from information extracted from Wikipedia. We used the June 2017 Wikipedia dump, processed it with Stanford CoreNLP (Manning et al., 2014), and sampled sentences from the introductory sections of approximately 50,000 popular pages.[5]

The annotators were given a sentence from the sample chosen at random, and were asked to generate a set of **claims** containing a single piece of information, focusing on the entity that its original Wikipedia page was about. We asked the annotators to generate claims about a single fact which could be arbitrarily complex and allowed for a variety of expressions for the entities.

If only the source sentences were used to generate claims then this would result in trivially verifiable claims, as the new claims would in essence be simplifications and paraphrases. At the other extreme, if we allowed world knowledge to be freely incorporated it would result in claims that would be hard to verify on Wikipedia alone. We address this issue by introducing a **dictionary**: a list of terms that were (hyper-)linked in the original sentence, along with the first sentence from their corresponding Wikipedia pages. Using this dictionary, we provide additional knowledge that can be used to increase the complexity of the generated claims in a controlled manner.

The annotators were also asked to generate **mutations** of the claims: altered versions of the original claims, which may or may not change whether they are supported by Wikipedia, or even if they can be verified against it. Inspired by the operators used in Natural Logic Inference (Angeli and Manning, 2014), we specified six types of mutation: paraphrasing, negation, substitution of an entity/relation with a similar/dissimilar one, and making the claim more general/specific.

During trials of the annotation task, we discovered that the majority of annotators had difficulty generating non-trivial negation mutations (e.g. mutations beyond adding "not" to the original). Besides providing numerous examples for each mutation, we also redesigned the annotation interface so that all mutation types were visible at once and highlighted mutations that contained "not" in order to discourage trivial negations. Finally, we provided the annotators with an ontology diagram to illustrate the different levels of entity similarity and class membership.

This process resulted in claims (both extracted and mutated) with a mean length of 9.4 tokens which is comparable to the average hypothesis length of 8.3 tokens in Bowman et al. (2015).

3.2 Task 2 - Claim Labeling

The annotators were asked to label each individual claim generated during Task 1 as SUPPORTED, REFUTED or NOTENOUGHINFO. For the first two cases, the annotators were asked to find the evidence from any page that supports or refutes the claim (see Figure 2 for a screenshot of the interface). In order to encourage inter-annotator consistency, we gave the following general guideline:

[4]The annotation guidelines for both stages are provided in the supplementary materials

[5]These consisted of 5,000 from a Wikipedia 'most accessed pages' list and the pages hyperlinked from them.

> If I was given only the selected sentences, do I have strong reason to believe the claim is true (supported) or stronger reason to believe the claim is false (refuted). If I'm not certain, what additional information (dictionary) do I have to add to reach this conclusion.

In the annotation interface, all sentences from the introductory section of the page for the main entity of the claim and of every linked entity in those sentences were provided as a default source of evidence (left-hand side in Fig. 2). Using this interface the annotators recorded the sentences necessary to justify their classification decisions. In order to allow exploration beyond the main and linked pages, we also allowed annotators to add an arbitrary Wikipedia page by providing its URL and the system would add its introductory section as additional sentences that could be then selected as evidence (right-hand side in Fig. 2). The title of the page could also be used as evidence to resolve co-reference, but this decision was not explicitly recorded. We did not set a hard time limit for the task, but the annotators were advised not to spend more than 2-3 minutes per claim. The label NOTENOUGHINFO was used if the claim could not be supported or refuted by any amount of information in Wikipedia (either because it is too general, or too specific).

3.3 Annotators

The annotation team consisted of a total of 50 members, 25 of which were involved only in the first task. All annotators were native US English speakers and were trained either directly by the authors, or by experienced annotators. The interface for both tasks was developed by the authors in collaboration with an initial team of two annotators. Their notes and suggestions were incorporated into the annotation guidelines.

The majority of the feedback received from the annotators was very positive: they found the task engaging and challenging, and after the initial stages of annotation they had developed an understanding of the needs of the task which let them discuss solutions about edge cases as a group.

3.4 Data Validation

Given the complexity of the second task (claim labeling), we conducted three forms of data validation: 5-way inter-annotator agreement, agreement against *super-annotators* (defined in Section 3.4.2), and manual validation by the authors. The validation for claim generation was done implicitly during claim labeling. As a result 1.01% of claims were skipped, 2.11% contained typos and 6.63% of the generated claims were flagged as too vague/ambiguous and were excluded e.g. or "Sons of Anarchy premiered.".

3.4.1 5-way Agreement

We randomly selected 4% ($n = 7506$) of claims which were not skipped to be annotated by 5 annotators. We calculated the Fleiss κ score (Fleiss, 1971) to be 0.6841 which we consider encouraging given the complexity of the task. In comparison Bowman et al. (2015) reported a κ of 0.7 for a simpler task, since the annotators were given the premise/evidence to verify a hypothesis against without the additional task of finding it.

3.4.2 Agreement against *Super-Annotators*

We randomly selected 1% of the data to be annotated by *super-annotators*: expert annotators with no suggested time restrictions. The purpose of this exercise was to provide as much coverage of evidence as possible. We instructed the *super-annotators* to search over the whole Wikipedia for every possible sentence that could be used as evidence. We compared the regular annotations against this set of evidence and the precision/recall was 95.42% and 72.36% respectively.

3.4.3 Validation by the Authors

As a final quality control step, we chose 227 examples and annotated them for accuracy of the labels and the evidence provided. We found that 91.2% of the examples were annotated correctly. 3% of the claims were mistakes in claim generation that had not been flagged during labeling. We found a similar number of these claims which did not meet the guidelines during a manual error analysis of the baseline system (Section 5.8).

3.4.4 Findings

When compared against the *super-annotators*, all except two annotators achieved $> 90\%$ precision and all but 9 achieved recall $> 70\%$ in evidence retrieval. The majority of the low-recall cases are for claims such as "Akshay Kumar is an actor." where the *super-annotator* added 34 sentences as evidence, most of them being filmography listings (e.g. "In 2000, he starred in the Priyadarshan-directed comedy Hera Pheri").

Figure 2: Screenshot of Task 2 - Claim Labeling

During the validation by the authors, we found that most of the examples that were annotated incorrectly were cases where the label was correct, but the evidence selected was not sufficient (only 4 out of 227 examples were labeled incorrectly according to the guidelines).

We tried to resolve this issue by asking our annotators to err on the side of caution. For example, while the claim "Shakira is Canadian" could be labeled as REFUTED by the sentence "Shakira is a Colombian singer, songwriter, dancer, and record producer", we advocated that unless more explicit evidence is provided (e.g. "She was denied Canadian citizenship"), the claim should be labeled as NOTENOUGHINFO, since dual citizenships are permitted, and the annotators' world knowledge should not be factored in.

A related issue is entity resolution. For a claim like "David Beckham was with United.", it might be trivial for an annotator to accept "David Beckham made his European League debut playing for Manchester United." as supporting evidence. This implicitly assumes that "United" refers to "Manchester United", however there are many Uniteds in Wikipedia and not just football clubs, e.g. United Airlines. The annotators knew the page of the main entity and thus it was relatively easy to resolve ambiguous entities. While we provide this information as part of the dataset, we argue that it should only be used for system training/development.

4 Baseline System Description

We construct a simple pipelined system comprising three components: document retrieval, sentence-level evidence selection and textual entailment. Each component is evaluated in isolation through oracle evaluations on the development set and we report the final accuracies on the test set.

Document Retrieval We use the document retrieval component from the DrQA system (Chen et al., 2017) which returns the k nearest documents for a query using cosine similarity between binned unigram and bigram Term Frequency-Inverse Document Frequency (TF-IDF) vectors.

Sentence Selection Our simple sentence selection method ranks sentences by TF-IDF similarity to the claim. We sort the most-similar sentences first and tune a cut-off using validation accuracy on the development set. We evaluate both DrQA and a simple unigram TF-IDF implementation to rank the sentences for selection. We further evaluate impact of sentence selection on the RTE module by predicting entailment given the original documents without sentence selection.

Recognizing Textual Entailment We compare two models for recognizing textual entailment. For a simple well-performing baseline, we selected Riedel et al. (2017)'s submission from the 2017 Fake News Challenge. It is a multi-layer perceptron (**MLP**) with a single hidden layer which uses term frequencies and TF-IDF cosine similarity between the claim and evidence as features.

Evaluating the state-of-the-art in RTE, we used a decomposable attention (**DA**) model between the claim and the evidence passage (Parikh et al., 2016). We selected it because at the time of development this model was the highest scoring system for the Stanford Natural Language Inference task (Bowman et al., 2015) with publicly available code that did not require the input text to be parsed syntactically, nor was an ensemble.

The RTE component must correctly classify a claim as NOTENOUGHINFO when the evidence retrieved is not relevant or informative. However, the instances labeled as NOTENOUGHINFO have no evidence annotated, thus cannot be used to train RTE for this class. To overcome this issue, we simulate training instances for the NOTENOUGHINFO through two methods: sampling a sentence from the nearest page (NEARESTP) to the claim as evidence using our document retrieval component and sampling a sentence from Wikipedia uniformly at random (RANDOMS).

5 Experiments

5.1 Dataset Statistics

We partitioned the annotated claims into training, development and test sets. We ensured that each Wikipedia page used to generate claims occurs in exactly one set. We reserved a further 19,998 examples for use as a test set for a shared task.

Split	SUPPORTED	REFUTED	NEI
Training	80,035	29,775	35,639
Dev	3,333	3,333	3,333
Test	3,333	3,333	3,333
Reserved	6,666	6,666	6,666

Table 1: Dataset split sizes for SUPPORTED, REFUTED and NOTENOUGHINFO (NEI) classes

5.2 Evaluation

Predicting whether a claim is SUPPORTED, REFUTED or NOTENOUGHINFO is a 3-way classification task that we evaluate using accuracy. In the case of the first two classes, appropriate evidence must be provided, at a sentence-level, to justify the classification. We consider an answer returned correct for the first two classes only if correct evidence is returned. Given that the development and test datasets have balanced class distributions, a random baseline will have $\sim 33\%$ accuracy if one ignores the requirement for evidence for SUPPORTED and REFUTED.

We evaluate the correctness of the evidence retrieved by computing the F_1-score of all the predicted sentences in comparison to the human-annotated sentences for those claims requiring evidence on our complete pipeline system (Section 5.7). As in Fig. 1, some claims require multi-hop inference involving sentences from more than one document to be correctly supported as SUPPORTED/REFUTED. In this case all sentences must be selected for the evidence to be marked as correct. We report this as the proportion of *fully supported claims*. Some claims may be equally supported by different pieces of evidence; in this case one complete set of sentences should be predicted.

Systems that select information that the annotators did not will be penalized in terms of precision. We recognize that it is not feasible to ensure that the evidence selection annotations are complete, nevertheless we argue that they are useful for automatic evaluation during system development. For a more reliable evaluation we advocate crowd-sourcing annotations of false-positive predictions at a later date in a similar manner to the TAC KBP Slot Filler Validation (Ellis et al., 2016).

5.3 Document Retrieval

The document retrieval component of the baseline system returns the k nearest documents to the claim using the DrQA (Chen et al., 2017) TF-IDF implementation to return the k-nearest documents. In the scenario where evidence from multiple documents is required, k must be greater than this figure. We simulate the upper bound in accuracy using an oracle 3-way RTE classifier that predicts SUPPORTED/REFUTED ones correctly only if the documents containing the supporting/refuting evidence are returned by document retrieval and always predicts NOTENOUGHINFO instances correctly independently of the evidence. Results are shown in Table 2.

5.4 Sentence Selection

Mirroring document retrieval, we extract the top l-most similar sentences from the k-most relevant documents using TF-IDF vector similarity. We modified document retrieval component of DrQA (Chen et al., 2017) to select sentences using bigram TF-IDF with binning and compared this to a simple unigram TF-IDF implementation using NLTK (Loper and Bird, 2002). Using the param-

k	Fully Supported (%)	Oracle Accuracy (%)
1	25.31	50.21
5	55.30	70.20
10	65.86	77.24
25	75.92	83.95
50	82.49	90.13
100	86.59	91.06

Table 2: Dev. set document retrieval evaluation.

eters $k = 5$ documents and $l = 5$ sentences, 55.30% of claims (excluding NOTENOUGHINFO) can be fully supported or refuted by the retrieved documents before sentence selection (see Table 2). After applying the sentence selection component, 44.22% of claims can be fully supported using the extracted sentences with DrQA and only 34.03% with NLTK. This would yield oracle accuracies of 62.81% and 56.02% respectively.

5.5 Recognizing Textual Entailment

The RTE component is trained on labeled claims paired with sentence-level. Where multiple sentences are required as evidence, the strings are concatenated. As discussed in Section 4, such data is not annotated for claims labeled NOTENOUGHINFO, thus we compare random sampling-based and similarity-based strategies for generating it. We evaluate classification accuracy on the development set in an oracle evaluation, assuming correct evidence sentences are selected (Table 3). Additionally, for the DA model, we predict entailment given evidence, using the AllenNLP (Gardner et al., 2017) pre-trained Stanford Natural Language Inference (SNLI) model for comparison.

Model	Accuracy (%)		
	NEARESTP	RANDOMS	SNLI
MLP	65.13	73.81	-
DA	80.82	88.00	38.54

Table 3: Oracle classification on claims in the development set using gold sentences as evidence

The random sampling (RANDOMS) approach (where a sentence is sampled at random from Wikipedia in place of evidence for claims labeled as NOTENOUGHINFO) yielded sentences that were not only semantically different to the claim, but also unrelated. While the the accuracy of models trained with sampling approach is higher in oracle evaluation setting, this may not yield a better system in the pipeline setting. In contrast, the nearest page (NEARESTP) method samples a sentence from the highest-ranked page returned by our document retrieval module. This simulates finding related information that may not be sufficient to support or refute a claim. We will evaluate both RANDOMS and NEARESTP in the full pipeline setting, but we will not pursue the SNLI-trained model further as it performed substantially worse.

5.6 Full Pipeline

The complete pipeline consists of the DrQA document retrieval module (Section 5.3), DrQA-based sentence retrieval module (Section 5.4), and the decomposable attention RTE model (Section 5.5). The two parameters: k, describing the number documents and l, describing the number sentences to return were found using grid-search optimizing the RTE accuracy with the DA model. For the pipeline, we set $k = 5$ and $l = 5$ and report the development set accuracy, both with and without the requirement to provide correct evidence for the SUPPORTED/REFUTED predictions (marked as **ScoreEv** and **NoScoreEv** respectively).

Model	Accuracy (%)	
	NoScoreEv	ScoreEv
MLP / NP	41.86	19.04
MLP / RS	40.63	19.42
DA / NP	52.09	**32.57**
DA / RS	50.37	23.53

Table 4: Full pipeline results on development set

The decomposable attention model trained with NEARESTP is the most accurate when evidence is considered. Inspection of the confusion matrices shows that the RANDOMS strategy harms recall for the NOTENOUGHINFO class. This is due to the difference between the sampled pages in the training set and the ones retrieved in the development set causing related but uninformative evidence to be misclassified as SUPPORTED and REFUTED.

Ablation of the sentence selection module We evaluate the impact of the sentence selection module on both the RTE accuracy by removing it.

While the sentence selection module may improve accuracy in the RTE component, it is discarding sentences that are required as evidence to support claims, harming performance (see Section 5.4). We assess the accuracies in both oracle setting (similar to Section 5.5) (see Table 5) as well as in the full pipeline (see Table 6).

In the oracle setting, the decomposable attention models are worst affected by removal of the sentence selection module: exhibiting an substantial decrease in accuracy. The NEARESTP training regime exhibits a 17% decrease and the RANDOMS accuracy decreases by 19%, despite near-perfect recall of the NOTENOUGHINFO class.

Model	Oracle Accuracy (%)	
	NEARESTP	RANDOMS
MLP	57.16	73.36
DA	63.68	69.05

Table 5: Oracle accuracy on claims in the dev. set using gold documents as evidence (c.f. Table 3).

In the pipeline setting, we run the RTE component without sentence selection using $k = 5$ most similar predicted documents. The removal of the sentence selection component decreased the accuracy (NOSCOREEV) approximately 10% for both decomposable attention models.

Model	Accuracy (%)	
	NEARESTP	RANDOMS
MLP	38.85	40.45
DA	41.57	40.62

Table 6: Pipeline accuracy on the dev. set without the sentence selection module (c.f. Table 4).

5.7 Evaluating Full Pipeline on Test Set

We evaluate our pipeline approach on the test set based on the results observed in Section 5.6. First, we use DrQA to select select 5 documents nearest to the claim. Then, we select 5 sentences using our DrQA-based sentence retrieval component and concatenate them. Finally, we predict entailment using the Decomposable Attention model trained with the NEARESTP strategy. The classification accuracy is 31.87%. Ignoring the requirement for correct evidence (**NoScoreEv**) the accuracy is 50.91%, which highlights that while the systems were predicting the correct label, the evidence selected was different to that which the human annotators chose. The recall of the document and sentence retrieval modules for claims which required evidence on the test set was 45.89% (considering complete groups of evidence) and the precision 10.79%. The resulting F_1 score is 17.47%.

5.8 Manual Error Analysis

Using the predictions on the test set, we sampled 961 of the predictions with an incorrect label or incorrect evidence and performed a manual analysis. Of these, 28.51% ($n = 274$) had the correct predicted label but did not satisfy the requirements for evidence. The information retrieval component of the pipeline failed to identify any correct evidence in 58.27% ($n = 560$) of cases which accounted for the large disparity between accuracy of the system when evidence was and was not considered. Where suitable evidence was found, the RTE component incorrectly classified 13.84% ($n = 133$) of claims.

The pipeline retrieved new evidence that had not been identified by annotators in 21.85% ($n = 210$) of claims. This was in-line with our expectation given the measured recall rate of annotators (see Section 3.4.2), who achieved recall of 72.36% of evidence identified by the super-annotators.

We found that 4.05% ($n = 41$) of claims did not meet our guidelines. Of these, there were 11 claims which could be checked without evidence as these either tautologous or self-contradictory. Some correct claims appeared ungrammatical due to the mis-parsing of named entities (e.g. *Exotic Birds* is the name of a band but could be parsed as a type of animal). Annotator errors (where the wrong label was applied) were present in 1.35% ($n = 13$) of incorrectly classified claims.

Interestingly, our system found new evidence that contradicted the gold evidence in 0.52% ($n = 5$) of cases. This was caused either by entity resolution errors or by inconsistent information present in Wikipedia pages (e.g. Pakistan was described as having both the 41st and 42nd largest GDP in two different pages).

5.9 Ablation of Training Data

To evaluate whether the size of the dataset is suitable for training the RTE component of the pipeline, we plot the learning curves for the DA and MLP models (Fig. 3). For each model, we

trained 5 models with different random initializations using the NEARESTP method (see Section 5.5). We selected the highest performing model when evaluated on development set and report the oracle RTE accuracy on the test set. We observe that with fewer than 6000 training instances, the accuracy of DA is unstable. However, with more data, its accuracy increases with respect to the log of the number of training instances and exceeds that of MLP. While both learning curves exhibit the typical diminishing return trends, they indicate that the dataset is large enough to demonstrate the differences of models with different learning capabilities.

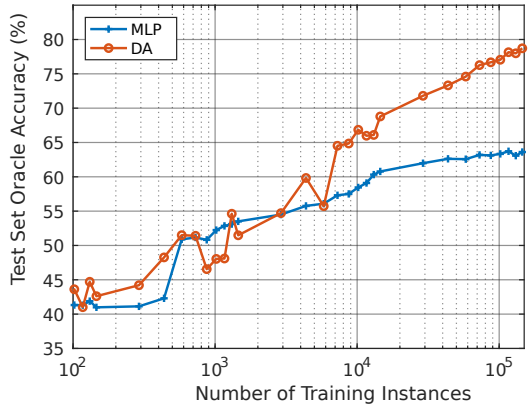


Figure 3: Learning curves for the RTE models.

6 Discussion

The pipeline presented and evaluated in the previous section is one possible approach to the task proposed in our dataset, but we envisage different ones to be equally valid and possibly better performing. For instance, it would be interesting to test how approaches similar to natural logic inference (Angeli and Manning, 2014) can be applied, where a knowledge base/graph is constructed by reading the textual sources and then a reasoning process over the claim is applied, possibly using recent advances in neural theorem proving (Rocktäschel and Riedel, 2017). A different approach could be to consider a combination of question generation (Heilman and Smith, 2010) followed by a question answering model such as BiDAF (Seo et al., 2016), possibly requiring modification as they are designed to select a single span of text from a document rather than return one or more sentences as per our scoring criteria. The sentence-level evidence annotation in our dataset will help develop models selecting and attending to the relevant information from multiple documents and non-contiguous passages. Not only will this enhance the interpretability of predictions, but also facilitate the development of new methods for reading comprehension.

Another use case for the FEVER dataset is claim extraction: generating short concise textual facts from longer encyclopedic texts. For sources like Wikipedia or news articles, the sentences can contain multiple individual claims, making them not only difficult to parse, but also hard to evaluate against evidence. During the construction on the FEVER dataset, we allowed for an extension of the task where simple claims can be extracted from multiple complex sentences.

Finally, we would like to note that while we chose Wikipedia as our textual source, we do not consider it to be the only source of information worth considering in verification, hence not using TRUE or FALSE in our classification scheme. We expect systems developed on the dataset presented to be portable to different textual sources.

7 Conclusions

In this paper we have introduced FEVER, a publicly available dataset for fact extraction and verification against textual sources. We discussed the data collection and annotation methods and shared some of the insights obtained during the annotation process that we hope will be useful to other large-scale annotation efforts.

In order to evaluate the challenge this dataset presents, we developed a pipeline approach that comprises information retrieval and textual entailment components. We showed that the task is challenging yet feasible, with the best performing system achieving an accuracy of 31.87%.

We also discussed other uses for the FEVER dataset and presented some further extensions that we would like to work on in the future. We believe that FEVER will provide a stimulating challenge for claim extraction and verification systems.

Acknowledgments

The work reported was partly conducted while James Thorne was at Amazon Research Cambridge. Andreas Vlachos is supported by the EU H2020 SUMMA project (grant agreement number 688139). The authors would like to thank the team of annotators involved in preparing this dataset.

References

Gabor Angeli and Christopher D. Manning. 2014. NaturalLI: Natural logic inference for common sense reasoning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. pages 534–545.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1870–1879. https://doi.org/10.18653/v1/P17-1171.

Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. 2009. Recognizing textual entailment: Rational, evaluation and approaches. *Natural Language Engineering* 15(4):i–xvii. https://doi.org/10.1017/S1351324909990209.

Joe Ellis, Jeremy Getman, Dana Fore, Neil Kuster, Zhiyi Song, Ann Bies, and Stephanie Strassel. 2016. Overview of Linguistic Resources for the TAC KBP 2016 Evaluations : Methodologies and Results. *Proceedings of TAC KBP 2016 Workshop, National Institute of Standards and Technology, Maryland, USA* (Ldc).

William Ferreira and Andreas Vlachos. 2016. Emergent: a novel data-set for stance classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California, pages 1163–1168.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin* 76(5):378.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2017. AllenNLP: A Deep Semantic Natural Language Processing Platform .

Michael Heilman and Noah A. Smith. 2010. Good Question! statistical ranking for question generation. In *Proceedings of the 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. pages 609–617.

Mio Kobayashi, Ai Ishii, Chikara Hoshino, Hiroshi Miyashita, and Takuya Matsuzaki. 2017. Automated historical fact-checking by passage retrieval, word statistics, and virtual question-answering. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. volume 1, pages 967–975.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, ETMTNLP '02, pages 63–70. https://doi.org/10.3115/1118108.1118117.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL (System Demonstrations)*. pages 55–60.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. pages 523–534.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 2249–2255. https://aclweb.org/anthology/D16-1244.

Dean Pomerleau and Delip Rao. 2017. Fake news challenge. `http://fakenewschallenge.org/`.

P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Benjamin Riedel, Isabelle Augenstein, George Spithourakis, and Sebastian Riedel. 2017. A simple but tough-to-beat baseline for the Fake News Challenge stance detection task. *CoRR* abs/1707.03264. http://arxiv.org/abs/1707.03264.

Tim Rocktäschel and Sebastian Riedel. 2017. End-to-end differentiable proving. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, California, United States*. volume abs/1705.11040. http://arxiv.org/abs/1705.11040.

Álvaro Rodrigo, Anselmo Peñas, and Felisa Verdejo. 2009. Overview of the answer validation exercise 2008. In Carol Peters, Thomas Deselaers, Nicola Ferro, Julio Gonzalo, Gareth J. F. Jones, Mikko Kurimo, Thomas Mandl, Anselmo Peñas, and Vivien Petras, editors, *Evaluating Systems for Multilingual and Multimodal Information Access: 9th Workshop of the Cross-Language Evaluation Forum*. pages 296–313.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *CoRR* abs/1611.01603. http://arxiv.org/abs/1611.01603.

Craig Silverman. 2015. Lies, Damn Lies and Viral Content. http://towcenter.org/research/lies-damn-lies-and-viral-content/.

Niket Tandon, Gerard de Melo, and Gerhard Weikum. 2011. Deriving a Web-scale common sense fact database. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI 2011)*. AAAI Press, Palo Alto, CA, USA, pages 152–157.

Andreas Vlachos and Sebastian Riedel. 2014. Fact checking: Task definition and dataset construction. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*. ACL. http://www.aclweb.org/anthology/W14-2508.

William Yang Wang. 2017. "Liar, Liar Pants on Fire": A new benchmark dataset for fake news detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. http://aclweb.org/anthology/P17-2067.

Global Relation Embedding for Relation Extraction

Yu Su*, **Honglei Liu***, **Semih Yavuz**, **Izzeddin Gür**
University of California, Santa Barbara
{ysu,honglei,syavuz,izzeddingur}@cs.ucsb.edu

Huan Sun
The Ohio State University
sun.397@osu.edu

Xifeng Yan
University of California, Santa Barbara
xyan@cs.ucsb.edu

Abstract

We study the problem of textual relation embedding with distant supervision. To combat the wrong labeling problem of distant supervision, we propose to embed textual relations with *global statistics* of relations, i.e., the co-occurrence statistics of textual and knowledge base relations collected from the entire corpus. This approach turns out to be more robust to the training noise introduced by distant supervision. On a popular relation extraction dataset, we show that the learned textual relation embedding can be used to augment existing relation extraction models and significantly improve their performance. Most remarkably, for the top 1,000 relational facts discovered by the best existing model, the precision can be improved from 83.9% to 89.3%.

1 Introduction

Relation extraction requires deep understanding of the relation between entities. Early studies mainly use hand-crafted features (Kambhatla, 2004; Zhou et al., 2005), and later kernel methods are introduced to automatically generate features (Zelenko et al., 2003; Culotta and Sorensen, 2004; Bunescu and Mooney, 2005; Zhang et al., 2006). Recently neural network models have been introduced to embed words, relations, and sentences into continuous feature space, and have shown a remarkable success in relation extraction (Socher et al., 2012; Zeng et al., 2014; Xu et al., 2015b; Zeng et al., 2015; Lin et al., 2016). In this work, we study the problem of embedding *textual relations*, defined as the shortest dependency path[1] between two entities in the dependency graph of a sentence, to improve relation extraction.

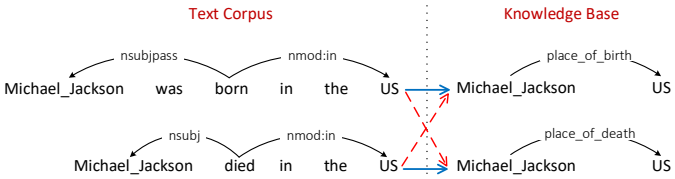
Textual relations are one of the most discriminative textual signals that lay the foundation of many relation extraction models (Bunescu and Mooney, 2005). A number of recent studies have explored textual relation embedding under the supervised setting (Xu et al., 2015a,b, 2016; Liu et al., 2016), but the reliance on supervised training data limits their scalability. In contrast, we embed textual relations with *distant supervision* (Mintz et al., 2009), which provides much larger-scale training data without the need of manual annotation. However, the assertion of distant supervision, "*any* sentence containing a pair of entities that participate in a knowledge base (KB) relation is likely to express the relation," can be violated more often than not, resulting in many wrongly labeled training examples. A representative example is shown in Figure 1. Embedding quality is thus compromised by the noise in training data.

Our main contribution is a novel way to combat the wrong labeling problem of distant supervision. Traditional embedding methods (Xu et al., 2015a,b, 2016; Liu et al., 2016) are based on *local statistics*, i.e., individual textual-KB relation pairs like in Figure 1 (Left). Our key hypothesis is that *global statistics is more robust to noise than local statistics*. For individual examples, the relation label from distant supervision may be wrong from time to time. But when we zoom out to consider the entire corpus, and collect the global co-occurrence statistics of textual and KB relations, we will have a more comprehensive view of relation semantics: The semantics of a textual relation can then be represented by its co-occurrence distribution of KB relations. For example, the distribution in Figure 1 (Right) indicates that the textual relation SUBJECT $\xleftarrow{\text{nsubjpass}}$ *born* $\xrightarrow{\text{nmod:in}}$ OBJECT mostly means `place_of_birth`, and is also a good indicator of `nationality`, but not `place_of_death`. Although it is still wrongly labeled with `place_of_death` a number of times, the negative impact becomes negligible. Similarly,

* Equally contributed.

[1] We use fully lexicalized shortest dependency path with directional and typed dependency relations.

Proceedings of NAACL-HLT 2018, pages 820–830
New Orleans, Louisiana, June 1 - 6, 2018.



	$\xleftarrow{\text{nsubjpass}}$ *born* $\xrightarrow{\text{nmod:in}}$	$\xleftarrow{\text{nsubj}}$ *died* $\xrightarrow{\text{nmod:in}}$
place_of_birth	1868	14
nationality	389	20
place_of_death	37	352
...	...	...

Figure 1: The wrong labeling problem of distant supervision, and how to combat it with global statistics. *Left*: conventional distant supervision. Each of the textual relations will be labeled with both KB relations, while only one is correct (blue and solid), and the other is wrong (red and dashed). *Right*: distant supervision with global statistics. The two textual relations can be clearly distinguished by their co-occurrence distribution of KB relations. Statistics are based on the annotated ClueWeb data released in (Toutanova et al., 2015).

we can confidently believe that SUBJECT $\xleftarrow{\text{nsubj}}$ *died* $\xrightarrow{\text{nmod:in}}$ OBJECT means `place_of_death` in spite of the noise. Textual relation embedding learned on such global statistics is thus more robust to the noise introduced by the wrong labeling problem.

We augment existing relation extractions using the learned textual relation embedding. On a popular dataset introduced by Riedel et al. (2010), we show that a number of recent relation extraction models, which are based on local statistics, can be greatly improved using our textual relation embedding. Most remarkably, a new best performance is achieved when augmenting the previous best model with our relation embedding: The precision of the top 1,000 relational facts discovered by the model is improved from 83.9% to 89.3%, a 33.5% decrease in error rate. The results suggest that relation embedding with global statistics can capture complementary information to existing local statistics based models.

The rest of the paper is organized as follows. In Section 2 we discuss related work. For the modeling part, we first describe how to collect global co-occurrence statistics of relations in Section 3, then introduce a neural network based embedding model in Section 4, and finally discuss how to combine the learned textual relation embedding with existing relation extraction models in Section 5. We empirically evaluate the proposed method in Section 6, and conclude in Section 7.

2 Related Work

Relation extraction is an important task in information extraction. Early relation extraction methods are mainly feature-based (Kambhatla, 2004; Zhou et al., 2005), where features in various levels, including POS tags, syntactic and dependency parses, are integrated in a max entropy model. With the popularity of kernel methods, a large number of kernel-based relation extraction methods have been proposed (Zelenko et al., 2003; Culotta and Sorensen, 2004; Bunescu and Mooney, 2005; Zhang et al., 2006). The most related work to ours is by Bunescu and Mooney (Bunescu and Mooney, 2005), where the importance of shortest dependency path for relation extraction is first validated.

More recently, relation extraction research has been revolving around neural network models, which can alleviate the problem of exact feature matching of previous methods and have shown a remarkable success (e.g., (Socher et al., 2012; Zeng et al., 2014)). Among those, the most related are the ones embedding shortest dependency paths with neural networks (Xu et al., 2015a,b, 2016; Liu et al., 2016). For example, Xu et al. (2015b) use a RNN with LSTM units to embed shortest dependency paths without typed dependency relations, while a convolutional neural network is used in (Xu et al., 2015a). However, they are all based on the supervised setting with a limited scale. In contrast, we embed textual relations with distant supervision (Mintz et al., 2009), which provides much larger-scale training data at a low cost.

Various efforts have been made to combat the long-criticized wrong labeling problem of distant supervision. Riedel et al. (2010), Hoffmann et al. (2011), and Surdeanu et al. (2012) have attempted a multi-instance learning (Dietterich et al., 1997) framework to soften the assumption of distant supervision, but their models are still feature-based. Zeng et al. (2015) combine multi-instance learning with neural networks, with the assumption that at least one of the contextual sentences of an entity pair is expressing the target relation, but this will lose useful information in the neglected sentences. Instead, Lin et al. (2016) use all the contextual sentences, and introduce an attention mechanism to weight the contextual sentences. Li et al. (2017) also use an attention

mechanism to weight contextual sentences, and incorporate additional entity description information from knowledge bases. Luo et al. (2017) manage to alleviate the negative impact of noise by modeling and learning noise transition patterns from data. Liu et al. (2017) propose to infer the true label of a context sentence using a truth discovery approach (Li et al., 2016). Wu et al. (2017) incorporate adversarial training, i.e., injecting random perturbations in training, to improve the robustness of relation extraction. Using PCNN+ATT (Lin et al., 2016) as base model, they show that adversarial training can improve its performance by a good margin. However, the base model implementation used by them performed inferior to the one in the original paper and in ours, and therefore the results are not directly comparable. No prior study has exploited global statistics to combat the wrong labeling problem of distant supervision. Another unique aspect of this work is that we focus on compact textual relations, while previous studies along this line have focused on whole sentences.

In universal schema (Riedel et al., 2013) for KB completion and relation extraction as well as its extensions (Toutanova et al., 2015; Verga et al., 2016), a binary matrix is constructed from the entire corpus, with entity pairs as rows and textual/KB relations as columns. A matrix entry is 1 if the relational fact is observed in training, and 0 otherwise. Embeddings of entity pairs and relations, either directly or via neural networks, are then learned on the matrix entries, which are still individual relational facts, and the wrong labeling problem remains. Global co-occurrence frequencies (see Figure 1 (Right)) are not taken into account, which is the focus of this study. Another distinction is that our method directly models the association between textual and KB relations, while universal schema learns embedding for shared entity pairs and use that as a bridge between the two types of relations. It is an interesting venue for future research to comprehensively compare these two modeling approaches.

3 Global Statistics of Relations

When using a corpus to train statistical models, there are two levels of statistics to exploit: *local* and *global*. Take word embedding as an example. The skip-gram model (Mikolov et al., 2013) is based on local statistics: During training, we sweep through the corpus and slightly tune the embedding model in each local window (e.g., 10 consecutive words). In contrast, in global statistics based methods, exemplified by latent semantic analysis (Deerwester et al., 1990) and GloVe (Pennington et al., 2014), we process the entire corpus to collect global statistics like word-word co-occurrence counts, normalize the raw statistics, and train an embedding model directly on the normalized global statistics.

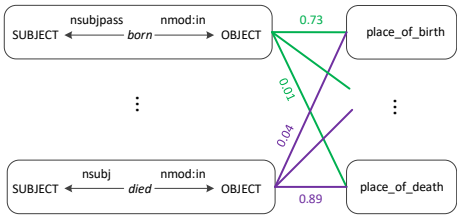


Figure 2: Relation graph. The left node set is textual relations, and the right node set is KB relations. The raw co-occurrence counts are normalized such that the KB relations corresponding to the same textual relation form a valid probability distribution. Edges are colored by textual relation and weighted by normalized co-occurrence statistics.

Most existing studies on relation extraction are based on local statistics of relations, i.e., models are trained on individual relation examples. In this section, we describe how we collect global co-occurrence statistics of textual and KB relations, and how to normalize the raw statistics. By the end of this section a bipartite *relation graph* like Figure 2 will be constructed, with one node set being textual relations $\mathcal{T}$, and the other being KB relations $\mathcal{R}$. The edges are weighted by the normalized co-occurrence statistics of relations.

3.1 Relation Graph Construction

Given a corpus and a KB, we first do entity linking on each sentence, and do dependency parsing if at least two entities are identified[2]. For each entity pair (e, e') in the sentence, we extract the fully lexicalized shortest dependency path as a textual relation t, forming a *relational fact* (e, t, e'). There are two outcomes from this step: a set of textual relations $\mathcal{T} = \{t_i\}$, and the *support* $S(t_i)$ for each t_i. The support of a textual relation is a *multiset* containing the entity pairs of the textual relation. The *multiplicity* of an entity pair, $m_{S(t_i)}(e, e')$, is the number of occurrences of the corresponding relational fact (e, t_i, e') in

[2]In the experiments entity linking is assumed given, and dependency parsing is done using Stanford Parser (Chen and Manning, 2014) with universal dependencies.

the corpus. For example, if the support of t_i is $S(t_i) = \{(e_1, e_1'), (e_1, e_1'), (e_2, e_2'), \dots\}$, entity pair (e_1, e_1') has a multiplicity of 2 because the relational fact (e_1, t_i, e_1') occur in two sentences. We also get a set of KB relations $\mathcal{R} = \{r_j\}$, and the support $S(r_j)$ of a KB relation r_j is the set of entity pairs having this relation in the KB, i.e., there is a relational fact (e, r_j, e') in the KB. The number of *co-occurrences* of a textural relation t_i and a KB relation r_j is

$$n_{ij} = \sum_{(e,e') \in S(r_j)} m_{S(t_i)}(e, e'), \qquad (1)$$

i.e., every occurrence of relational fact (e, t_i, e') is counted as a co-occurrence of t_i and r_j if $(e, e') \in S(r_j)$. A bipartite relation graph can then be constructed, with $\mathcal{T}$ and $\mathcal{R}$ as the node sets, and the edge between t_i and r_j has weight n_{ij} (no edge if $n_{ij} = 0$), which will be normalized later.

3.2 Normalization

The raw co-occurrence counts have a heavily skewed distribution that spans several orders of magnitude: A small portion of relation pairs co-occur highly frequently, while most relation pairs co-occur only a few times. For example, a textual relation, SUBJECT $\xleftarrow{\text{nsubjpass}}$ *born* $\xrightarrow{\text{nmod:in}}$ OBJECT, may co-occur with the KB relation `place_of_birth` thousands of times (e.g., "*Michelle Obama was born in Chicago*"), while a synonymous but slightly more compositional textual relation, SUBJECT $\xleftarrow{\text{nsubjpass}}$ *born* $\xrightarrow{\text{nmod:in}}$ *city* $\xrightarrow{\text{nmod:of}}$ OBJECT, may only co-occur with the same KB relation a few times in the entire corpus (e.g., "*Michelle Obama was born in the city of Chicago*"). Learning directly on the raw co-occurrence counts, an embedding model may put a disproportionate amount of weight on the most frequent relations, and may not learn well on the majority of rarer relations. Proper normalization is therefore necessary, which will encourage the embedding model to learn good embedding not only for the most frequent relations, but also for the rarer relations.

A number of normalization strategies have been proposed in the context of word embedding, including correlation- and entropy-based normalization (Rohde et al., 2005), positive pointwise mutual information (PPMI) (Bullinaria and Levy, 2007), and some square root type transformation (Lebret and Collobert, 2014). A shared goal is to reduce the impact of the most frequent words, e.g., "the" and "is," which tend to be less informative for the purpose of embedding.

We have experimented with a number of normalization strategies and found that the following strategy works best for textual relation embedding: For each textual relation, we normalize its co-occurrence counts to form a probability distribution over KB relations. The new edge weights of the relation graph thus become $w_{ij} = \tilde{p}(r_j|t_i) = n_{ij} / \sum_{j'} n_{ij'}$. Every textual relation is now associated with a set of edges whose weights sum up to 1. We also experimented with PPMI and smoothed PPMI with $\alpha = 0.75$ (Levy et al., 2015) that are commonly used in word embedding. However, the learned textual relation embedding turned out to be not very helpful for relation extraction. One possible reason is that PPMI (even the smoothed version) gives inappropriately large weights to rare relations (Levy et al., 2015). There are many textual relations that correspond to none of the target KB relations but are falsely labeled with some KB relations a few times by distant supervision. PPMI gives large weights to such falsely labeled cases because it thinks these events have a chance significantly higher than random.

4 Textual Relation Embedding

Next we discuss how to learn embedding of textual relations based on the constructed relation graph. We call our approach **Glo**bal **R**elation **E**mbedding (GloRE) in light of global statistics of relations.

4.1 Embedding via RNN

Given the relation graph, a straightforward way of relation embedding is matrix factorization, similar to latent semantic analysis (Deerwester et al., 1990) for word embedding. However, textual relations are different from words in that they are sequences composed of words and typed dependency relations. Therefore, we use recurrent neural networks (RNNs) for embedding, which respect the compositionality of textual relations and can learn the shared sub-structures of different textual relations (Toutanova et al., 2015). For the examples in Figure 1, an RNN can learn, from both textual relations, that the shared dependency relation "nmod:in" is indicative of location modifiers. It is worth noting that other models like convolutional neural networks can also be used, but it is not the focus of this paper to compare all the alternative embedding models; rather, we aim to show

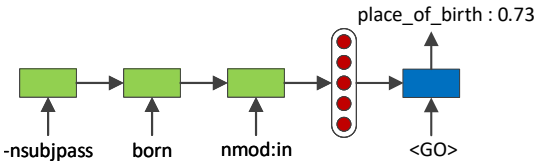


Figure 3: Embedding model. *Left*: A RNN with GRU for embedding. *Middle*: embedding of textual relation. *Right*: a separate GRU cell to map a textual relation embedding to a probability distribution over KB relations.

the effectiveness of global statistics with a reasonable embedding model.

For a textual relation, we first decompose it into a sequence of tokens $\{x_1, ..., x_m\}$, which includes lexical words and directional dependency relations. For example, the textual relation SUBJECT $\xleftarrow{\text{nsubjpass}}$ *born* $\xrightarrow{\text{nmod:in}}$ OBJECT is decomposed to a sequence of three tokens $\{-\text{nsubjpass}, \text{born}, \text{nmod:in}\}$, where "$-$" represents a left arrow. Note that we include directional dependency relations, because both the relation type and the direction are critical in determining the meaning of a textual relation. For example, the dependency relation "nmod:in" often indicates a location modifier and is thus strongly associated with location-related KB relations like `place_of_birth`. The direction also plays an important role. Without knowing the direction of the dependency relations, it is impossible to distinguish `child_of` and `parent_of`.

An RNN with gated recurrent units (GRUs) (Cho et al., 2014) is then applied to consecutively process the sequence as shown in Figure 3. We have also explored more advanced constructs like attention, but the results are similar, so we opt for a vanilla RNN in consideration of model simplicity.

Let ϕ denote the function that maps a token x_l to a fixed-dimensional vector, the hidden state vectors of the RNN are calculated recursively:

$$\boldsymbol{h}_l = \text{GRU}\big(\phi(x_l), \boldsymbol{h}_{l-1}\big). \tag{2}$$

GRU follows the definition in Cho et al. (2014).

4.2 Training Objective

We use global statistics in the relation graph to train the embedding model. Specifically, we model the semantics of a textual relation as its co-occurrence distribution of KB relations, and learn textual relation embedding to reconstruct the corresponding co-occurrence distributions.

We use a separate GRU cell followed by softmax to map a textual relation embedding to a distribution over KB relations; the full model thus resembles the sequence-to-sequence architecture (Sutskever et al., 2014). Given a textual relation t_i and its embedding $\boldsymbol{h}_m$, the predicted conditional probability of a KB relation r_j is thus:

$$p(r_j|t_i) = \text{softmax}(\text{GRU}(\phi(<\text{GO}>), \boldsymbol{h}_m))_j, \tag{3}$$

where $()_j$ denotes the j-th element of a vector, and <GO>is a special token indicating the start of decoding. The training objective is to minimize

$$\Theta = \frac{1}{|\mathcal{E}|} \sum_{i,j:\tilde{p}(r_j|t_i)>0} \left(\log p(r_j|t_i) - \log \tilde{p}(r_j|t_i)\right)^2, \tag{4}$$

where $\mathcal{E}$ is the edge set of the relation graph. It is modeled as a regression problem, similar to GloVe (Pennington et al., 2014).

Baseline. We also define a baseline approach where the unnormalized co-occurrence counts are directly used. The objective is to maximize:

$$\Theta' = \frac{1}{\sum_{i,j} n_{ij}} \sum_{i,j:n_{ij}>0} n_{ij} \log p(r_j|t_i). \tag{5}$$

It also corresponds to local statistics based embedding, i.e., when the embedding model is trained on individual occurrences of relational facts with distant supervision. Therefore, we call it **Lo**cal **R**elation **E**mbedding (LoRE).

5 Augmenting Relation Extraction

Learned from global co-occurrence statistics of relations, our approach provides semantic matching information of textual and KB relations, which is often complementary to the information captured by existing relation extraction models. In this section we discuss how to combine them together to achieve better relation extraction performance.

We follow the setting of distantly supervised relation extraction. Given a text corpus and a KB with relation set $\mathcal{R}$, the goal is to find new relational facts from the text corpus that are not already contained in the KB. More formally, for each entity pair (e, e') and a set of *contextual sentences* C containing this entity pair, a relation extraction model assigns a score $E(z|C)$ to each candidate relational fact $z = (e, r, e'), r \in \mathcal{R}$. On the

other hand, our textual relation embedding model works on the sentence level. It assign a score $G(z|s)$ to each contextual sentence s in C as for how well the textual relation t between the entity pair in the sentence matches the KB relation r, i.e., $G(z|s) = p(r|t)$. It poses a challenge to aggregate the sentence-level scores to get a set-level score $G(z|C)$, which can be used to combine with the original score $E(z|C)$ to get a better evaluation of the candidate relational fact.

One straightforward aggregation is max pooling, i.e., only using the largest score $\max_{s \in C} G(z|s)$, similar to the at-least-one strategy used by Zeng et al. (2015). But it will lose the useful signals from those neglected sentences (Lin et al., 2016). Because of the wrong labeling problem, mean pooling is problematic as well. The wrongly labeled contextual sentences tend to make the aggregate scores more evenly distributed and therefore become less informative. The number of contextual sentences positively supporting a relational fact is also an important signal, but is lost in mean pooling.

Instead, we use summation with a trainable cap:

$$G(z|C) = \min\left(cap, \sum_{s \in C} G(z|s)\right), \qquad (6)$$

In other words, we additively aggregate the signals from all the contextual sentences, but only to a bounded degree.

We simply use a weighted sum to combine $E(z|C)$ and $G(z|C)$, where the trainable weights will also handle the possibly different scale of scores generated by different models:

$$\tilde{E}(z|C) = w_1 E(z|C) + w_2 G(z|C). \qquad (7)$$

The original score $E(z|C)$ is then replaced by the new score $\tilde{E}(z|C)$. To find the optimal values for w_1, w_2 and cap, we define a hinge loss:

$$\Theta_{Merge} = \frac{1}{K} \sum_{k=1}^{K} \max\left\{0, 1 + \tilde{E}(z_k^-) - \tilde{E}(z_k^+)\right\}, \qquad (8)$$

where $\{z_k^+\}_{k=1}^K$ are the true relational facts from the KB, and $\{z_k^-\}_{k=1}^K$ are false relational facts generated by replacing the KB relation in true relational facts with incorrect KB relations.

Data	# of sentences	# of entity pairs	# of relational facts from KB
Train	570,088	291,699	19,429
Test	172,448	96,678	1,950

Table 1: Statistics of the NYT dataset.

6 Experiments

In this experimental study, we show that GloRE can greatly improve the performance of several recent relation extraction models, including the previous best model on a standard dataset.

6.1 Experimental Setup

Dataset. Following the literature (Hoffmann et al., 2011; Surdeanu et al., 2012; Zeng et al., 2015; Lin et al., 2016), we use the relation extraction dataset introduced in (Riedel et al., 2010), which was generated by aligning New York Times (NYT) articles with Freebase (Bollacker et al., 2008). Articles from year 2005-2006 are used as training, and articles from 2007 are used as testing. Some statistics are listed in Table 1. There are 53 target KB relations, including a special relation NA indicating that there is no target relation between entities.

We follow the approach described in Section 3 to construct the relation graph from the NYT training data. The constructed relation graph contains 321,447 edges with non-zero weight. We further obtain a training set and a validation set from the edges of the relation graph. We have observed that using a validation set totally disjoint from the training set leads to unstable validation loss, so we randomly sample 300K edges as the training set, and another 60K as the validation set. The two sets can have some overlap. For the merging model (Eq. 8), 10% of the edges are reserved as the validation set.

Relation extraction models. We evaluate with four recent relation extraction models whose source code is publicly available[3]. We use the optimized parameters provided by the authors.

- **CNN+ONE** and **PCNN+ONE** (Zeng et al., 2015): A convolutional neural network (CNN) is used to embed contextual sentences for relation classification. Multi-instance learning with at-least-one (ONE) assumption is used to combat the wrong labeling problem. In PCNN, piecewise max pooling is

[3] https://github.com/thunlp/NRE

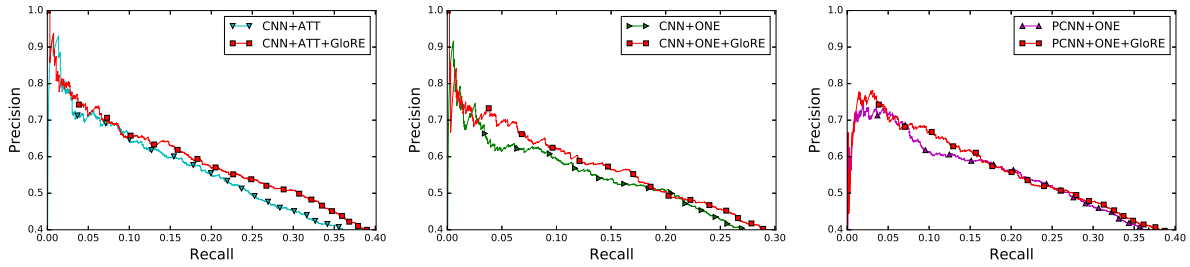


Figure 4: Held-out evaluation: other base relation extraction models and the improved versions when augmented with GloRE.

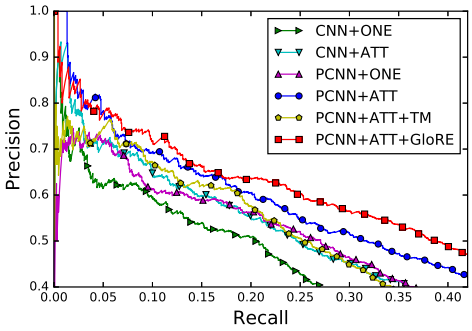


Figure 5: Held-out evaluation: the previous best-performing model can be further improved when augmented with GloRE. PCNN+ATT+TM is a recent model (Luo et al., 2017) whose performance is slightly inferior to PCNN+ATT. Because the source code is not available, we did not experiment to augment this model with GloRE. Another recent method (Wu et al., 2017) incorporates adversarial training to improve PCNN+ATT, but the results are not directly comparable (see Section 2 for discussion). Finally, Ji et al. (2017) propose a model similar to PCNN+ATT, but the performance is inferior to PCNN+ATT and is not shown here for clarity.

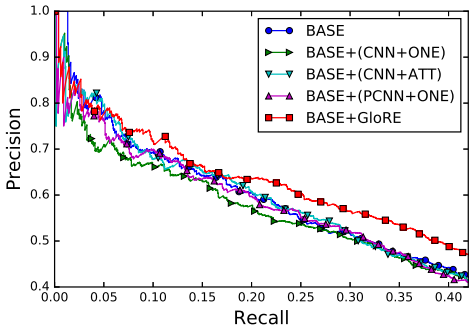


Figure 6: Held-out evaluation: GloRE brings the largest improvement to BASE (PCNN+ATT), which further shows that GloRE captures useful information for relation extraction that is complementary to existing models.

used to handle the three pieces of a contextual sentence (split by the two entities) separately.

- **CNN+ATT** and **PCNN+ATT** (Lin et al., 2016): Different from the at-least-one assumption which loses information in the neglected sentences, these models learn soft attention weights (ATT) over contextual sentences and thus can use the information of all the contextual sentences. *PCNN+ATT is the best-performing model on the NYT dataset.*

Evaluation settings and metrics. Similar to previous work (Riedel et al., 2010; Zeng et al., 2015), we use two settings for evaluation: (1) Held-out evaluation, where a subset of relational facts in KB is held out from training (Table 1), and is later used to compare against newly discovered relational facts. This setting avoids human labor but can introduce some false negatives because of the incompleteness of the KB. (2) Manual evaluation, where the discovered relational facts are manually judged by human experts. For held-out evaluation, we report the precision-recall curve. For manual evaluation, we report $Precision@N$, i.e., the precision of the top N discovered relational facts.

Implementation. Hyper-parameters of our model are selected based on the validation set. For the embedding model, the mini-batch size is set to 128, and the state size of the GRU cells is 300. For the merging model, the mini-batch size is set to 1024. We use Adam with parameters recommended by the authors for optimization. Word embeddings are initialized with the 300-dimensional word2vec vectors pre-trained on the Google News corpus[4]. Early stopping based on the validation set is employed. Our model is implemented using Tensorflow (Abadi et al., 2016), and the source code is available at https://github.com/pputiu/GloRE.

[4] https://code.google.com/archive/p/word2vec/

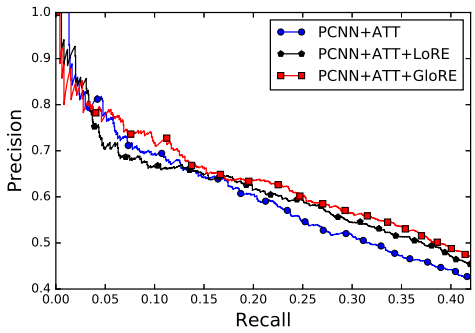


Figure 7: Held-out evaluation: LoRE vs. GloRE.

6.2 Held-out Evaluation

Existing Models + GloRE. We first show that our approach, GloRE, can improve the performance of the previous best-performing model, PCNN+ATT, leading to a new state of the art on the NYT dataset. As shown in Figure 5, when PCNN+ATT is augmented with GloRE, a consistent improvement along the precision-recall curve is observed. It is worth noting that although PCNN+ATT+GloRE seems to be inferior to PCNN+ATT when recall < 0.05, as we will show via manual evaluation, it is actually due to false negatives.

We also show in Figure 4 that the improvement brought by GloRE is general and not specific to PCNN+ATT; the other models also get a consistent improvement when augmented with GloRE.

To investigate whether the improvement brought by GloRE is simply from ensemble, we also augment PCNN+ATT with the other three base models in the same way as described in Section 5. The results in Figure 6 show that pairwise ensemble of existing relation extraction models does not yield much improvement, and GloRE brings much larger improvement than the other models.

In summary, the held-out evaluation results suggest that GloRE captures useful information for relation extraction that is not captured by these local statistics based models.

LoRE v.s. GloRE. We compare GloRE with the baseline approach LoRE (Section 4) to show the advantage of normalization on global statistics. We use PCNN+ATT as the base relation extraction model. As shown in Figure 7, GloRE consistently outperforms LoRE. It is worth noting that LoRE can still improve the base relation extraction model when recall > 0.15, further confirming the usefulness of directly embedding textual relations in addition to sentences.

Precision@N	100	300	500	700	900	1000
PCNN+ATT	**97.0**	93.7	92.8	89.1	85.2	83.9
PCNN+ATT+LoRE	**97.0**	95.0	94.2	91.6	89.6	87.0
PCNN+ATT+GloRE	**97.0**	**97.3**	**94.6**	**93.3**	**90.1**	**89.3**

Table 2: Manual evaluation: false negatives from held-out evaluation are manually corrected by human experts.

6.3 Manual Evaluation

Due to the incompleteness of the knowledge base, held-out evaluation introduces some false negatives. The precision from held-out evaluation is therefore a lower bound of the true precision. To get a more accurate evaluation of model performance, we have human experts to manually check the false relational facts judged by held-out evaluation in the top 1,000 predictions of three models, PCNN+ATT, PCNN+ATT+LoRE and PCNN+ATT+GloRE, and report the corrected results in Table 2. Each prediction is examined by two human experts who reach agreement with discussion. To ensure fair comparison, the experts are not aware of the provenance of the predictions. Under manual evaluation, PCNN+ATT+GloRE achieves the best performance in the full range of N. In particular, for the top 1,000 predictions, GloRE improves the precision of the previous best model PCNN+ATT from 83.9% to 89.3%. The manual evaluation results reinforce the previous observations from held-out evaluation.

6.4 Case Study

Table 3 shows two examples. For better illustration, we choose entity pairs that have only one contextual sentence.

For the first example, PCNN+ATT predicts that most likely there is no KB relation between the entity pair, while both LoRE and GloRE identify the correct relation with high confidence. The textual relation clearly indicates that the head entity is (appos) a criminologist at (nmod:at) the tail entity.

For the second example, there is no KB relation between the entity pair, and PCNN+ATT is indeed able to rank NA at the top. However, it is still quite confused by `nationality`, probably because it has learned that sentences about a person and a country with many words about profession ("poet," "playwright," and "novelist")

Contextual Sentence	Textual Relation	PCNN+ATT Predictions	LoRE Predictions	GloRE Predictions
[**Alfred Blumstein**]$_{head}$, a criminologist at [**Carnegie Mellon University**]$_{tail}$, called ...	$\xleftarrow{\text{appos}}$ criminologist $\xrightarrow{\text{nmod:at}}$	NA (0.63) **employee_of** (0.36) founder_of (0.00)	**employee_of** (1.00) NA (0.00) founder_of (0.00)	**employee_of** (0.96) NA (0.02) founder_of (0.02)
[**Langston Hughes**]$_{head}$, the American poet, playwright and novelist, came to [**Spain**]$_{tail}$ to ...	$\xleftarrow{\text{-nsubj}}$ came $\xrightarrow{\text{to}}$	**NA** (0.58) nationality (0.38) place_lived (0.01)	place_of_death (0.35) **NA** (0.33) nationality (0.21)	**NA** (0.73) contain_location (0.07) employee_of (0.06)

Table 3: Case studies. We select entity pairs that have only one contextual sentence, and the head and tail entities are marked. The top 3 predictions from each model with the associated probabilities are listed, with the correct relation bold-faced.

likely express the person's nationality. As a result, its prediction on NA is not very confident. On the other hand, GloRE learns that if a person "came to" a place, likely it is not his/her birthplace. In the training data, due to the wrong labeling problem of distant supervision, the textual relation is wrongly labeled with `place_of_death` and `nationality` a couple of times, and both PCNN+ATT and LoRE suffer from the training noise. Taking advantage of global statistics, GloRE is more robust to such noise introduced by the wrong labeling problem.

7 Conclusion

Our results show that textual relation embedding trained on global co-occurrence statistics captures useful relational information that is often complementary to existing methods. As a result, it can greatly improve existing relation extraction models. Large-scale training data of embedding can be easily solicited from distant supervision, and the global statistics of relations provide a natural way to combat the wrong labeling problem of distant supervision.

The idea of relation embedding based on global statistics can be further expanded along several directions. In this work we have focused on embedding textual relations, but it is in principle beneficial to jointly embed knowledge base relations and optionally entities. Recently a joint embedding approach has been attempted in the context of knowledge base completion (Toutanova et al., 2015), but it is still based on local statistics, i.e., individual relational facts. Joint embedding with global statistics remains an open problem. Compared with the size of the training corpora for word embedding (up to hundred of billions of tokens), the NYT dataset is quite small in scale. Another interesting venue for future research is to construct much larger-scale distant supervision datasets to train general-purpose textual relation embedding that can help a wide range of downstream relational tasks such as question answering and textual entailment.

Acknowledgements

The authors would like to thank the anonymous reviewers for their thoughtful comments. This research was sponsored in part by the Army Research Laboratory under cooperative agreements W911NF09-2-0053 and NSF IIS 1528175. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notice herein.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv:1603.04467* .

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International conference on Management of data*. ACM, pages 1247–1250.

John A Bullinaria and Joseph P Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior research methods* 39(3):510–526.

Razvan C Bunescu and Raymond J Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 724–731.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 740–750.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv:1406.1078* .

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science* 41(6):391.

Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez. 1997. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence* 89(1):31–71.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 541–550.

Guoliang Ji, Kang Liu, Shizhu He, Jun Zhao, et al. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the ACL on Interactive poster and demonstration sessions*. Association for Computational Linguistics.

Rémi Lebret and Ronan Collobert. 2014. Word embeddings through hellinger PCA. *European Chapter of the Association for Computational Linguistics* .

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics* 3:211–225.

Yaliang Li, Jing Gao, Chuishi Meng, Qi Li, Lu Su, Bo Zhao, Wei Fan, and Jiawei Han. 2016. A survey on truth discovery. *Acm Sigkdd Explorations Newsletter* 17(2):1–16.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 2124–2133.

Liyuan Liu, Xiang Ren, Qi Zhu, Shi Zhi, Huan Gui, Heng Ji, and Jiawei Han. 2017. Heterogeneous supervision for relation extraction: A representation learning approach. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*.

Yang Liu, Sujian Li, Furu Wei, and Heng Ji. 2016. Relation classification via modeling augmented dependency paths. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)* 24(9):1585–1594.

Bingfeng Luo, Yansong Feng, Zheng Wang, Zhanxing Zhu, Songfang Huang, Rui Yan, and Dongyan Zhao. 2017. Learning with noise: Enhance distantly supervised relation extraction with dynamic transition matrix. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. pages 430–439.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the Annual Conference on Neural Information Processing Systems*. pages 3111–3119.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 1003–1011.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1532–1543.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, pages 148–163.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Douglas LT Rohde, Laura M Gonnerman, and David C Plaut. 2005. An improved model of semantic similarity based on lexical co-occurrence. *Communications of the ACM* 8:627–633.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, pages 1201–1211.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 455–465.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the Annual Conference on Neural Information Processing Systems*. pages 3104–3112.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1499–1509.

Patrick Verga, David Belanger, Emma Strubell, Benjamin Roth, and Andrew McCallum. 2016. Multilingual relation extraction using compositional universal schema. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Yi Wu, David Bamman, and Stuart Russell. 2017. Adversarial training for relation extraction. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015a. Semantic relation classification via convolutional neural networks with simple negative sampling. *arXiv:1506.07650* .

Yan Xu, Ran Jia, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu, and Zhi Jin. 2016. Improved relation classification by deep recurrent neural networks with data augmentation. *arXiv:1601.03651* .

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015b. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1785–1794.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of machine learning research* 3(Feb):1083–1106.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1753–1762.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation classification via convolutional deep neural network. In *Proceedings of the International Conference on Computational Linguistics*. pages 2335–2344.

Min Zhang, Jie Zhang, and Jian Su. 2006. Exploring syntactic features for relation extraction using a convolution tree kernel. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*. Association for Computational Linguistics, pages 288–295.

Zhou Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 427–434.

Implicit Argument Prediction with Event Knowledge

Pengxiang Cheng
Department of Computer Science
The University of Texas at Austin
pxcheng@cs.utexas.edu

Katrin Erk
Department of Linguistics
The University of Texas at Austin
katrin.erk@mail.utexas.edu

Abstract

Implicit arguments are not syntactically connected to their predicates, and are therefore hard to extract. Previous work has used models with large numbers of features, evaluated on very small datasets. We propose to train models for implicit argument prediction on a simple cloze task, for which data can be generated automatically at scale. This allows us to use a neural model, which draws on narrative coherence and entity salience for predictions. We show that our model has superior performance on both synthetic and natural data. [1]

1 Introduction

When parts of an event description in a text are missing, this event cannot be easily extracted, and it cannot easily be found as the answer to a question. This is the case with *implicit arguments*, as in this example from the reading comprehension dataset of Hermann et al. (2015):

> **Text:** More than 2,600 people have been infected by Ebola in Liberia, Guinea, Sierra Leone and Nigeria since the *outbreak* began in December, according to the World Health Organization. Nearly 1,500 have *died*.
>
> **Question:** The X outbreak has killed nearly 1,500.

In this example, it is Ebola that broke out, and Ebola was also the cause of nearly 1,500 people dying, but the text does not state this explicitly. *Ebola* is an implicit argument of both *outbreak* and *die*, which is crucial to answering the question.

We are particularly interested in implicit arguments that, like *Ebola* in this case, do appear in the text, but not as syntactic arguments of their predicates. Event knowledge is key to determining implicit arguments. In our example, diseases are maybe the single most typical things to *break out*, and diseases also typically kill people.

The task of identifying implicit arguments was first addressed by Gerber and Chai (2010) and Ruppenhofer et al. (2010). However, the datasets for the task were very small, and to our knowledge there has been very little further development on the task since then.

In this paper, we address the data issue by training models for implicit argument prediction on a simple cloze task, similar to the narrative cloze task (Chambers and Jurafsky, 2008), for which data can be generated automatically at scale. This allows us to train a neural network to perform the task, building on two insights. First, event knowledge is crucial for implicit argument detection. Therefore we build on models for narrative event prediction (Granroth-Wilding and Clark, 2016; Pichotta and Mooney, 2016a), using them to judge how coherent the narrative would be when we fill in a particular entity as the missing (implicit) argument. Second, the omitted arguments tend to be salient, as *Ebola* is in the text from which the above example is taken. So in addition to narrative coherence, our model takes into account entity salience (Dunietz and Gillick, 2014).

In an evaluation on a large automatically generated dataset, our model clearly outperforms even strong baselines, and we find salience features to be important to the success of the model. We also evaluate against a variant of the Gerber and Chai (2012) model that does not rely on gold features, finding that our simple neural model outperforms their much more complex model.

Our paper thus makes two major contributions. 1) We propose an argument cloze task to generate synthetic training data at scale for implicit argument prediction. 2) We show that neural event

[1] Our code is available at https://github.com/pxch/event_imp_arg.

Proceedings of NAACL-HLT 2018, pages 831–840
New Orleans, Louisiana, June 1 - 6, 2018.

models for narrative schema prediction can be used on implicit argument prediction, and that a straightforward combination of event knowledge and entity salience can do well on the task.

2 Related Work

While dependency parsing and semantic role labeling only deal with arguments that are available in the syntactic context of the predicate, implicit argument labeling seeks to find argument that are not syntactically connected to their predicates, like *Ebola* in our introductory example.

The most relevant work on implicit argument prediction came from Gerber and Chai (2010), who built an implicit arguments dataset by selecting 10 nominal predicates from NomBank (Meyers et al., 2004) and manually annotating implicit arguments for all occurrences of these predicates. In an analysis of their data they found implicit arguments to be very frequent, as their annotation added 65% more arguments to NomBank. Gerber and Chai (2012) also trained a linear classifier for the task relying on many hand-crafted features, including gold features from FrameNet (Baker et al., 1998), PropBank (Palmer et al., 2005) and NomBank. This classifier has, to the best of our knowledge, not been outperformed by follow-up work (Laparra and Rigau, 2013; Schenk and Chiarcos, 2016; Do et al., 2017). We evaluate on the Gerber and Chai dataset below. Ruppenhofer et al. (2010) also introduced an implicit argument dataset, but we do not evaluate on it as it is even smaller and much more complex than Gerber and Chai (2010). More recently, Modi et al. (2017) introduced the referent cloze task, in which they predicted a manually removed discourse referent from a human annotated narrative text. This task is closely related to our argument cloze task.

Since we intend to exploit event knowledge in predicting implicit arguments, we here refer to recent work on statistical script learning, started by Chambers and Jurafsky (2008, 2009). They introduced the idea of using statistical information on coreference chains to induce prototypical sequences of narrative events and participants, which is related to the classical notion of a script (Schank and Abelson, 1977). They also proposed the narrative cloze evaluation, in which one event is removed at random from a sequence of narrative events, then the missing event is predicted given all context events. We use a similar trick to define a cloze task for implicit argument prediction, discussed in Section 3.

Many follow-up papers on script learning have used neural networks. Rudinger et al. (2015) showed that sequences of events can be efficiently modeled by a log-bilinear language model. Pichotta and Mooney (2016a,b) used an LSTM to model a sequence of events. Granroth-Wilding and Clark (2016) built a network that produces an event representation by composing its components. To do the cloze task, they select the most probable event based on pairwise event coherence scores. For our task we want to do something similar: We want to predict how coherent a narrative would be with a particular entity candidate filling the implicit argument position. So we take the model of Granroth-Wilding and Clark (2016) as our starting point.

The Hermann et al. (2015) reading comprehension task, like our cloze task, requires systems to guess a removed entity. However in their case the entity is removed in a summary, not in the main text. In their case, the task typically amounts to finding a main text passage that paraphrases the sentence with the removed entity; this is not the case in our cloze task.

3 The Argument Cloze Task

We present the **argument cloze** task, which allows us to automatically generate large scale data for training (Section 6.1) and evaluation (Section 5.1).

In this task, we randomly remove an entity from an argument position of one event in the text. The entity in question needs to appear in at least one other place in the text. The task is then for the model to pick, from all entities appearing in the text, the one that has been removed. We first define what we mean by an event, then what we mean by an entity. Like Pichotta and Mooney (2016a); Granroth-Wilding and Clark (2016), we define an *event* e as consisting of a verbal predicate v, a subject s, a direct object o, and a prepositional object p (along with the preposition). Here we only allow one prepositional argument in the structure, to avoid variable length input in the event composition model.[2] By an *entity*, we mean a coreference chain with a length of at least two – that is, the entity needs to appear at least twice in the text.

For example, from a piece of raw text (Figure

[2]In case of multiple prepositional objects, we select the one that is closest to the predicate.

Manville Corp. said it will build a $ 24 million power plant to provide electricity to its Igaras pulp and paper mill in Brazil .

The company said the plant will ensure that it has adequate energy for the mill and will reduce the mill's energy costs .

(a) A piece of raw text from OntoNotes corpus.

x_0 = The company x_1 = mill x_2 = power plant

e_0: (*build-pred*, x_0-*subj*, x_2-*dobj*, —)
e_1: (*provide-pred*, —, *electricity-dobj*, x_1-*prep_to*)
e_2: (*ensure-pred*, x_2-*subj*, —, —)
e_3: (*has-pred*, x_0-*subj*, *energy-dobj*, x_1-*prep_for*)
e_4: (*reduce-pred*, x_2-*subj*, *cost-dobj*, —)

(b) Extracted events (e_0~e_4) and entities (x_0~x_2), using gold annotations from OntoNotes.

e_0, e_2, e_3, e_4: same as above
e_1: (*provide-pred*, —, *electricity-dobj*, ***??-prep_to***)

x_0 = The company x_1 = mill x_2 = power plant

(c) Example of an argument cloze task for *prep_to* of e_1.

Figure 1: Example of automatically extracted events and entities and an argument cloze task.

1a), we automatically extract a sequence of events from a dependency parse, and a list of entities from coreference chains. In Figure 1b, e_0~e_4 are events, x_0~x_2 are entities. The arguments *electricity-dobj* and *energy-dobj* are not in coreference chains and are thus not candidates for removal. An example of the argument cloze task is shown in Figure 1c. Here the *prep_to* argument of e_1 has been removed.

Coreference resolution is very noisy. Therefore we use gold coreference annotation for creating evaluation data, but automatically generated coreference chains for creating training data.

4 Methods

4.1 Modeling Narrative Coherence

We model implicit argument prediction as selecting the entity that, when filled in as the implicit argument, makes the overall most coherent narrative. Suppose we are trying to predict the direct object argument of some target event e_t. Then we complete e_t by putting an entity candidate into the direct object argument position, and check the coherence of the resulting event with the rest of the narrative. Say we have a sequence of events $e_1, e_2, \ldots, e_n$ in a narrative, and a list of entity candidates $x_1, x_2, \ldots, x_m$. Then for any candidate x_j, we first complete the target event to be

$$e_t(j) = (v_t, s_t, x_j, p_t), \quad j = 1, \ldots, m \qquad (1)$$

where v_t, s_t, and p_t are the predicate, subject, and prepositional object of e_t respectively, and x_j is filled as the direct object. (Event completion for omitted subjects and prepositional objects is analogous.)

Then we compute the narrative coherence score S_j of the candidate x_j by[3]

$$S_j = \max_{c=1,\, c\neq t}^{n} coh\left(\vec{e_t(j)}, \vec{e_c}\right), \quad j = 1, \ldots, m \qquad (2)$$

where $\vec{e_t(j)}$ and $\vec{e_c}$ are representations for the completed target event $e_t(j)$ and one context event e_c, and coh is a function computing a coherence score between two events, both depending on the model being used. The candidate x_j with the highest score S_j is then selected as our prediction.

4.2 The Event Composition Model

To model coherence (coh) between a context event and a target event, we build an event composition model consisting of three parts, as shown in Figure 2: event components are representated through **event-based word embeddings**, which encode event knowledge in word representations; the **argument composition network** combines the components to produce event representations; and the **pair composition network** compute a coherence score for two event representations.

This basic architecture is as in the model of Granroth-Wilding and Clark (2016). However our model is designed for a different task, argument cloze rather than narrative cloze, and for our task entity-specific information is more important. We therefore create the training data in a different way, as described in Section 4.2.1. We now discuss the three parts of the model in more detail.

Event-Based Word Embeddings The model takes word embeddings of both predicates and

[3] We have also tried using the sum instead of the maximum, but it did not perform as well across different models and datasets.

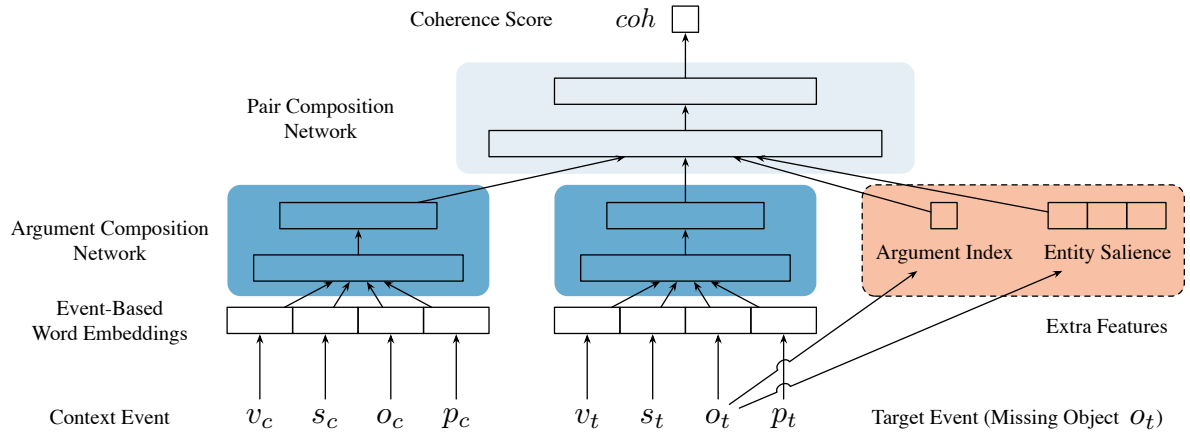


Figure 2: Diagram for event composition model. ***Input***: a context event and a target event. ***Event-Based Word Embeddings***: embeddings for components of both events that encodes event knowledge. ***Argument Composition Network***: produces an event representation from its components. ***Pair Composition Network***: computes a coherence score *coh* from two event representations. ***Extra Features***: argument index and entity salience features as additional input to the pair composition network.

arguments as input to compute event representations. To better encode event knowledge in word level, we train an SGNS (skip-gram with negative sampling) word2vec model (Mikolov et al., 2013) with event-specific information. For each extracted event sequence, we create a sentence with the predicates and arguments of all events in the sequence. An example of such a training sentence is given in Figure 3.

build-pred company-subj plant-dobj provide-pred electricity-dobj mill-prep_to ensure-pred plant-*subj has-pred company-subj energy-dobj mill-prep_for reduce-pred plant-subj cost-dobj*

Figure 3: Event-based word2vec training sentence, constructed from events and entities in Figure 1b.

Argument Composition Network The argument composition network (dark blue area in Figure 2) is a two-layer feedforward neural network that composes an event representation from the embeddings of its components. Non-existent argument positions are filled with zeros.

Pair Composition Network The pair composition network (light blue area in Figure 2) computes a coherence score *coh* between 0 and 1, given the vector representations of a context event and a target event. The coherence score should be high when the target event contains the correct argument, and low otherwise. So we construct the training objective function to distinguish the correct argument from wrong ones, as described in Equation 3.

4.2.1 Training for Argument Prediction

To train the model to pick the correct candidate, we automatically construct training samples as event triples consisting of a context event e_c, a positive event e_p, and a negative event e_n. The context event and positive event are randomly sampled from an observed sequence of events, while the negative event is generated by replacing one argument of positive event by a random entity in the narrative, as shown in Figure 4.

x_0 = The company x_1 = mill x_2 = power plant

Context: (*build-pred*, x_0-*subj*, x_2-*dobj*, –)
Positive: (*reduce-pred*, **x_2-*subj***, *cost-dobj*, –)
Negative: (*reduce-pred*, **x_1-*subj***, *cost-dobj*, –)

Figure 4: Example of an event triple constructed from events and entities in Figure 1b.

We want the coherence score between e_c and e_p to be close to 1, while the score for e_c and e_n should be close to 0. Therefore, we train the model to minimize cross-entropy as follows:

$$\frac{1}{m}\sum_{i=1}^{m} -\log(coh(e_{ci}, e_{pi})) - \log(1 - coh(e_{ci}, e_{ni})) \quad (3)$$

where e_{ci}, e_{pi}, and e_{ni} are the context, positive, and negative events of the ith training sample respectively.

4.3 Entity Salience

Implicit arguments tend to be salient entities in the document. So we extend our model by entity salience features, building on recent work by Dunietz and Gillick (2014), who introduced a simple model with several surface level features for entity salience detection. Among the features they used, we discard those that require external resources, and only use the remaining three features, as illustrated in Table 1. Dunietz and Gillick found *mentions* to be the most powerful indicator for entity salience among all features. We expect similar results in our experiments, however we include all three features in our event composition model for now, and conduct an ablation test afterwards.

Feature	Description
1st_loc	Index of the sentence where the first mention of the entity appears
head_count	Number of times the head word of the entity appears
mentions	A vector containing the numbers of named, nominal, pronominal, and total mentions of the entity

Table 1: Entity salience features from Dunietz and Gillick (2014).

The entity salience features are directly passed into the pair composition network as additional input. We also add an extra feature for argument position index (encoding whether the missing argument is a subject, direct object, or prepositional object), as shown in the red area in Figure 2.

5 Evaluation Datasets

5.1 Argument Cloze Evaluation

Previous implicit argument datasets were very small. To overcome that limitation, we automatically create a large and comprehensive evaluation dataset, following the **argument cloze** task setting in Section 3.

Since the events and entities are extracted from dependency labels and coreference chains, we do not want to introduce systematic error into the evaluation from imperfect parsing and coreference algorithms. Therefore, we create the evaluation set from OntoNotes (Hovy et al., 2006), which contains human-labeled dependency and coreference annotation for a large corpus. So the extracted events and entities in the evaluation set are gold. Note that this is only for evaluation; in training we do not rely on any gold annotations (Section 6.1).

There are four English sub-corpora in OntoNotes Release 5.0[4] that are annotated with dependency labels and coreference chains. Three of them, which are mainly from broadcast news, share similar statistics in document length, so we combine them into a single dataset and name it **ON-SHORT** as it consists mostly of short documents. The fourth subcorpus is from the *Wall Street Journal* and has significantly longer documents. We call this subcorpus **ON-LONG** and evaluate on it separately. Some statistics are shown in Table 2.

	ON-SHORT	ON-LONG
# doc	1027	597
# test cases	13018	18208
Avg # entities	12.06	36.95

Table 2: Statistics on argument cloze datasets.

5.2 The Gerber and Chai (G&C) Dataset

The implicit argument dataset from Gerber and Chai (2010) (referred as **G&C** henceforth) consists of 966 human-annotated implicit argument instances on 10 nominal predicates.

To evaluate our model on G&C, we convert the annotations to the input format of our model as follows: We map nominal predicates to their verbal form, and semantic role labels to syntactic argument types based on the NomBank frame definitions. One of the examples (after mapping semantic role labels) is as follows:

> [Participants]$_{subj}$ will be able to transfer [money]$_{dobj}$ to [other investment funds]$_{prep\_to}$. The [investment]$_{pred}$ choices are limited to [a stock fund and a money-market fund]$_{prep\_to}$.

For the nominal predicate *investment*, there are three arguments missing (*subj*, *dobj*, *prep_to*). The model first needs to determine that each of those argument positions in fact has an implicit filler. Then, from a list of candidates (not shown here), it

[4]LDC Catalog No. LDC2013T19

needs to select *Participants* as the implicit *subj* argument, *money* as the implicit *dobj* argument, and either *other investment funds* or *a stock fund and a money-market fund* as the implicit *prep_to*.

6 Experiments

6.1 Implementation Details

We train our neural model using synthetic data as described in Section 3. For creating the training data, we do not use gold parses or gold coreference chains. We use the 20160901 dump of English Wikipedia[5], with 5,228,621 documents in total. For each document, we extract plain text and break it into paragraphs, while discarding all structured data like lists and tables[6]. We construct a sequence of events and entities from each paragraph, by running Stanford CoreNLP (Manning et al., 2014) to obtain dependency parses and coreference chains. We lemmatize all verbs and arguments. We incorporate negation and particles in verbs, and normalize passive constructions. We represent each argument by the corresponding entity's representative mention if it is linked to an entity, otherwise by its head lemma. We keep verbs and arguments with counts over 500, together with the 50 most frequent prepositions, leading to a vocabulary of 53,345 tokens; all other words are replaced with an out-of-vocabulary token. The most frequent verbs (with counts over 100,000) are down-sampled.

For training the event-based word embeddings, we create pseudo-sentences (Section 4.2) from all events of all sequences (approximately 87 million events) as training samples. We train an SGNS word2vec model with embedding size = 300, window size = 10, subsampling threshold = 10^{-4}, and negative samples = 10, using the Gensim package (Řehůřek and Sojka, 2010).

For training the event composition model, we follow the procedure described in Section 4.2.1, and extract approximately 40 million event triples as training samples[7]. We use a two-layer feed-forward neural network with layer sizes 600 and 300 for the argument composition network, and another two-layer network with layer sizes 400 and 200 for the pair composition network. We use cross-entropy loss with ℓ_2 regularization of 0.01. We train the model using stochastic gradient descent (SGD) with a learning rate of 0.01 and a batch size of 100 for 20 epochs.

To study how the size of the training set affects performance, we downsample the 40 million training samples to another set of 8 million training samples. We refer to the resulting models as **EVENTCOMP-8M** and **EVENTCOMP-40M**.

6.2 Evaluation on Argument Cloze

For the synthetic argument cloze task, we compare our model with 3 baselines.

RANDOM Randomly select one entity from the candidate list.

MOSTFREQ Always select the entity with highest number of mentions.

EVENTWORD2VEC Use the event-based word embeddings described in Section 4.2 for predicates and arguments. The representation of an event e is the sum of the embeddings of its components, i.e.,

$$\vec{e} = \vec{v} + \vec{s} + \vec{o} + \vec{p} \tag{4}$$

where $\vec{v}, \vec{s}, \vec{o}, \vec{p}$ are the embeddings of verb, subject, object, and prepositional object, respectively. The coherence score of two events in this baseline model is their cosine similarity. Like in our main model, the coherence score of the candidate is then the maximum pairwise coherence score, as described in Section 4.1.

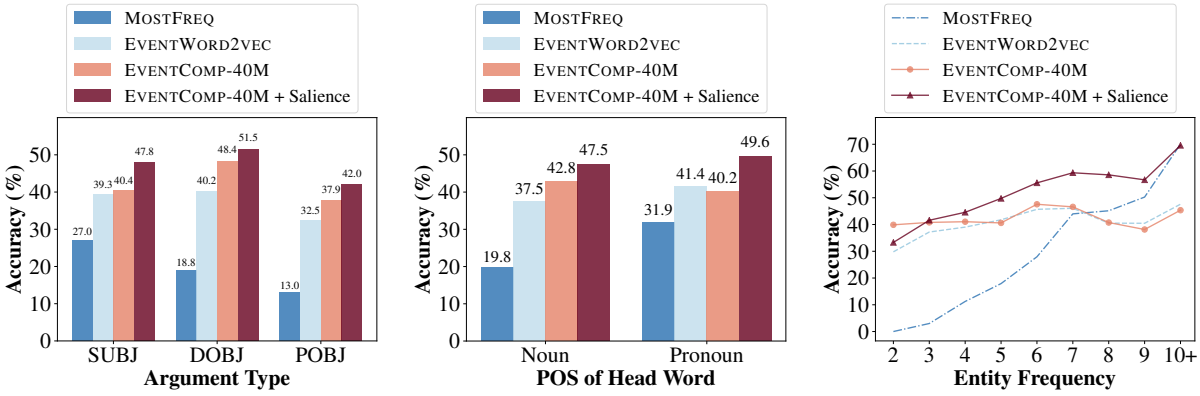
The evaluation results on the ON-SHORT dataset are shown in Table 3. The EVENTWORD2VEC baseline is much stronger than the other two, achieving an accuracy of 38.40%. In fact, EVENTCOMP-8M by itself does not do better than EVENTWORD2VEC, but adding entity salience greatly boosts performance. Using more training data (EVENTCOMP-40M) helps by a substantial margin both with and without entity salience features.

To see which of the entity salience features are important, we conduct an ablation test with the EVENTCOMP-8M model on ON-SHORT. From the results in Table 4, we can see that in our task, as in Dunietz and Gillick (2014), the entity mentions features, i.e., the numbers of named, nominal, pronominal, and total mentions of the entity, are most helpful. In fact, the other two features even decrease performance slightly.

[5] https://dumps.wikimedia.org/enwiki/

[6] We use the WikiExtractor tool at https://github.com/attardi/wikiextractor.

[7] We only sample one negative event for each pair of context and positive events for fast training, though more training samples are easily accessible.



(a) Accuracy by Argument Type (b) Accuracy by POS of Head Word (c) Accuracy by Entity Frequency

Figure 5: Performance of EventComp (with and without entity salience) and two baseline models by (a) argument type, (b) part-of-speech tag of the head word of the entity, and (c) entity frequency.

	Accuracy (%)
Random	8.29
MostFreq	22.76
EventWord2vec	38.40
EventComp-8M	38.26
+ entity salience	45.05
EventComp-40M	41.89
+ entity salience	**47.75**

Table 3: Evaluation on ON-Short.

Features	Accuracy (%)
no entity salience feature	38.26
– *mentions*	39.02
– *head_count*	**45.71**
– *1st_loc*	**45.65**
all entity salience features	45.05

Table 4: Ablation test on entity salience features. (Using EventComp-8M on ON-Short.)

We take a closer look at several of the models in Figure 5. Figure 5a breaks down the results by the argument type of the removed argument. On subjects, the EventWord2vec baseline matches the performance of EventComp, but not on direct objects and prepositional objects. Subjects are semantically much less diverse than the other argument types, as they are very often animate. A similar pattern is apparent in Figure 5b, which has results by the part-of-speech tag of the head word of the removed entity. Note that an entity is a coreference chain, not a single mention; so when the head word is a pronoun, this is an entity which has only pronoun mentions. A pronoun entity provides little semantic content beyond, again, animacy. And again, EventWord2vec performs well on pronoun entities, but less so on entities described by a noun. It seems that EventWord2vec can pick up on a coarse-grained pattern such as animate/inanimate, but not on more fine-grained distinctions needed to select the right noun, or to select a fitting direct object or prepositional object. This matches the fact that EventWord2vec gets a less clear signal on the task, in two respects: It gets much less information than EventComp on the distinction between argument positions,[8] and it only looks at overall event similarity while EventComp is trained to detect narrative coherence. Entity salience contributes greatly across all argument types and parts of speech, but more strongly on subjects and pronouns. This is again because subjects, and pronouns, are semantically less distinct, so they can only be distinguished by relative salience.

Figure 5c analyzes results by the frequency of the removed entity, that is, by its number of mentions. The MostFreq baseline, unsurprisingly, only does well when the removed entity is a highly frequent one. The EventComp model is much better than MostFreq at picking out the right entity when it is a rare one, as it can look at the semantic content of the entity as well as its frequency. Entity salience boosts the performance of EventComp in particular for frequent entities.

The ON-Long dataset, as discussed in Section 5.1, consists of OntoNotes data with much

[8] As shown in Figure 3, the "words" for which embeddings are computed are role-lemma pairs.

longer documents than found in ON-SHORT. Evaluation results on ON-LONG are shown in Table 5. Although the overall numbers are lower than those for ON-SHORT, we are selecting from 36.95 candidates on average, more than 3 times more than for ON-SHORT. Considering that the accuracy of randomly selecting an entity is as low as 2.71%, the performance of our best performing model, with an accuracy of 27.87%, is quite good.

	Accuracy (%)
RANDOM	2.71
MOSTFREQ	17.23
EVENTWORD2VEC	21.49
EVENTCOMP-8M	18.79
+ entity salience	26.23
EVENTCOMP-40M	21.79
+ entity salience	**27.87**

Table 5: Evaluation on ON-LONG.

6.3 Evaluation on G&C

The G&C data differs from the Argument Cloze data in two respects. First, not every argument position that seems to be open needs to be filled: The model must additionally make a **fill / no-fill decision**. Whether a particular argument position is typically filled is highly predicate-specific. As the small G&C dataset does not provide enough data to train our neural model on this task, we instead train a simple logistic classifier, the **fill / no-fill classifier**, with a small subset of shallow lexical features used in Gerber and Chai (2012), to make the decision. These features describe the syntactic context of the predicate. We use only 14 features; the original Gerber and Chai model had more than 80 features, and our re-implementation, described below, has around 60.

The second difference is that in G&C, an event may have multiple open argument positions. In that case, the task is not just to select a candidate entity, but also to determine which of the open argument positions it should fill. So the model must do **multi implicit argument prediction**. We can flexibly adapt our method for training data generation to this case. In particular, we create extra negative training events, in which an argument of the positive event has been moved to another argument position in the same event, as shown in Figure 6. We can then simply train our EVENTCOMP model on this extended training data. We refer to the extra training process as **multi-arg training**.

x_0 = The company x_1 = mill x_2 = power plant

Context: (*build-pred*, x_0-*subj*, x_2-*dobj*, —)
Positive: (*reduce-pred*, **x_2-*subj***, *cost-dobj*, —)
Negative: (*reduce-pred*, —, *cost-dobj*, **x_2-*prep***)

Figure 6: Event triples for training multi implicit argument prediction.

We compare our models to that of Gerber and Chai (2012). However, their original logistic regression model used many features based on gold annotation from FrameNet, PropBank and NomBank. To create a more realistic evaluation setup, we re-implement a variant of their original model by removing gold features, and name it GCAUTO. Results from GCAUTO are directly comparable to our models, as both are trained on automatically generated features. [9]

	P	R	F_1
Gerber and Chai (2012)	57.9	44.5	50.3
GCAUTO	49.9	40.1	44.5
EVENTCOMP-8M	8.9	27.9	13.5
+ fill / no-fill classifier	22.0	22.3	22.1
+ multi-arg training	43.5	44.1	43.8
+ entity salience	45.7	46.4	**46.1**
EVENTCOMP-40M	9.4	30.3	14.3
+ fill / no-fill classifier	23.7	24.0	23.9
+ multi-arg training	46.7	47.3	47.0
+ entity salience	49.3	49.9	**49.6**

Table 6: Evaluation on G&C dataset.

We present the evaluation results in Table 6. The original EVENTCOMP models do not perform well, which is as expected since the model is not designed to do the *fill / no-fill decision* and *multi implicit argument prediction* tasks as described above. With the fill / no-fill classifier, precision rises by around 13 points because this classifier prevents many false positives. With additional multi-arg training, F_1 score improves by another 22-23 points. At this point, our model

[9] To be fair, we also tested adding the fill / no-fill classifier to GCAUTO. However the classifier only increases precision at the cost of reducing recall, and GCAUTO already has higher precision than recall. The resulting F_1 score is actually worse, and thus is not reported here.

achieves a performance comparable to the much more complex G&C reimplementation GCAUTO. Adding entity salience features further boosts both precision and recall, showing that implicit arguments do tend to be filled by salient entities, as we had hypothesized. Again, more training data substantially benefits the task. Our best performing model, at 49.6 F_1, clearly outperforms GCAUTO, and is comparable with the original Gerber and Chai (2012) model trained with gold features. [10]

7 Conclusion

In this paper we have addressed the task of implicit argument prediction. To support training at scale, we have introduced a simple cloze task for which data can be generated automatically. We have introduced a neural model, which frames implicit argument prediction as the task of selecting the textual entity that completes the event in a maximally narratively coherent way. The model prefers salient entities, where salience is mainly defined through the number of mentions. Evaluating on synthetic data from OntoNotes, we find that our model clearly outperforms even strong baselines, that salience is important throughout for performance, and that event knowledge is particularly useful for the (more verb-specific) object and prepositional object arguments. Evaluating on the naturally occurring data from Gerber and Chai, we find that in a comparison without gold features, our model clearly outperforms the previous state-of-the-art model, where again salience information is important.

The current paper takes a first step towards predicting implicit arguments based on narrative coherence. We currently use a relatively simple model for local narrative coherence; in the future we will turn to models that can test global coherence for an implicit argument candidate. We also plan to investigate how the extracted implicit arguments can be integrated into a downstream task that makes use of event information, in particular we would like to experiment with reading comprehension.

Acknowledgments

This research was supported by NSF grant IIS 1523637. We also acknowledge the Texas Advanced Computing Center for providing grid resources that contributed to these results, and we would like to thank the anonymous reviewers for their valuable feedback.

[10] We also tried fine tune our model on the G&C dataset with cross validation, but the model severely overfit, possibly due to the very small size of the dataset.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*. http://www.aclweb.org/anthology/P98-1013.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of ACL-08: HLT*. Association for Computational Linguistics, pages 789–797. http://www.aclweb.org/anthology/P08-1090.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Association for Computational Linguistics, pages 602–610. http://www.aclweb.org/anthology/P09-1068.

Quynh Ngoc Thi Do, Steven Bethard, and Marie-Francine Moens. 2017. Improving implicit semantic role labeling by predicting semantic frame arguments. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Asian Federation of Natural Language Processing, pages 90–99. http://www.aclweb.org/anthology/I17-1010.

Jesse Dunietz and Daniel Gillick. 2014. A new entity salience task with millions of training examples. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*. Association for Computational Linguistics, pages 205–209. https://doi.org/10.3115/v1/E14-4040.

Matthew Gerber and Joyce Chai. 2010. Beyond NomBank: A study of implicit arguments for nominal predicates. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 1583–1592. http://www.aclweb.org/anthology/P10-1160.

Matthew Gerber and Joyce Y. Chai. 2012. Semantic role labeling of implicit arguments for nominal predicates. *Computational Linguistics* 38(4). https://doi.org/10.1162/COLI_a_00110.

Mark Granroth-Wilding and Stephen Clark. 2016. What happens next? event prediction using a compositional neural network model. In *AAAI Conference on Artificial Intelligence*. pages 2727–2733.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*. pages 1693–1701.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*. http://www.aclweb.org/anthology/N06-2015.

Egoitz Laparra and German Rigau. 2013. ImpAr: A deterministic algorithm for implicit semantic role labelling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1180–1189. http://www.aclweb.org/anthology/P13-1116.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, pages 55–60. https://doi.org/10.3115/v1/P14-5010.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. The NomBank project: An interim report. In *Proceedings of the Workshop Frontiers in Corpus Annotation at HLT-NAACL 2004*. pages 24–31. http://www.aclweb.org/anthology/W04-2705.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Ashutosh Modi, Ivan Titov, Vera Demberg, Asad Sayeed, and Manfred Pinkal. 2017. Modelling semantic expectation: Using script knowledge for referent prediction. *Transactions of the Association of Computational Linguistics* 5:31–44. http://www.aclweb.org/anthology/Q17-1003.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics* 31(1). http://www.aclweb.org/anthology/J05-1004.

Karl Pichotta and Raymond J Mooney. 2016a. Learning statistical scripts with LSTM recurrent neural networks. In *AAAI Conference on Artificial Intelligence*. pages 2800–2806.

Karl Pichotta and Raymond J. Mooney. 2016b. Using sentence-level LSTM language models for script inference. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 279–289. https://doi.org/10.18653/v1/P16-1027.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, Valletta, Malta, pages 45–50. http://is.muni.cz/publication/884893/en.

Rachel Rudinger, Pushpendre Rastogi, Francis Ferraro, and Benjamin Van Durme. 2015. Script induction as language modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1681–1686. https://doi.org/10.18653/v1/D15-1195.

Josef Ruppenhofer, Caroline Sporleder, Roser Morante, Collin Baker, and Martha Palmer. 2010. SemEval-2010 Task 10: Linking events and their participants in discourse. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, pages 45–50. http://www.aclweb.org/anthology/S10-1008.

Roger C Schank and Robert Abelson. 1977. Scripts, goals, plans, and understanding.

Niko Schenk and Christian Chiarcos. 2016. Unsupervised learning of prototypical fillers for implicit semantic role labeling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 1473–1479. https://doi.org/10.18653/v1/N16-1173.

Improving Temporal Relation Extraction with a Globally Acquired Statistical Resource

Qiang Ning,[1] **Hao Wu,**[2] **Haoruo Peng,**[1] **Dan Roth**[1,2]
Department of Computer Science
[1]University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
[2]University of Pennsylvania, Philadelphia, PA 19104, USA
{qning2,hpeng7}@illinois.edu, {haowu4,danroth}@seas.upenn.edu

Abstract

Extracting temporal relations (*before, after, overlapping*, etc.) is a key aspect of understanding events described in natural language. We argue that this task would gain from the availability of a resource that provides prior knowledge in the form of the temporal order that events *usually* follow. This paper develops such a resource – a probabilistic knowledge base acquired in the news domain – by extracting temporal relations between events from the New York Times (NYT) articles over a 20-year span (1987–2007). We show that existing temporal extraction systems can be improved via this resource. As a byproduct, we also show that interesting statistics can be retrieved from this resource, which can potentially benefit other time-aware tasks. The proposed system and resource are both publicly available[1].

1 Introduction

Time is an important dimension of knowledge representation. In natural language, temporal information is often expressed as relations between events. Reasoning over these relations can help figuring out when things happened, estimating how long things take, and summarizing the timeline of a series of events. Several recent SemEval workshops are a good showcase of the importance of this topic (Verhagen et al., 2007, 2010; Uz-Zaman et al., 2013; Llorens et al., 2015; Minard et al., 2015; Bethard et al., 2015, 2016, 2017).

One of the challenges in temporal relation extraction is that it requires high-level prior knowledge of the temporal order that events *usually* follow. In Example 1, we have deleted events from several snippets from CNN, so that we cannot use our prior knowledge of those events. We are also told that ***e1*** and ***e2*** have the same tense, and ***e3*** and ***e4*** have the same tense, so we cannot resort to their tenses to tell which one happens earlier. As a result, it is very difficult even for humans to figure out the temporal relations (referred to as "TempRels" hereafter) between those events. This is because rich temporal information is encoded in the events' names, and this often plays an indispensable role in making our decisions. In the first paragraph of Example 1, it is difficult to understand what really happened without the actual event verbs; let alone the TempRels between them. In the second paragraph, things are even more interesting: if we had ***e3:dislike*** and ***e4:stop***, then we would know easily that "I dislike" occurs *after* "they stop the column". However, if we had ***e3:ask*** and ***e4:help***, then the relation between ***e3*** and ***e4*** is now reversed and ***e3*** is *before* ***e4***. We are in need of the event names to determine the TempRels; however, we do not have them in Example 1. In Example 2, where we show the complete sentences, the task has become much easier for humans due to our prior knowledge, namely, that explosion *usually* leads to casualties and that people *usually* ask before they get help. Motivated by these examples (which are in fact very common), we believe in the importance of such a prior knowledge in determining TempRels between events.

Example 1: Difficulty in understanding TempRels when event content is missing. Note that ***e1*** and ***e2*** have the same tense, and ***e3*** and ***e4*** have the same tense.
More than 10 people have (***e1:***), police said. A car (***e2:***) on Friday in the middle of a group of men playing volleyball.
The first thing I (***e3:***) is that they (***e4:***) writing this column.

However, most existing systems only make use of rather local features of these events, which cannot represent the prior knowledge humans have

[1] http://cogcomp.org/page/publication_view/830

Proceedings of NAACL-HLT 2018, pages 841–851
New Orleans, Louisiana, June 1 - 6, 2018.

Example Pairs		Before (%)	After (%)
accept	determine	42	26
ask	help	86	9
attend	schedule	1	82
accept	propose	10	77
die	explode	14	83
...			

Table 1: **TEMPROB is a unique source of information of the temporal order that events *usually* follow.** The probabilities below do not add up to 100% because less frequent relations are omitted. The word sense numbers are not shown here for convenience.

about these events and their "typical" order. As a result, existing systems almost always attempt to solve the situations shown in Example 1, even when they are actually presented with input as in Example 2. The **first contribution** of this work is thus the construction of such a resource in the form of a probabilistic knowledge base, constructed from a large New York Times (NYT) corpus. We hereafter name our resource *TEMporal relation PRObabilistic knowledge Base* (TEMPROB), which can potentially benefit many time-aware tasks. A few example entries of TEMPROB are shown in Table 1. **Second**, we show that existing TempRel extraction systems can be improved using TEMPROB, either in a local method or in a global method (explained later), by a significant margin in performance on the benchmark TimeBank-Dense dataset (Cassidy et al., 2014).

Example 2: The original sentences in Example 1.
More than 10 people have (***e1:died***), police said. A car (***e2:exploded***) on Friday in the middle of a group of men playing volleyball.
The first thing I (***e3:ask***) is that they (***e4:help***) writing this column.

The rest of the paper is organized as follows. Section 2 provides a literature review of TempRels extraction in NLP. Section 3 describes in detail the construction of TEMPROB. In Sec. 4, we show that TEMPROB can be used in existing TempRels extraction systems and lead to significant improvement. Finally, we conclude in Sec. 5.

2 Related Work

The TempRels between events can be represented by an edge-labeled graph, where the nodes are events, and the edges are labeled with TempRels (Chambers and Jurafsky, 2008; Do et al., 2012; Ning et al., 2017). Given all the nodes, we work on the TempRel extraction task, which is to assign labels to the edges in a temporal graph (a "vague" or "none" label is often included to account for the non-existence of an edge).

Early work includes Mani et al. (2006); Chambers et al. (2007); Bethard et al. (2007); Verhagen and Pustejovsky (2008), where the problem was formulated as learning a classification model for determining the label of every edge *locally* without referring to other edges (i.e., local methods). The predicted temporal graphs by these methods may violate the transitive properties that a temporal graph should possess. For example, given three nodes, ***e1***, ***e2***, and ***e3***, a local method can possibly classify (***e1***,***e2***)=*before*, (***e2***,***e3***)=*before*, and (***e1***,***e3***)=*after*, which is obviously wrong since *before* is a transitive relation and (***e1***,***e2***)=*before* and (***e2***,***e3***)=*before* dictate that (***e1***,***e3***)=*before*. Recent state-of-the-art methods, (Chambers et al., 2014; Mirza and Tonelli, 2016), circumvented this issue by growing the predicted temporal graph in a multi-step manner, where transitive graph closure is performed on the graph every time a new edge is labeled. This is conceptually solving the structured prediction problem greedily. Another family of methods resorted to Integer Linear Programming (ILP) (Roth and Yih, 2004) to get exact inference to this problem (i.e., global methods), where the entire graph is solved simultaneously and the transitive properties are enforced naturally via ILP constraints (Bramsen et al., 2006; Chambers and Jurafsky, 2008; Denis and Muller, 2011; Do et al., 2012). A most recent work brought this idea even further, by incorporating structural constraints into the learning phase as well (Ning et al., 2017).

The TempRel extraction task has a strong dependency on prior knowledge, as shown in our earlier examples. However, very limited attention has been paid to generating such a resource and to make use of it; to our knowledge, the TEMPROB proposed in this work is completely new. We find that the *time-sensitive relations* proposed in Jiang et al. (2016) is a close one in literature (although it is still very different). Jiang et al. (2016) worked on the knowledge graph completion task. Based on YAGO2 (Hoffart et al., 2013) and Freebase (Bollacker et al., 2008), it manually selects a small number of relations that are time-sensitive (10 relations from YAGO2 and 87 relations from Freebase, respectively). Exemplar relations are wasBornIn→diedIn→ and graduateFrom→workAt, where → means temporally be-

fore.

Our work significantly differs from the time-sensitive relations in Jiang et al. (2016) in the following aspects. First, scale difference: Jiang et al. (2016) can only extract a small number of relations (<100), but we work on general semantic frames (tens of thousands) and the relations between any two of them, which we think has broader applications. Second, granularity difference: the smallest granularity in Jiang et al. (2016) is one year[2], i.e., only when two events happened in different years can they know the temporal order of them, but we can handle implicit temporal orders without having to refer to the physical time points of events (i.e., the granularity can be arbitrarily small). Third, domain difference: while Jiang et al. (2016) extracts time-sensitive relations from structured knowledge bases (where events are explicitly anchored to a time point), we extract relations from unstructured natural language text (where the physical time points may not even exist in text). Our task is more general and it allows us to extract much more relations, as reflected by the 1st difference above.

Another related work is the VerbOcean (Chklovski and Pantel, 2004), which extracts temporal relations between pairs of verbs using manually designed lexico-syntactic patterns (there are in total 12 such patterns), in contrast to the automatic extraction method proposed in this work. In addition, the only termporal relation considered in VerbOceans is *before*, while we also consider relations such as *after*, *includes*, *included*, *equal*, and *vague*. As expected, the total numbers of verbs and *before* relations in VerbOcean is about 3K and 4K, respectively, both of which are much smaller than TEMPROB, which contains 51K verb frames (i.e., disambiguated verbs), 9.2M $(verb1, verb2, relation)$ entries, and up to 80M temporal relations altogether.

All these differences necessitate the construction of a new resource for TempRel extraction, which we explain below.

3 TEMPROB: A Probabilistic Resource for TempRels

In the TempRel extraction task, people have usually assumed that events are already given. However, to construct the desired resource, we need to extract events (Sec. 3.1) and extract TempRels (Sec. 3.2), from a large, unannotated[3] corpus (Sec. 3.3). We also show some interesting statistics discovered in TEMPROB that may benefit other tasks (Sec. 3.4). In the next, we describe each of these elements.

3.1 Event Extraction

Extracting events and the relations between them (e.g., coreference, causality, entailment, and temporal) have long been an active area in the NLP community. Generally speaking, an event is considered to be an action associated with corresponding participants involved in this action. In this work, following (Peng and Roth, 2016; Peng et al., 2016; Spiliopoulou et al., 2017) we consider semantic-frame based events, which can be directly detected via off-the-shelf semantic role labeling (SRL) tools. This aligns well with previous works on event detection (Hovy et al., 2013; Peng et al., 2016).

Depending on the events of interest, the SRL results are often a superset of events and need to be filtered afterwards (Spiliopoulou et al., 2017). For example, in ERE (Song et al., 2015) and Event Nugget Detection (Mitamura et al., 2015), events are limited to a set of predefined types (such as "Business", "Conflict", and "Justice"); in the context of TempRels, existing datasets have focused more on predicate verbs rather than nominals[4] (Pustejovsky et al., 2003; Graff, 2002; UzZaman et al., 2013). Therefore, we only look at verb semantic frames in this work due to the difficulty of getting TempRel annotation for nominal events, and we will use "verb (semantic frames)" interchangeably with "events" hereafter in this paper.

3.2 TempRel Extraction

Given the events extracted in a given article (i.e., given the nodes in a graph), we next explain how the TempRels are extracted (that is, the edge labels in the graph).

3.2.1 Features

We adopt the commonly used feature set in TempRel extraction (Do et al., 2012; Ning et al., 2017) and here we simply list them for reproducibility. For each pair of nodes, the follow-

[2]We notice that the smallest granularity in Freebase itself is one day, but Jiang et al. (2016) only used years.

[3]Unannotated with TempRels.

[4]Some nominal events were indeed annotated in TimeBank (Pustejovsky et al., 2003), but their annotation did not align well with modern nominal-SRL methods.

ing features are extracted. (i) The part-of-speech (POS) tags from each individual verb and from its neighboring three words. (ii) The distance between them in terms of the number of tokens. (iii) The modal verbs between the event mention (i.e., will, would, can, could, may and might). (iv) The temporal connectives between the event mentions (e.g., before, after and since). (v) Whether the two verbs have a common synonym from their synsets in WordNet (Fellbaum, 1998). (vi) Whether the input event mentions have a common derivational form derived from WordNet. (vii) The head word of the preposition phrase that covers each verb, respectively.

3.2.2 Learning

With the features defined above, we need to train a system that can annotate the TempRels in each document. The TimeBank-Dense dataset (TB-Dense) (Cassidy et al., 2014) is known to have the best quality in terms of its high density of TempRels and is a benchmark dataset for the TempRel extraction task. It contains 36 documents from TimeBank (Pustejovsky et al., 2003) which were re-annotated using the dense event ordering framework proposed in (Cassidy et al., 2014). We follow its label set (denoted by R) of *before*, *after*, *includes*, *included*, *equal*, and *vague* in this study.

Due to the slight event annotation difference in TBDense, we collect our training data as follows. We first extract all the verb semantic frames from the raw text of TBDense. Then we only keep those semantic frames that are matched to an event in TBDense (about 85% semantic frames are kept in this stage). By doing so, we can simply use the TempRel annotations provided in TBDense. Hereafter the TBDense dataset used in this paper refers to this version unless otherwise specified.

We group the TempRels by the sentence distance of the two events of each relation[5]. Then we use the averaged perceptron algorithm (Freund and Schapire, 1998) implemented in the Illinois LBJava package (Rizzolo and Roth, 2010) to learn from the training data described above. Since only relations that have sentence distance 0 or 1 are annotated in TBDense, we will have two classifiers, one for same sentence relations, and one for neighboring sentence relations, respectively.

[5]That is, the difference of the appearance order of the sentence(s) containing the two target events.

Note that TBDense was originally split into Train (22 docs), Dev (5 docs), and Test (9 docs). In all subsequent analysis, we combined Train and Dev and we performed 3-fold cross validation on the 27 documents (in total about 10K relations) to tune the parameters in any classifier.

3.2.3 Inference

When generating TEMPROB, we need to process a large number of articles, so we adopt the greedy inference strategy described earlier due to its computational efficiency (Chambers et al., 2014; Mirza and Tonelli, 2016). Specifically, we apply the same-sentence relation classifier before the neighboring-sentence relation classifier; whenever a new relation is added in this article, a transitive graph closure is performed immediately. By doing this, if an edge is already labeled during the closure phase, it will not be labeled again, so conflicts are avoided.

3.3 Corpus

As mentioned earlier, the source corpus on which we are going to construct TEMPROB is comprised of NYT articles from 20 years (1987-2007)[6]. It contains more than 1 million documents and we extract events and corresponding features from each document using the Illinois Curator package (Clarke et al., 2012) on Amazon Web Services (AWS) Cloud. In total, we discovered 51K unique verb semantic frames and 80M relations among them in the NYT corpus (15K of the verb frames had more than 20 relations extracted and 9K had more than 100 relations).

3.4 Interesting Statistics

We first describe the notations that we are going to use. We denote the set of all verb semantic frames by V. Let $D_i, i = 1, \ldots, N$ be the i-th document in our corpus, where N is the total number of documents. Let $G_i = (V_i, E_i)$ be the temporal graph inferred from D_i using the approach described above, where $V_i \subseteq V$ is the set of verbs/events extracted in D_i and $E_i = \{(v_m, v_n, r_{mn})\}_{m<n} \subseteq V_i \times V_i \times R$ is the edge set of D_i, which is composed of TempRel triplets; specifically, a TempRel triplet $(v_m, v_n, r_{mn}) \in E_i$ represents that in document D_i, the TempRel between v_m and v_n is r_{mn}. Due to the symmetry in TempRels, we only keep the triplets with $m < n$ in E_i. Assuming that the

[6]https://catalog.ldc.upenn.edu/LDC2008T19

verbs in V_i are ordered by their appearance order in text, then $m < n$ means that in the i-th document, v_m appears earlier in text than v_n does.

Given the usual confusion between that one event is *temporally before* another and that one event is *physically appearing* before another in text, we will refer to temporally before as **T-Before** and physically before as **P-Before**. Using this language, for example, E_i only keeps the triplets that v_m is P-Before v_n in D_i.

3.4.1 Extreme cases

We first show extreme cases that some events are *almost always* labeled as T-Before or T-After in the corpus. Specifically, for each pair of verbs $v_i, v_j \in V$, we define the following ratios:

$$\eta_b = \frac{C(v_i, v_j, before)}{C(v_i, v_j, before) + C(v_i, v_j, after)}, \eta_a = 1 - \eta_b, \tag{1}$$

where $C(v_i, v_j, r)$ is the count of v_i P-Before v_j with TempRel $r \in R$:

$$C(v_i, v_j, r) = \sum_{i=1}^{N} \sum_{(v_m, v_n, r_{mn}) \in E_i} \mathcal{I}_{\{v_m = v_i \& v_n = v_j \& r_{mn} = r\}}, \tag{2}$$

where $\mathcal{I}_{\{\cdot\}}$ is the indicator function. Add-one smoothing technique from language modeling is used to avoid divided-by-zero errors. In Table 2, we show some event pairs with either $\eta_b > 0.9$ (upper part) or $\eta_a > 0.9$ (lower part).

We think the examples from Table 2 are intuitively appealing: ***chop*** happens before ***taste***, ***clean*** happens after ***contaminate***, etc. More interestingly, in the lower part of the table, we show pairs in which the physical order is different from the temporal order: for example, when ***achieve*** is P-Before ***desire***, it is still labeled as T-After in most cases (104 out of 111 times), which is correct intuitively. In practice, e.g., in the TBDense dataset (Cassidy et al., 2014), roughly 30%-40% of the P-Before pairs are T-After. Therefore, it is important to be able to capture their temporal order rather than simply taking their physical order if one wants to understand the temporal implication of verbs.

3.4.2 Distribution of Following Events

For each verb v, we define the marginal count of v being P-Before to arbitrary verbs with TempRel $r \in R$ as $C(v, r) = \sum_{v_i \in V} C(v, v_i, r)$. Then for every other verb v', we define

$$P(v \text{ T-Before } v' | v \text{ T-Before}) \triangleq \frac{C(v, v', before)}{C(v, before)}, \tag{3}$$

Example Pairs		#T-Before	#T-After
chop.01	taste.01	133	8
concern.01	protect.01	110	10
conspire.01	kill.01	113	6
debate.01	vote.01	48	5
dedicate.01	promote.02	67	7
fight.01	overthrow.01	98	8
achieve.01	desire.01	7	104
admire.01	respect.01	7	121
clean.02	contaminate.01	3	82
defend.01	accuse.01	13	160
die.01	crash.01	8	223
overthrow.01	elect.01	3	100

Table 2: **Several extreme cases from TEMPROB**, where some event is almost always labeled to be T-Before or T-After throughout the NYT corpus. By "extreme", we mean that either the probability of T-Before or T-After is larger than 90%. The upper part of the table shows the pairs that are both P-Before and T-Before, while the lower part shows the pairs that are P-Before but T-After. In TEMPROB, there are about 7K event pairs being extreme cases.

which is the probability of v T-Before v', conditioned on v T-Before anything. Similarly, we define

$$P(v \text{ T-After } v' | v \text{ T-After}) \triangleq \frac{C(v, v', after)}{C(v, after)}. \tag{4}$$

For a specific verb, e.g., v=***investigate***, each verb $v' \in V$ is sorted by the two conditional probabilities above. Then the most probable verbs that temporally precede or follow v are shown in Fig. 1, where the y-axes are the corresponding conditional probabilities. We can see reasonable event sequences like {***involve***, ***kill***, ***suspect***, ***steal***}→***investigate***→{***report***, ***prosecute***, ***pay***, ***punish***}, which indicates the possibility of using TEMPROB for event sequence predictions or story cloze tasks. There are also suspicious pairs like ***know*** in the T-Before list of ***investigate*** (Fig. 1a), ***report*** in the T-Before list of ***bomb*** (Fig. 1b), and ***play*** in the T-After list of ***mourn*** (Fig. 1c). Since the arguments of these verb frames are not considered here, whether these few seemingly counter-intuitive pairs come from system error or from a special context needs further investigation.

4 Experiments

In the above, we have explained the construction of TEMPROB and shown some interesting examples from it, which were meant to visualize its correctness. In this section, we first quantify the correctness of the prior obtained in TEMPROB, and

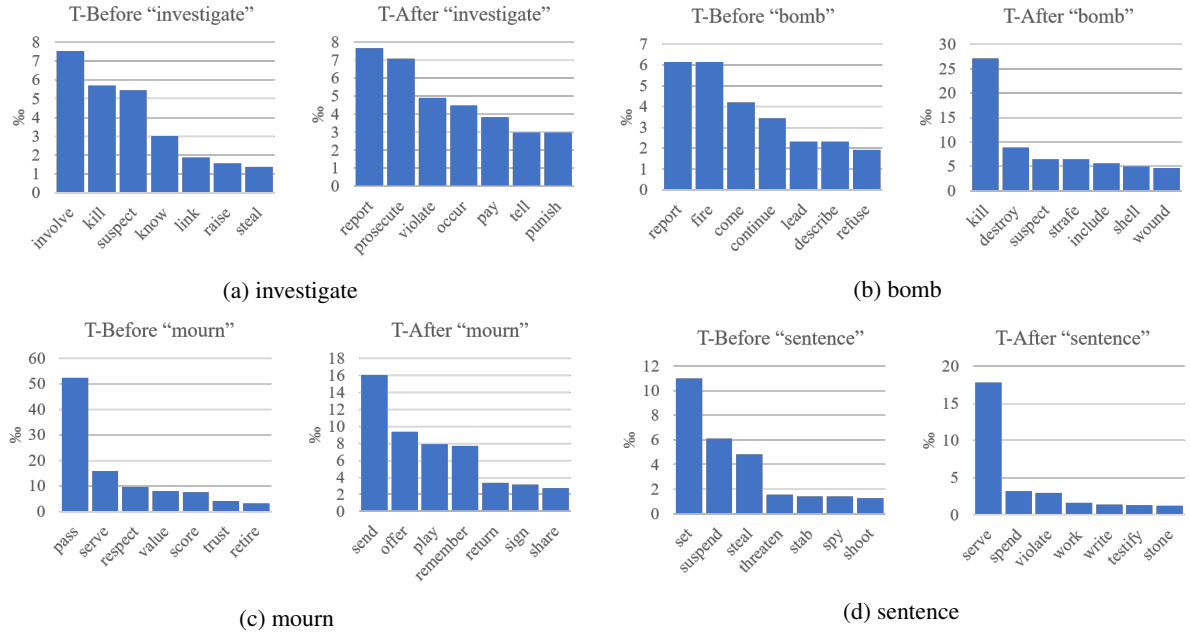


Figure 1: **Top events that most frequently precede or follow "investigate", "bomb", "mourn", or "sentence" in time**, sorted by their conditional probabilities in ‰. Word senses have been disambiguated and the "bomb" and "sentence" here are their verb meanings. There are some possible errors (e.g., ***report*** is T-Before ***bomb***) and some unclear pairs (e.g., ***know*** is T-Before ***investigate*** and ***play*** is T-After ***mourn***), but overall the event sequences discovered here are reasonable. More examples can be found in the appendix.

then show TEMPROB can be used to improve existing TempRel extraction systems.

4.1 Quality Analysis of TEMPROB

In Table 2, we showed examples with either η_b or $\eta_a > 0.9$. We argued that they *seem* correct. Here we quantify the "correctness" of η_b and η_a based on TBDense. Specifically, we collected all the gold T-Before and T-After pairs. Let $\tau \in [0.5, 1)$ be a constant threshold. Imagine a naive predictor such that for each pair of events v_i and v_j, if $\eta_b > \tau$, it predicts that v_i is T-Before v_j; if $\eta_a > \tau$, it predicts that v_i is T-After v_j; otherwise, it predicts that v_i is T-Vague to v_j. We expect that a higher η_b (or η_a) represents a higher confidence for an instance to be labeled T-Before (or T-After).

Table 3 shows the performance of this predictor, which meets our expectation and thus justifies the validity of TEMPROB. As we gradually increase the value of τ in Table 3, the precision increases in roughly the same pace with τ, which indicates that the values of η_b and η_a[7] from TEMPROB indeed represent the confidence level. The decrease in recall is also expected because more examples are labeled as T-Vague when τ is larger.

To further justify the quality, we also used another dataset that is not in the TempRel domain. Instead, we downloaded the EventCausality dataset[8] (Do et al., 2011). For each causally related pair ***e1*** and ***e2***, if EventCausality annotates that ***e1*** causes ***e2***, we changed it to be T-Before; if EventCausality annotates that ***e1*** is caused by ***e2***, we changed it to be T-after. Therefore, based on the assumption that the cause event is T-Before the result event, we converted the EventCausality dataset to be a TempRel dataset and it thus could also be used to evaluate the quality of TEMPROB. We adopted the same predictor used in Table 3

Threshold τ	Dist=0		Dist=1	
	P	R	P	R
0.5	65.6	61.3	58.5	53.3
0.6	69.8	44.5	60.5	36.9
0.7	74.6	29.2	63.6	18.7
0.8	81.0	13.9	64.8	6.9
0.9	82.9	5.0	76.9	1.2

Table 3: **Validating η_b and η_a from TEMPROB** based on the T-Before and T-After examples in TBDense. Performances are decomposed into same sentence examples (Dist=0) and contiguous sentence examples (Dist=1). A larger threshold leads to a higher precision, so η_b and η_a indeed represent a notion of confidence.

[7] Recall the definitions of η_b and η_a in Eq. (1).

[8] http://cogcomp.org/page/resource_view/27

with $\tau = 0.5$ and in Table 4, we compared it with two baselines: (i) always predicting T-Before and (ii) always predicting T-After. First, the accuracy (66.2%) in Table 4 is rather consistent with its counterpart in Table 3, confirming the stability of statistics from TEMPROB. Second, by directly using the prior statistics η_b and η_a from TEMPROB, we can improve the precision of both labels with a significant margin relative to the two baselines (17.0% for "T-Before" and 15.9% for "T-After"). Overall, the accuracy was improved by 11.5%.

System	T-Before		T-After		Acc.
	P	R	P	R	
T-Before Only	54.7	100.0	0	0	54.7
T-After Only	0	0	45.3	100	45.3
$\tau = 0.5$	71.7	63.3	61.2	69.8	66.2

Table 4: **Further justification of η_b and η_a from TEMPROB on the EventCausality dataset**. The thresholding predictor from Table 3 with $\tau = 0.5$ is used here. Compared to always predicting the majority label (i.e., T-Before in this case), $\tau = 0.5$ significantly improved the performance for both labels, with the overall accuracy improved by 11.5%.

4.2 Improving TempRel Extraction

The original purpose of TEMPROB was to improve TempRel extraction. We show it from two perspectives: How effective the prior distributions obtained from TEMPROB are (i) as features in local methods and (ii) as regularization terms in global methods. The results below were evaluated on the test split of TB-Dense (Cassidy et al., 2014).

4.2.1 Improving Local Methods

We first test how well the prior distributions from TEMPROB can be used as features in improving local methods for TempRel extraction. In Table 5, we used the original feature set proposed in Sec. 3.2.1 as the baseline, and added the prior distribution obtained from TEMPROB on top of it. Specifically, we added η_b (see Eq. (1)) and $\{f_r\}_{r\in R}$, respectively, where $\{f_r\}_{r\in R}$ is the prior distributions of all labels, i.e.,

$$f_r(v_i, v_j) = \frac{C(v_i, v_j, r)}{\sum_{r'\in R} C(v_i, v_j, r')},\ r \in R. \quad (5)$$

Recall function C is defined in Eq. (2). All comparisons were decomposed to same sentence relations (Dist=0) and neighboring sentence relations (Dist=1) for a better understanding of the behavior. All classifiers were trained using the averaged perceptron algorithm (Freund and Schapire, 1998) and tuned by 3-fold cross validation.

From Table 5, we can see that simply adding η_b into the feature set could improve the original system F_1 by 1.8% (Dist=0) and 3.0% (Dist=1). If we further add as features the full set of prior distributions $\{f_r\}_{r\in R}$, the improvement comes to 2.7% and 6.5%, respectively. Noticing that the feature is more helpful for Dist=1, we think that it is because distant pairs usually have less lexical dependency and thus need more prior information provided by our new feature. With Dist=0 and Dist=1 combined (numbers not shown in the Table), the 3rd line improved the "original" by 4.7% in F_1 and by 5.1% in the temporal awareness F-score (another metric used in the TempEval3 workshop).

Feature Set	Dist=0			Dist=1		
	P	R	F_1	P	R	F_1
Original	44.5	57.1	50.0	49.0	36.9	42.1
$+\eta_b$	46.2	58.9	51.8	55.3	38.1	45.1
$+\{f_r\}_{r\in R}$	46.9	60.1	52.7	51.3	46.2	48.6

Note The performances here are consistently lower than those in Table 3 because in Table 3, only T-Before and T-After examples are considered, but here all labels are taken into account and the problem is more practical and harder.

Table 5: **Using prior distributions derived from TEMPROB as features in an example local method.** Incorporating η_b to the original feature set already yields better performance. By using the full set of prior distributions, $\{f_r\}_{r\in R}$, the final system improves the original in almost all metrics, and the improvement is statistically significant with $p<0.005$ per the McNemar's test.

4.2.2 Improving Global Methods

As mentioned earlier in Sec. 2, many systems adopt a global inference method via integer linear programming (ILP) (Roth and Yih, 2004) to enforce transitivity constraints over an entire temporal graph (Bramsen et al., 2006; Chambers and Jurafsky, 2008; Denis and Muller, 2011; Do et al., 2012; Ning et al., 2017). In addition to the usage shown in Sec. 4.2.1, the prior distributions from TEMPROB can also be used to regularize the conventional ILP formulation. Specifically, in each document, let $\mathcal{I}_r(ij) \in \{0, 1\}$ be the indicator function of relation r for event i and event j; let $x_r(ij) \in [0, 1]$ be the corresponding softmax score obtained from the local classifiers (depending on the sentence distance between i and j). Then the ILP objective for global inference is

formulated as follows.

$$\hat{\mathcal{I}} = \underset{\mathcal{I}}{\text{argmax}} \sum_{ij \in \mathcal{E}} \sum_{r \in R} (x_r(ij) + \underline{\lambda f_r(ij)}) \mathcal{I}_r(ij) \quad (6)$$

$$\text{s.t.} \quad \underset{\text{(uniqueness)}}{\Sigma_r \mathcal{I}_r(ij) = 1}, \ \underset{\text{(symmetry)}}{\mathcal{I}_r(ij) = \mathcal{I}_{\bar{r}}(ji)},$$

$$\underset{\text{(transitivity)}}{\mathcal{I}_{r_1}(ij) + \mathcal{I}_{r_2}(jk) - \Sigma_{m=1}^{M} \mathcal{I}_{r_3^m}(ik) \leq 1},$$

for all distinct events i, j, and k, where $\mathcal{E} = \{ij \mid \text{sentence dist}(i,j) \leq 1\}$, λ adjusts the regularization term and was heuristically set to 0.5 in this work, $\bar{r}$ is the reverse relation of r, and M is the number of possible relations for r_3 when r_1 and r_2 are true. Note our difference from the ILP in (Ning et al., 2017) is the underlined regularization term $f_r(ij)$ (which itself is defined in Eq. (5)) obtained from TEMPROB.

No.	System	P	R	F_1	F_{aware}
1	Baseline	48.1	44.4	46.2	42.5
2	+Feature: $\{f_r\}_{r \in R}$	50.6	52.0	51.3	49.1
3	+Regularization	51.3	53.0	52.1	49.6

Table 6: **Regularizing global methods by the prior distribution derived from TEMPROB.** The "+" means adding a component on top of its preceding line. F_{aware} is the temporal awareness F-score, another evaluation metric used in TempEval3. The baseline system is to use (unregularized) ILP on top of the original system in Table 5. System 3 is the proposed. Per the McNemar's test, System 3 is significantly better than System 1 with $p<0.0005$.

We present our results on the test split of TBDense in Table 6, which is an ablation study showing step-by-step improvements in two metrics. In addition to the straightforward precision, recall, and F_1 metric, we also compared the F_1 of the temporal awareness metric used in TempEval3 (UzZaman et al., 2013). The awareness metric performs graph reduction and closure before evaluation so as to better capture how useful a temporal graph is. Details of this metric can be found in UzZaman and Allen (2011); UzZaman et al. (2013); Ning et al. (2017).

In Table 6, the baseline used the original feature set proposed in Sec. 3.2.1 and applied global ILP inference with transitivity constraints. Technically, it is to solve Eq. (6) with $\lambda = 0$ (i.e., unregularized) on top of the original system in Table 5. Apart from some implementation details, this baseline is also the same as many existing global methods as Chambers and Jurafsky (2008); Do et al. (2012). System 2, "+Feature: $\{f_r\}_{r \in R}$", is to add prior distributions as features when training the local classifiers. Technically, the scores $x_r(ij)$'s in Eq. (6) used by baseline were changed. We know from Table 5 that adding $\{f_r\}_{r \in R}$ made the local decisions better. Here the performance of System 2 shows that this was also the case for the global decisions made via ILP: both precision and recall got improved, and F_1 and awareness were both improved by a large margin, with 5.1% in F_1 and 6.6% in awareness F_1. On top of this, System 3 sets $\lambda = 0.5$ in Eq. (6) to add regularizations to the conventional ILP formulation. The sum of these regularization terms represents a confidence score of how coherent the predicted temporal graph is to our TEMPROB, which we also want to maximize. Even though a considerable amount of information from TEMPROB had already been encoded as features (as shown by the large improvements by System 2), these regularizations were still able to further improve the precision, recall and awareness scores. To sum up, the total improvement over the baseline system brought by TEMPROB is 5.9% in F_1 and 7.1% in awareness F_1, both with a notable margin. Table 7 furthermore decomposes this improvement into each TempRel label.

Label	P	R	F_1
before	+0.3	+15	+6
after	+4	+4	+4
equal	+11	0	+2
includes	+17	0	+0.2
included	+8	0	+2
vague	+3	-4	-1

Table 7: **Label-wise performance improvement of System 3 over System 1 in Table 6.** We can see that incorporating TEMPROB improves the recall of *before* and *after*, and improves the precision of all labels, with a slight drop in the recall of *vague*.

To compare with state-of-the-art systems, which all used gold event properties (i.e., Tense, Aspect, Modality, and Polarity), we retrained System 3 in Table 6 with these gold properties and show the results in Table 8. We reproduced the results of CAEVO[9] (Chambers et al., 2014) and Ning et al. (2017)[10] and evaluated them on the partial TBDense test split[11]. Under both metrics, the

[9] https://github.com/nchambers/caevo

[10] http://cogcomp.org/page/publication_view/822

[11] There are 731 relations in the partial TBDense test split (201 *before*, 138 *after*, 39 *includes*, 31 *included*, 14 *equal*, and 308 *vague*).

proposed system achieved the best performance. An interesting fact is that even without these gold properties, our System 3 in Table 6 was already better than CAEVO (on Line 1) and Ning et al. (2017) (on Line 2) in both metrics. This is appealing because in practice, those gold properties may not exist, but our proposed system can still generate state-of-the-art performance without them.

No.	System	P	R	F_1	F_{aware}
Partial TBDense: Focus of this work.*					
1	CAEVO	52.3	43.7	47.6	46.7
2	Ning et al. (2017)	47.4	56.3	51.5	49.1
3	Proposed	50.0	62.4	55.5	52.8
Complete TBDense: Naive augmentation.					
4	CAEVO	51.8	32.6	40.0	45.7
5	Ning et al. (2017)	46.2	40.6	43.2	48.5
6	Proposed**	47.2	42.4	44.7	49.2

*Note that TEMPROB is only available for events extracted by SRL (See Sec. 3.2.2 for details).
**Augment the output of Line 3 with predictions from Ning et al. (2017).

Table 8: **Comparison of the proposed TempRel extraction method with two best-so-far systems using two metrics.** Since TEMPROB is only on SRL verb events, *Partial* TBDense is the focus of our work, where we can see significant improvement brought by simply using the prior knowledge from TEMPROB. Per the McNemar's test, Line 3 is better than Line 2 with $p<0.0005$. For interested readers, we also naively augmented the proposed method to the *complete* TBDense and show state-of-the-art performance on it.

For readers who are interested in the complete TBDense dataset, we also performed a naive augmentation as follows. Recall that System 3 only makes predictions to a subset of the complete TBDense dataset. We kept this subset of predictions, and filled the missing predictions by Ning et al. (2017). Performances of this naively augmented proposed system is compared with CAEVO and Ning et al. (2017) on the *complete* TBDense dataset. We can see that by replacing with predictions from our proposed system, Ning et al. (2017) got a better precision, recall, F_1, and awareness F_1, which is the new state-of-the-art on all reported performances on this dataset. Note that the awareness F_1 scores on Lines 4-5 are consistent with reported values in Ning et al. (2017). To our knowledge, the results in Table 8 is the first in literature that reports performances in both metrics, and it is promising to see that the proposed method outperformed state-of-the-art methods in both metrics.

5 Conclusion

Temporal relation (TempRel) extraction is an important and challenging task in NLP, partly due to its strong dependence on prior knowledge. Motivated by practical examples, this paper argues that a resource of the temporal order that events *usually* follow is helpful. To construct such a resource, we automatically processed a large corpus from NYT with more than 1 million documents using an existing TempRel extraction system and obtained the *TEMporal relation PRObabilistic knowledge Base* (TEMPROB). The TEMPROB is a good showcase of the capability of such prior knowledge, and it has shown its power in improving existing TempRel extraction systems on a benchmark dataset, TBDense. The resource and the system reported in this paper are both publicly available[12] and we hope that it can foster more investigations into time-related tasks.

Acknowledgements

We thank all the reviewers for providing useful comments. This research is supported in part by a grant from the Allen Institute for Artificial Intelligence (allenai.org); the IBM-ILLINOIS Center for Cognitive Computing Systems Research (C3SR) - a research collaboration as part of the IBM AI Horizons Network; by DARPA under agreement number FA8750-13-2-0008; and by the Army Research Laboratory (ARL) under agreement W911NF-09-2-0053.

The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government. Any opinions, findings, conclusions or recommendations are those of the authors and do not necessarily reflect the view of the ARL.

References

Steven Bethard, Leon Derczynski, Guergana Savova, James Pustejovsky, and Marc Verhagen. 2015. SemEval-2015 Task 6: Clinical TempEval. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Association for

[12] http://cogcomp.org/page/publication_view/830

Computational Linguistics, Denver, Colorado, pages 806–814.

Steven Bethard, James H Martin, and Sara Klingenstein. 2007. Timelines from text: Identification of syntactic temporal relations. In *IEEE International Conference on Semantic Computing (ICSC)*. pages 11–18.

Steven Bethard, Guergana Savova, Wei-Te Chen, Leon Derczynski, James Pustejovsky, and Marc Verhagen. 2016. SemEval-2016 Task 12: Clinical TempEval. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. Association for Computational Linguistics, San Diego, California, pages 1052–1062.

Steven Bethard, Guergana Savova, Martha Palmer, and James Pustejovsky. 2017. SemEval-2017 Task 12: Clinical TempEval. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics, pages 565–572.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, pages 1247–1250.

P. Bramsen, P. Deshpande, Y. K. Lee, and R. Barzilay. 2006. Inducing temporal graphs. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*. pages 189–198.

Taylor Cassidy, Bill McDowell, Nathanel Chambers, and Steven Bethard. 2014. An annotation framework for dense event ordering. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. pages 501–506.

N. Chambers and D. Jurafsky. 2008. Jointly combining implicit constraints improves temporal ordering. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard. 2014. Dense event ordering with a multi-pass architecture. *Transactions of the Association for Computational Linguistics* 2:273–284.

Nathanael Chambers, Shan Wang, and Dan Jurafsky. 2007. Classifying temporal relations between events. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*. pages 173–176.

T. Chklovski and P. Pantel. 2004. VerbOcean: Mining the Web for Fine-Grained Semantic Verb Relations. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*. pages 33–40.

J. Clarke, V. Srikumar, M. Sammons, and D. Roth. 2012. An nlp curator (or: How i learned to stop worrying and love nlp pipelines). In *Proc. of the International Conference on Language Resources and Evaluation (LREC)*.

Pascal Denis and Philippe Muller. 2011. Predicting globally-coherent temporal structures from texts via endpoint inference and graph decomposition. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. volume 22, page 1788.

Q. Do, Y. Chan, and D. Roth. 2011. Minimally supervised event causality identification. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Edinburgh, Scotland.

Q. Do, W. Lu, and D. Roth. 2012. Joint inference for event timeline construction. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

C. Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Y. Freund and R. Schapire. 1998. Large margin classification using the Perceptron algorithm. In *Proceedings of the Annual ACM Workshop on Computational Learning Theory (COLT)*. pages 209–217.

David Graff. 2002. The AQUAINT corpus of english news text. *Linguistic Data Consortium, Philadelphia* .

Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. Yago2: A spatially and temporally enhanced knowledge base from wikipedia (extended abstract). In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Eduard Hovy, Teruko Mitamura, Felisa Verdejo, Jun Araki, and Andrew Philpot. 2013. Events are not simple: Identity, non-identity, and quasi-identity. In *Workshop on Events*.

Tingsong Jiang, Tianyu Liu, Tao Ge, Lei Sha, Baobao Chang, Sujian Li, and Zhifang Sui. 2016. Towards time-aware knowledge graph completion. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 1715–1724.

Hector Llorens, Nathanael Chambers, Naushad UzZaman, Nasrin Mostafazadeh, James Allen, and James Pustejovsky. 2015. SemEval-2015 Task 5: QA TEMPEVAL - evaluating temporal information understanding with question answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. pages 792–800.

Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. 2006.

Machine learning of temporal relations. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. pages 753–760.

Anne-Lyse Minard, Manuela Speranza, Eneko Agirre, Itziar Aldabe, Marieke van Erp, Bernardo Magnini, German Rigau, Ruben Urizar, and Fondazione Bruno Kessler. 2015. SemEval-2015 Task 4: TimeLine: Cross-document event ordering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. pages 778–786.

Paramita Mirza and Sara Tonelli. 2016. CATENA: CAusal and TEmporal relation extraction from NAtural language texts. In *The 26th International Conference on Computational Linguistics*. pages 64–75.

T. Mitamura, Y. Yamakawa, S. Holm, Z. Song, A. Bies, S. Kulick, and S. Strassel. 2015. Event nugget annotation: Processes and issues. In *Proceedings of the Workshop on Events at NAACL-HLT*.

Qiang Ning, Zhili Feng, and Dan Roth. 2017. A structured learning approach to temporal relation extraction. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*. Copenhagen, Denmark.

Haoruo Peng and Dan Roth. 2016. Two discourse driven language models for semantics. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Haoruo Peng, Yangqiu Song, and Dan Roth. 2016. Event detection and co-reference with minimal supervision. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, et al. 2003. The TIMEBANK corpus. In *Corpus linguistics*. volume 2003, page 40.

N. Rizzolo and D. Roth. 2010. Learning based java for rapid development of nlp systems. In *Proc. of the International Conference on Language Resources and Evaluation (LREC)*. Valletta, Malta.

D. Roth and W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In Hwee Tou Ng and Ellen Riloff, editors, *Proc. of the Conference on Computational Natural Language Learning (CoNLL)*. pages 1–8.

Zhiyi Song, Ann Bies, Stephanie Strassel, Tom Riese, Justin Mott, Joe Ellis, Jonathan Wright, Seth Kulick, Neville Ryant, and Xiaoyi Ma. 2015. From light to rich ere: Annotation of entities, relations, and events. In *Proceedings of the The 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation*. Association for Computational Linguistics, Denver, Colorado, pages 89–98.

Evangelia Spiliopoulou, Eduard Hovy, and Teruko Mitamura. 2017. Event detection using frame-semantic parser. In *Proceedings of the Events and Stories in the News Workshop*. Association for Computational Linguistics, Vancouver, Canada, pages 15–20.

Naushad UzZaman and James F Allen. 2011. Temporal evaluation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. pages 351–356.

Naushad UzZaman, Hector Llorens, James Allen, Leon Derczynski, Marc Verhagen, and James Pustejovsky. 2013. SemEval-2013 Task 1: TempEval-3: Evaluating time expressions, events, and temporal relations. In *Second Joint Conference on Lexical and Computational Semantics*. volume 2, pages 1–9.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Graham Katz, and James Pustejovsky. 2007. SemEval-2007 Task 15: TempEval temporal relation identification. In *SemEval*. pages 75–80.

Marc Verhagen and James Pustejovsky. 2008. Temporal processing with the TARSQI toolkit. In *22nd International Conference on on Computational Linguistics: Demonstration Papers*. pages 189–192.

Marc Verhagen, Roser Sauri, Tommaso Caselli, and James Pustejovsky. 2010. SemEval-2010 Task 13: TempEval-2. In *SemEval*. pages 57–62.

Multimodal Named Entity Recognition for Short Social Media Posts

Seungwhan Moon[1,2], **Leonardo Neves**[2], **Vitor Carvalho**[3]
[1] Language Technologies Institute, Carnegie Mellon University
[2] Snap Research
[3] Intuit
seungwhm@cs.cmu.edu, lneves@snap.com, vitor_carvalho@intuit.com

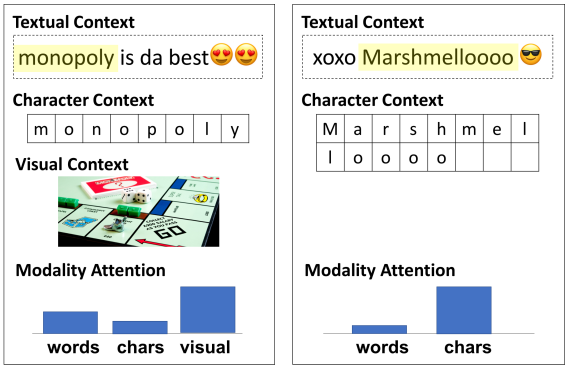
Abstract

We introduce a new task called Multimodal Named Entity Recognition (MNER) for noisy user-generated data such as tweets or Snapchat captions, which comprise short text with accompanying images. These social media posts often come in inconsistent or incomplete syntax and lexical notations with very limited surrounding textual contexts, bringing significant challenges for NER. To this end, we create a new dataset for MNER called SnapCaptions (Snapchat image-caption pairs submitted to public and crowd-sourced stories with fully annotated named entities). We then build upon the state-of-the-art Bi-LSTM word/character based NER models with 1) a deep image network which incorporates relevant visual context to augment textual information, and 2) a generic *modality-attention* module which learns to attenuate irrelevant modalities while amplifying the most informative ones to extract contexts from, adaptive to each sample and token. The proposed MNER model with modality attention significantly outperforms the state-of-the-art text-only NER models by successfully leveraging provided visual contexts, opening up potential applications of MNER on myriads of social media platforms.

1 Introduction

Social media with abundant user-generated posts provide a rich platform for understanding events, opinions and preferences of groups and individuals. These insights are primarily hidden in unstructured forms of social media posts, such as in free-form text or images without tags. Named entity recognition (NER), the task of recognizing named entities from free-form text, is thus a critical step for building structural information, allowing for its use in personalized assistance, recommendations, advertisement, etc.

While many previous approaches (Lample et al., 2016; Ma and Hovy, 2016; Chiu and Nichols, 2015; Huang et al., 2015; Lafferty et al., 2001) on NER have shown success for well-formed text in recognizing named entities via word context resolution (*e.g.* LSTM with word embeddings) combined with character-level features (*e.g.* CharLSTM/CNN), several additional challenges remain for recognizing named entities from extremely short and coarse text found in social media posts. For instance, short social media posts often do not provide enough textual contexts to resolve polysemous entities (*e.g.* "monopoly is da best 😍", where 'monopoly' may refer to a board game (named entity) or a term in economics). In addition, noisy text includes a huge number of unknown tokens due to inconsistent lexical notations and frequent mentions of various newly trending entities (*e.g.* "xoxo Marshmellooo 😎", where 'Marshmellooo' is a mis-spelling of a known entity 'Marshmello', a



(a) (b)

Figure 1: **Multimodal NER + modality attention**. (a) Visual contexts help recognizing polysemous entity names ('Monopoly' as in a board game versus an economics term). (b) Modality attention successfully suppresses word embeddings of a unknown token ('Marshmellooo' with erroneously trailing 'o's), and focuses on character-based context (*e.g.* capitalized first letter, and lexical similarity to a known named entity ('Marshmello', a music producer)) for correct prediction.

Proceedings of NAACL-HLT 2018, pages 852–860
New Orleans, Louisiana, June 1 - 6, 2018.

music producer), making word embeddings based neural networks NER models vulnerable.

To address the challenges above for social media posts, we build upon the state-of-the-art neural architecture for NER with the following two novel approaches (Figure 1). First, we propose to leverage auxiliary modalities for additional context resolution of entities. For example, many popular social media platforms now provide ways to compose a post in multiple modalities - specifically image and text (*e.g. Snapchat* captions, *Twitter* posts with image URLs), from which we can obtain additional context for understanding posts. While "monopoly" in the previous example is ambiguous in its textual form, an accompanying snap image of a board game can help disambiguate among polysemous entities, thereby correctly recognizing it as a named entity.

Second, we also propose a general modality attention module which chooses per decoding step the most informative modality among available ones (in our case, word embeddings, character embeddings, or visual features) to extract context from. For example, the modality attention module lets the decoder attenuate the word-level signals for unknown word tokens (*e.g.* "Marshmellooooo" with trailing 'o's) and amplifies character-level features intsead (*e.g.* capitalized first letter, lexical similarity to other known named entity token 'Marshmello', etc.), thereby suppressing noise information ("UNK" token embedding) in decoding steps. Note that most of the previous literature in NER or other NLP tasks combine word and character-level information with naive concatenation, which is vulnerable to noisy social media posts. When an auxiliary image is available, the modality attention module determines to amplify this visual context *e.g.* in disambiguating polysemous entities, or to attenuate visual contexts when they are irrelevant to target named entities, *e.g.* selfies, etc. Note that the proposed modality attention module is distinct from how attention is used in other sequence-to-sequence literature (*e.g.* attending to a specific token within an input sequence). Section 2 provides the detailed literature review.

Our contributions are three-fold: we propose (1) an LSTM-CNN hybrid multimodal NER network that takes as input both image and text for recognition of a named entity in text input. To the best of our knowledge, our approach is the first work to incorporate visual contexts for named entity recognition tasks. (2) We propose a general *modality attention* module that selectively chooses modalities to extract primary context from, maximizing information gain and suppressing irrelevant contexts from each modality (we treat words, characters, and images as separate modalities). (3) We show that the proposed approaches outperform the state-of-the-art NER models (**both** with and without using additional visual contexts) on our new MNER dataset *SnapCaptions*, a large collection of informal and extremely short social media posts paired with unique images.

2 Related Work

Neural models for NER have been recently proposed, producing state-of-the-art performance on standard NER tasks. For example, some of the end-to-end NER systems (Passos et al., 2014; Chiu and Nichols, 2015; Huang et al., 2015; Lample et al., 2016; Ma and Hovy, 2016) use a recurrent neural network usually with a CRF (Lafferty et al., 2001; McCallum and Li, 2003) for sequence labeling, accompanied with feature extractors for words and characters (CNN, LSTMs, etc.), and achieve the state-of-the-art performance mostly without any use of gazetteers information. Note that most of these work aggregate textual contexts via concatenation of word embeddings and character embeddings. Recently, several work have addressed the NER task specifically on noisy short text segments such as Tweets, etc. (Baldwin et al., 2015; Aguilar et al., 2017). They report performance gains from leveraging external sources of information such as lexical information (*e.g.* POS tags, etc.) and/or from several preprocessing steps (*e.g.* token substitution, etc.). Our model builds upon these state-of-the-art neural models for NER tasks, and improves the model in two critical ways: (1) incorporation of visual contexts to provide auxiliary information for short media posts, and (2) addition of the modality attention module, which better incorporates word embeddings and character embeddings, especially when there are many missing tokens in the given word embedding matrix. Note that we do not explore the use of gazetteers information or other auxiliary information (POS tags, etc.) (Ratinov and Roth, 2009) as it is not the focus of our study.

Attention modules are widely applied in several deep learning tasks (Xu et al., 2015; Chan

et al., 2015; Sukhbaatar et al., 2015; Yao et al., 2015). For example, they use an attention module to attend to a subset within a single input (a part/region of an image, a specific token in an input sequence of tokens, etc.) at each decoding step in an encoder-decoder framework for image captioning tasks, etc. (Rei et al., 2016) explore various attention mechanisms in NLP tasks, but do not incorporate visual components or investigate the impact of such models on noisy social media data. (Moon and Carbonell, 2017) propose to use attention for a subset of discrete source samples in transfer learning settings. Our modality attention differs from the previous approaches in that we attenuate or amplifies each modality input as a whole among multiple available modalities, and that we use the attention mechanism essentially to map heterogeneous modalities in a single joint embedding space. Our approach also allows for reuse of the same model for predicting labels even when some of the modalities are missing in input, as other modalities would still preserve the same semantics in the embeddings space.

Multimodal learning is studied in various domains and applications, aimed at building a joint model that extracts contextual information from multiple modalities (views) of parallel datasets.

The most relevant task to our multimodal NER system is the task of multimodal machine translation (Elliott et al., 2015; Specia et al., 2016), which aims at building a better machine translation system by taking as input a sentence in a source language as well as a corresponding image. Several standard sequence-to-sequence architectures are explored (*e.g.* a target-language LSTM decoder that takes as input an image first).

Other previous literature include study of Canonical Correlation Analysis (CCA) (Dhillon et al., 2011) to learn feature correlations among multiple modalities, which is widely used in many applications. Other applications include image captioning (Xu et al., 2015), audio-visual recognition (Moon et al., 2015), visual question answering systems (Antol et al., 2015), etc.

To the best of our knowledge, our approach is the first work to incorporate visual contexts for named entity recognition tasks.

3 Proposed Methods

Figure 2 illustrates the proposed multimodal NER (MNER) model. First, we obtain word embeddings, character embeddings, and visual features (Section 3.1). A Bi-LSTM-CRF model then takes as input a sequence of tokens, each of which comprises a word token, a character sequence, and an image, in their respective representation (Section 3.2). At each decoding step, representations from each modality are combined via the modality attention module to produce an entity label for each token (3.3). We formulate each component of the model in the following subsections.

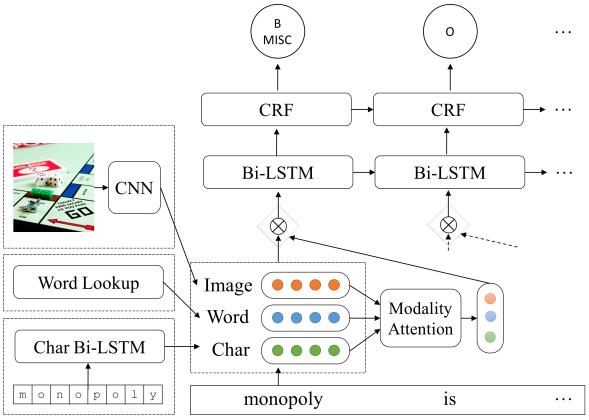


Figure 2: The main architecture for our multimodal NER (MNER) network with modality attention. At each decoding step, word embeddings, character embeddings, and visual features are merged with modality attention. Bi-LSTM/CRF takes as input each token and produces an entity label.

Notations: Let $\mathbf{x} = \{\mathbf{x}_t\}_{t=1}^{T}$ a sequence of input tokens with length T, with a corresponding label sequence $\mathbf{y} = \{\mathbf{y}_t\}_{t=1}^{T}$ indicating named entities (*e.g.* in standard BIO formats). Each input token is composed of three modalities: $\mathbf{x}_t = \{\mathbf{x}_t^{(w)}, \mathbf{x}_t^{(c)}, \mathbf{x}_t^{(v)}\}$ for word embeddings, character embeddings, and visual embeddings representations, respectively.

3.1 Features

Similar to the state-of-the-art NER approaches (Lample et al., 2016; Ma and Hovy, 2016; Aguilar et al., 2017; Passos et al., 2014; Chiu and Nichols, 2015; Huang et al., 2015), we use both word embeddings and character embeddings.

Word embeddings are obtained from an unsupervised learning model that learns co-occurrence statistics of words from a large external corpus, yielding word embeddings as distributional semantics (Mikolov et al., 2013). Specifically, we use pre-trained embeddings from GloVE (Pennington et al., 2014).

Character embeddings are obtained from a Bi-LSTM which takes as input a sequence of characters of each token, similarly to (Lample et al., 2016). An alternative approach for obtaining character embeddings is using a convolutional neural network as in (Ma and Hovy, 2016), but we find that Bi-LSTM representation of characters yields empirically better results in our experiments.

Visual embeddings: To extract features from an image, we take the final hidden layer representation of a modified version of the convolutional network model called Inception (GoogLeNet) (Szegedy et al., 2014, 2015) trained on the ImageNet dataset (Russakovsky et al., 2015) to classify multiple objects in the scene. Our implementation of the Inception model has deep 22 layers, training of which is made possible via "network in network" principles and several dimension reduction techniques to improve computing resource utilization. The final layer representation encodes discriminative information describing what objects are shown in an image, which provide auxiliary contexts for understanding textual tokens and entities in accompanying captions.

Incorporating this visual information onto the traditional NER system is an open challenge, and multiple approaches can be considered. For instance, one may provide visual contexts only as an initial input to decoder as in some encoder-decoder image captioning systems (Vinyals et al., 2015). However, we empirically observe that an NER decoder which takes as input the visual embeddings at every decoding step (Section 3.2), combined with the modality attention module (Section 3.3), yields better results.

Lastly, we add a transform layer for each feature *e.g.* $\mathbf{x}_t^{(w)}, \mathbf{x}_t^{(c)}, \mathbf{x}_t^{(v)} := \sigma_w(\mathbf{x}_t^{(w)}), \sigma_c(\mathbf{x}_t^{(c)}), \sigma_v(\mathbf{x}_t^{(v)})$ before it is fed to the NER entity LSTM.

3.2 Bi-LSTM + CRF for Multimodal NER

Our MNER model is built on a Bi-LSTM and CRF hybrid model. We use the following implementation for the entity Bi-LSTM.

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_{xi}\mathbf{h}_{t-1} + \mathbf{W}_{ci}\mathbf{c}_{t-1}) \\
\mathbf{c}_t &= (1 - \mathbf{i}_t) \odot \mathbf{c}_{t-1} \\
&\quad + \mathbf{i}_t \odot \tanh(\mathbf{W}_{xc}\overline{\mathbf{x}_t} + \mathbf{W}_{hc}\mathbf{h}_{t-1}) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_{xo}\overline{\mathbf{x}_t} + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{W}_{co}\mathbf{c}_t) \\
\mathbf{h}_t &= \text{LSTM}(\overline{\mathbf{x}_t}) \qquad (1) \\
&= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned}$$

where $\overline{\mathbf{x}_t}$ is a weighted average of three modalities $\mathbf{x}_t = \{\mathbf{x}_t^{(w)}; \mathbf{x}_t^{(c)}; \mathbf{x}_t^{(v)}\}$ via the modality attention module, which will be defined in Section 3.3. Bias terms for gates are omitted here for simplicity of notation.

We then obtain bi-directional entity token representations $\overleftrightarrow{\mathbf{h}_t} = [\overrightarrow{\mathbf{h}_t}; \overleftarrow{\mathbf{h}_t}]$ by concatenating its left and right context representations. To enforce structural correlations between labels in sequence decoding, $\overleftrightarrow{\mathbf{h}_t}$ is then passed to a conditional random field (CRF) to produce a label for each token maximizing the following objective.

$$\mathbf{y}^* = \underset{\mathbf{y}}{\text{argmax}}\, p(\mathbf{y}|\overleftrightarrow{\mathbf{h}}; \mathbf{W}_{\text{CRF}}) \qquad (2)$$

$$p(\mathbf{y}|\overleftrightarrow{\mathbf{h}}; \mathbf{W}_{\text{CRF}}) = \frac{\prod_t \psi_t(\mathbf{y}_{t-1}, \mathbf{y}_t; \overleftrightarrow{\mathbf{h}})}{\sum_{\mathbf{y}'} \prod_t \psi_t(\mathbf{y'}_{t-1}, \mathbf{y'}_t; \overleftrightarrow{\mathbf{h}})}$$

where $\psi_t(\mathbf{y}', \mathbf{y}'; \overleftrightarrow{\mathbf{h}})$ is a potential function, $\mathbf{W}_{\text{CRF}}$ is a set of parameters that defines the potential functions and weight vectors for label pairs $(\mathbf{y}', \mathbf{y}')$. Bias terms are omitted for brevity of formulation.

The model can be trained via log-likelihood maximization for the training set $\{(\mathbf{x}_i, \mathbf{y}_i)\}$:

$$\mathcal{L}(\mathbf{W}_{\text{CRF}}) = \sum_i \log p(\mathbf{y}|\overleftrightarrow{\mathbf{h}}; \mathbf{W}) \qquad (3)$$

3.3 Modality Attention

The modality attention module learns a unified representation space for multiple available modalities (*e.g.* words, characters, images, etc.), and produces a single vector representation with aggregated knowledge among multiple modalities, based on their weighted importance. We motivate this module from the following observations.

A majority of the previous literature combine the word and character-level contexts by simply concatenating the word and character embeddings at each decoding step, *e.g.* $\mathbf{h}_t = \text{LSTM}([\mathbf{x}_t^{(w)}; \mathbf{x}_t^{(c)}])$ in Eq.1. However, this naive concatenation of two modalities (word and characters) results in inaccurate decoding, specifically for unknown word token embeddings (*e.g.* an all-zero vector $\mathbf{x}_t^{(w)} = \mathbf{0}$ or a random vector $\mathbf{x}_t^{(w)} = \boldsymbol{\epsilon} \sim U(-\sigma, +\sigma)$ is assigned for any unknown token $\mathbf{x}_t$, thus $\mathbf{h}_t = \text{LSTM}([\mathbf{0}; \mathbf{x}_t^{(c)}])$ or $\text{LSTM}([\epsilon; \mathbf{x}_t^{(c)}])$). While this concatenation approach does not cause significant errors for well-formatted text, we observe that it induces performance degradation for our social media post

datasets which contain a significant number of missing tokens.

Similarly, naive merging of textual and visual information (*e.g.* $\mathbf{h}_t = \text{LSTM}([\mathbf{x}_t^{(w)}; \mathbf{x}_t^{(c)}; \mathbf{x}_t^{(v)}])$) yields suboptimal results as each modality is treated equally informative, whereas in our datasets some of the images may contain irrelevant contexts to textual modalities. Hence, ideally there needs a mechanism in which the model can effectively turn the *switch* on and off the modalities adaptive to each sample.

To this end, we propose a general modality attention module, which adaptively attenuates or emphasizes each modality as a whole at each decoding step t, and produces a soft-attended context vector $\overline{\mathbf{x}_t}$ as an input token for the entity LSTM.

$$[\mathbf{a}_t^{(w)}, \mathbf{a}_t^{(c)}, \mathbf{a}_t^{(v)}] = \sigma\big(\mathbf{W}_m \cdot [\mathbf{x}_t^{(w)}; \mathbf{x}_t^{(c)}; \mathbf{x}_t^{(v)}] + \mathbf{b}_m\big)$$

$$\alpha_t^{(m)} = \frac{\exp(\mathbf{a}_t^{(m)})}{\sum_{m' \in \{w,c,v\}} \exp(\mathbf{a}_t^{(m')})} \quad \forall m \in \{w, c, v\}$$

$$\overline{\mathbf{x}_t} = \sum_{m \in \{w,c,v\}} \alpha_t^{(m)} \mathbf{x}_t^{(m)} \tag{4}$$

where $\alpha_t = [\alpha_t^{(w)}; \alpha_t^{(c)}; \alpha_t^{(v)}] \in \mathbb{R}^3$ is an attention vector at each decoding step t, and $\overline{\mathbf{x}_t}$ is a final context vector at t that maximizes information gain for $\mathbf{x}_t$. Note that the optimization of the objective function (Eq.1) with modality attention (Eq.4) requires each modality to have the same dimension (*e.g.* $\mathbf{x}_t^{(w)}, \mathbf{x}_t^{(c)}, \mathbf{x}_t^{(v)} \in \mathbb{R}^p$), and that the transformation via $\mathbf{W}_m$ essentially enforces each modality to be mapped into the same unified subspace, where the weighted average of which encodes discrimitive features for recognition of named entities.

When visual context is not provided with each token (as in the traditional NER task), we can define the modality attention for word and character embeddings only in a similar way:

$$[\mathbf{a}_t^{(w)}, \mathbf{a}_t^{(c)}] = \sigma\big(\mathbf{W}_m \cdot [\mathbf{x}_t^{(w)}; \mathbf{x}_t^{(c)}] + \mathbf{b}_m\big) \tag{5}$$

$$\alpha_t^{(m)} = \frac{\exp(\mathbf{a}_t^{(m)})}{\sum_{m' \in \{w,c\}} \exp(\mathbf{a}_t^{(m')})} \quad \forall m \in \{w, c\}$$

$$\overline{\mathbf{x}_t} = \sum_{m \in \{w,c\}} \alpha_t^{(m)} \mathbf{x}_t^{(m)}$$

Note that while we apply this modality attention module to the Bi-LSTM+CRF architecture (Section 3.2) for its empirical superiority, the module itself is flexible and thus can work with other NER architectures or for other multimodal applications.

4 Empirical Evaluation

4.1 SnapCaptions Dataset

The **SnapCaptions** dataset is composed of 10K user-generated image (snap) and textual caption pairs where named entities in captions are manually labeled by expert human annotators (entity types: PER, LOC, ORG, MISC). These captions are collected exclusively from snaps submitted to public and crowd-sourced stories (aka Snapchat *Live Stories* or *Our Stories*). Examples of such public crowd-sourced stories are "New York Story" or "Thanksgiving Story", which comprise snaps that are aggregated for various public events, venues, etc. All snaps were posted between year 2016 and 2017, and do not contain raw images or other associated information (only textual captions and obfuscated visual descriptor features extracted from the pre-trained Inception-Net are available). We split the dataset into train (70%), validation (15%), and test sets (15%). The captions data have average length of 30.7 characters (5.81 words) with vocabulary size 15,733, where 6,612 are considered unknown tokens from Stanford GloVE embeddings (Pennington et al., 2014). Named entities annotated in the SnapCaptions dataset include many of new and emerging entities, and they are found in various surface forms (various nicknames, typos, etc.) To the best of our knowledge, *SnapCaptions* is the only dataset that contains natural image-caption pairs with expert-annotated named entities.

4.2 Baselines

Task: given a caption and a paired image (if used), the goal is to label every token in a caption in BIO scheme (B: beginning, I: inside, O: outside) (Sang and Veenstra, 1999). We report the performance of the following state-of-the-art NER models as baselines, as well as several configurations of our proposed approach to examine contributions of each component (W: word, C: char, V: visual).

- Bi-LSTM/CRF (W only): only takes word token embeddings (Stanford GloVE) as input. The rest of the architecture is kept the same.
- Bi-LSTM/CRF + Bi-CharLSTM (C only): only takes a character sequence of each word token as input. (No word embeddings)

Modalities	Model	4 Entity Types (%)			Segmentation (%)		
		Prec.	Recall	F1	Prec.	Recall	F1
C	Bi-LSTM/CRF + Bi-CharLSTM	5.0	28.1	8.5	68.6	10.8	18.6
W	Bi-LSTM/CRF	38.2	53.3	44.6	82.5	50.1	62.4
W + C	(Aguilar et al., 2017)	45.9	48.9	47.4	74.0	61.7	67.3
W + C	(Ma and Hovy, 2016)	46.0	51.9	48.7	76.8	61.0	68.0
W + C	(Lample et al., 2016)	47.7	49.9	48.8	74.4	63.3	68.4
W + C	Bi-LSTM/CRF + Bi-CharLSTM w/ Modality Attention	49.4	51.7	50.5	75.7	63.3	68.9
W + C + V	Bi-LSTM/CRF + Bi-CharLSTM + Inception	**50.5**	52.3	51.4	71.9	**66.5**	**69.1**
W + C + V	Bi-LSTM/CRF + Bi-CharLSTM + Inception w/ Modality Attention	48.7	**58.7**	**52.4**	**77.4**	60.6	68.0

Table 1: NER performance on the *SnapCaptions* dataset with varying modalities (W: word, C: char, V: visual). We report precision, recall, and F1 score for both entity types recognition (PER, LOC, ORG, MISC) and entity segmentation (untyped recognition - named entity or not) tasks.

- Bi-LSTM/CRF + Bi-CharLSTM (W+C) (Lample et al., 2016): takes as input both word embeddings and character embeddings extracted from a Bi-CharLSTM. Entity LSTM takes concatenated vectors of word and character embeddings as input tokens.
- Bi-LSTM/CRF + CharCNN (W+C) (Ma and Hovy, 2016): uses character embeddings extracted from a CNN instead.
- Bi-LSTM/CRF + CharCNN (W+C) + Multi-task (Aguilar et al., 2017): trains the model to perform both recognition (into multiple entity types) as well as segmentation (binary) tasks.
- (**proposed**) Bi-LSTM/CRF + Bi-CharLSTM with modality attention (W+C): uses the modality attention to merge word and character embeddings.
- (**proposed**) Bi-LSTM/CRF + Bi-CharLSTM + Inception (W+C+V): takes as input visual contexts extracted from InceptionNet as well, concatenated with word and char vectors.
- (**proposed**) Bi-LSTM/CRF + Bi-CharLSTM + Inception with modality attention (W+C+V): uses the modality attention to merge word, character, and visual embeddings as input to entity LSTM.

4.3 Results: SnapCaptions Dataset

Table 1 shows the NER performance on the *Snap Captions* dataset. We report both entity types recognition (PER, LOC, ORG, MISC) and named entity segmentation (named entity or not) results.

Parameters: We tune the parameters of each model with the following search space (bold indicate the choice for our final model): character embeddings dimension: {25, 50, 100, **150**, 200, 300}, word embeddings size: {25, 50, 100, **150**, 200, 300}, LSTM hidden states: {25, 50, **100**, 150, 200, 300}, and $\overline{x}$ dimension: {25, 50, 100, **150**, 200, 300}. We optimize the parameters with Adagrad (Duchi et al., 2011) with batch size 10, learning rate 0.02, epsilon 10^{-8}, and decay 0.0.

Main Results: When visual context is available (W+C+V), we see that the model performance greatly improves over the textual models (W+C), showing that visual contexts are complimentary to textual information in named entity recognition tasks. In addition, it can be seen that the modality attention module further improves the entity type recognition performance for (W+C+V). This result indicates that the modality attention is able to focus on the most effective modality (visual, words, or characters) adaptive to each sample to maximize information gain. Note that our text-only model (W+C) with the modality attention module also significantly outperform the state-of-the-art baselines (Aguilar et al., 2017; Ma and Hovy, 2016; Lample et al., 2016) that use the same textual modalities (W+C), showing the effectiveness of the modality attention module for textual models as well.

Error Analysis: Table 2 shows example cases where incorporation of visual contexts affects prediction of named entities. For example, the token 'curry' in the caption *"The curry's* 🏆*"* is polysemous and may refer to either a type of food or a famous basketball player 'Stephen Curry', and the surrounding textual contexts do not provide

	Caption (target)	Visual Tags	GT	Prediction	
				(W+C+V)	(W+C)
+	*"The curry's 🏆"*	parade, marching, urban area, ...	B-PER	B-PER	O
	"Grandma w dat lit Apple Crisp"	funnel cake, melting, frozen, ...	O	O	B-ORG
	"Okay duke dumont 😍"	DJ, guitarist, circus, ...	B,I-PER	B,I-PER	O,O
	"CSI with my hubby"	TV, movie, television, ...	B-MISC	B-MISC	B-ORG
	"Twin day at angel stadium"	stadium, arena, stampede, ...	B,I-LOC	B,I-LOC	O,O
	"LETS GO CID"	drum, DJ, drummer, ...	B-PER	B-PER	O
	"MARSHMELLOOOOOOOOS"	DJ, night, martini, ...	B-PER	B-PER	O
-	*"Y'all come see me at bojangles. 😊"*	floor, tile, airport terminal, ...	B-ORG	O	B-ORG
	"If u're not watching this season of bachelorette ur doing LIFE WRONG"	monitor, suite, cubicle, ...	B-MISC	O	B-MISC

Table 2: Error analysis: **when do images help NER**? Ground-truth labels (GT) and predictions of our model with vision input (W+C+V) and the one without (W+C) for the underlined named entities (or false positives) are shown. For interpretability, visual tags (label output of InceptionNet) are presented instead of actual feature vectors used.

enough information to disambiguate it. On the other hand, visual contexts (visual tags: 'parade', 'urban area', ...) provide similarities to the token's distributional semantics from other training examples (*e.g.* snaps from "NBA Championship Parade Story"), and thus the model successfully predicts the token as a named entity. Similarly, while the text-only model erroneously predicts 'Apple' in the caption *"Grandma w dat lit Apple Crisp"* as an organization (*e.g.* Apple Inc.), the visual contexts (describing objects related to food) help disambiguate the token, making the model predict it correctly as a non-named entity (a fruit). Trending entities (musicians or DJs such as 'CID', 'Duke Dumont', 'Marshmello', etc.) are also recognized correctly with strengthened contexts from visual information (describing concert scenes) despite lack of surrounding textual contexts. A few cases where visual contexts harmed the performance mostly include visual tags that are unrelated to a token or its surrounding textual contexts.

Visualization of Modality Attention: Figure 3 visualizes the modality attention module at each decoding step (each column), where amplified modality is represented with darker color, and attenuated modality is represented with lighter color.

For the image-aided model (W+C+V; upper row in Figure 3), we confirm that the modality attention successfully attenuates irrelevant signals (*e.g.* selfies, etc.) and amplifies relevant modality-based contexts in prediction of a given token. In the example of "*disney word essential = coffee*" with visual tags *selfie, phone, person*, the modality attention successfully attenuates distracting visual signals and focuses on textual modalities, consequently making correct predictions. The named entities in the examples of "*Beautiful night atop The Space Needle*" and "*Splash Mountain*" are challenging to predict because they are composed of common nouns (space, needle, splash, mountain), and thus they often need additional contexts to correctly predict. In the training data, visual contexts make stronger indicators for these named entities (space needle, splash mountain), and the modality attention module successfully attends more to stronger signals.

For text-only model (W+C), we observe that performance gains mostly come from the modality attention module better handling tokens unseen during training or unknown tokens from the pre-trained word embeddings matrix. For example, while *WaRriOoOrs* and *Kooler Matic* are missing tokens in the word embeddings matrix, it successfully amplifies character-based contexts (*e.g.* capitalized first letters, similarity to known entities 'Golden State Warriors') and suppresses word-based contexts (word embeddings for unknown tokens *e.g.* 'WaRriOoOrs'), leading to correct predictions. This result is significant because it shows performance of the model, with an almost identical architecture, can still improve without having to scale the word embeddings matrix indefinitely.

Figure 3 (b) shows the cases where the modality attention led to incorrect predictions. For example, the model predicts missing tokens *HUUUGE* and *Shampooer* incorrectly as named entities by amplifying misleading character-based contexts (*e.g.* capitalized first letters) or visual contexts (*e.g.*

Figure 3: **Visualization of modality attention** (a) successful cases and (b) unsuccessful ones from *SnapCaptions* test data. For each decoding step of a token (column), the modality attention module amplifies the most relevant modality (darker) while attenuating irrelevant modalities (lighter). The model makes final predictions based on weighted signals from all modalities. For interpretability, visual tags (label output of InceptionNet) are presented instead of actual feature vectors used. GT: ground-truth, Pred: prediction by our model. Modalities- W: words, C: characters, V: visual.

Vocab Size	w/o M.A.	w/ M.A.
100%	48.8	**50.5**
75%	48.7	**50.1**
50%	47.8	**49.6**
25%	46.4	**48.7**

Table 3: NER performance (F1) on SnapCaptions with **varying word embeddings vocabulary size**. Models being compared: (W+C) Bi-LSTM/CRF + Bi-CharLSTM w/ and w/o modality attention (M.A.)

concert scenes, associated contexts of which often include named entities in the training dataset).

Sensitivity to Word Embeddings Vocabulary Size: In order to isolate the effectiveness of the modality attention module on textual models in handling missing tokens, we report the performance with varying word embeddings vocabulary sizes in Table 3. By increasing the number of missing tokens artificially by randomly removing words from the word embeddings matrix (original vocab size: 400K), we observe that while the overall performance degrades, the modality attention module is able to suppress the peformance degradation. Note also that the performance gap generally gets bigger as we decrease the vocabulary size of the word embeddings matrix. This result is significant in that the modality attention is able to improve the model more robust to missing tokens without having to train an indefinitely large word embeddings matrix for arbitrarily noisy social media text datasets.

5 Conclusions

We proposed a new multimodal NER (MNER: image + text) task on short social media posts. We demonstrated for the first time an effective MNER system, where visual information is combined with textual information to outperform traditional text-based NER baselines. Our work can be applied to myriads of social media posts or other articles across multiple platforms which often include both text and accompanying images. In addition, we proposed the *modality attention* module, a new neural mechanism which learns optimal integration of different modes of correlated information. In essence, the modality attention learns to attenuate irrelevant or uninformative modal information while amplifying the primary modality to extract better overall representations. We showed that the modality attention based model outperforms other state-of-the-art baselines when text was the only modality available, by better combining word and character level information.

References

Gustavo Aguilar, Suraj Maharjan, A. Pastor Lopez-Monroy, and Thamar Solorio. 2017. A multi-task approach for named entity recognition in social media data. *ACL WNUT Workshop* .

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *ICCV*.

Timothy Baldwin, Marie-Catherine de Marneffe, Bo Han, Young-Bum Kim, Alan Ritter, and Wei Xu. 2015. Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In *Proceedings of the Workshop on Noisy User-generated Text*.

William Chan, Navdeep Jaitly, Quoc V Le, and Oriol Vinyals. 2015. Listen, attend and spell. *arXiv:1508.01211* .

Jason PC Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional lstm-cnns. *arXiv:1511.08308* .

Paramveer Dhillon, Dean P Foster, and Lyle H Ungar. 2011. Multi-view learning of word embeddings via cca. In *NIPS*. pages 199–207.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR* .

Desmond Elliott, Stella Frank, and Eva Hasler. 2015. Multi-language image description with neural sequence models. *CoRR, abs/1510.04709* .

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv:1508.01991* .

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data .

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *NAACL* .

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354* .

Andrew McCallum and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *NAACL*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *ICLR* .

Seungwhan Moon and Jaime Carbonell. 2017. Completely heterogeneous transfer learning with attention: What and what not to transfer. *IJCAI* .

Seungwhan Moon, Suyoun Kim, and Haohan Wang. 2015. Multimodal transfer deep learning with applications in audio-visual recognition. In *NIPS MMML Workshop*.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. *aarXiv:1404.5367* .

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

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *CoNLL*.

Marek Rei, Gamal KO Crichton, and Sampo Pyysalo. 2016. Attending to characters in neural sequence labeling models. *COLING* .

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *IJCV* .

Erik F Sang and Jorn Veenstra. 1999. Representing text chunks. In *EACL*.

Lucia Specia, Stella Frank, Khalil Sima'an, and Desmond Elliott. 2016. A shared task on multimodal machine translation and crosslingual image description. In *WMT*.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *NIPS*.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. 2014. Going deeper with convolutions. *CVPR* .

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2015. Rethinking the inception architecture for computer vision. *CoRR* .

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *CVPR*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. *ICML* .

Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, and Aaron Courville. 2015. Describing videos by exploiting temporal structure. In *ICCV*.

Nested Named Entity Recognition Revisited

Arzoo Katiyar and **Claire Cardie**
Department of Computer Science
Cornell University
Ithaca, NY, 14853, USA
{arzoo, cardie}@cs.cornell.edu

Abstract

We propose a novel recurrent neural network-based approach to simultaneously handle nested named entity recognition and nested entity mention detection. The model learns a hypergraph representation for nested entities using features extracted from a recurrent neural network. In evaluations on three standard data sets, we show that our approach significantly outperforms existing state-of-the-art methods, which are feature-based. The approach is also efficient: it operates linearly in the number of tokens and the number of possible output labels at any token. Finally, we present an extension of our model that jointly learns the head of each entity mention.

1 Introduction

Named entity recognition (or named entity detection) is the task of identifying text spans associated with proper names and classifying them according to their semantic class such as person, organization, etc. It is related to the task of *mention detection* (or entity mention recognition) in which text spans referring to named, nominal or prominal entities are identified and classified according to their semantic class (Florian et al., 2004). Both named entity recognition and entity mention detection are fundamental components in information extraction systems: several downstream tasks such as relation extraction (Mintz et al., 2009), coreference resolution (Chang et al., 2013) and fine-grained opinion mining (Choi et al., 2006) rely on both.

Many approaches have been successfully employed for the tasks of named entity recognition and mention detection, including linear-chain conditional random fields (Lafferty et al., 2001) and semi-Markov conditional random fields (Sarawagi and Cohen, 2005). However, most such methods suffer from an inability to handle *nested* named entities, *nested* entity mentions, or both. As a result, the downstream tasks necessarily ignore these nested entities along with any semantic relations among them. Consider, for example, the excerpts below:

(S1) Employing the [EBV - transformed [human B cell line]$_{\text{CELL\_LINE}}$]$_{\text{CELL\_LINE}}$ SKW6.4, we demonstrate . . .

(S2) . . . [the burial site of [Sheikh Abbad]$_{\text{PERSON}}$]$_{\text{LOCATION}}$ is located . . .

S1 shows a nested named entity from the GENIA dataset (Ohta et al., 2002): "human B cell line" and "EBV - transformed human B cell line" are both considered named entities of type CELL_LINE where the former is embedded inside the latter. S2, derived from the ACE corpora[1], shows a PERSON named entity ("Sheikh Abbad") nested in an entity mention of type LOCATION ("the burial site of Sheikh Abbad"). Most existing methods for named entity recognition and entity mention detection would miss the nested entity in each sentence.

Unfortunately, nested entities can be fairly common: 17% of the entities in the GENIA corpus are embedded within another entity; in the ACE corpora, 30% of sentences contain nested named entities or entity mentions, thus warranting the development of efficient models to effectively handle these linguistic phenomena.

Feature-based methods are the most common among those proposed for handling nested named entity and entity mention recognition. Alex et al.

[1] https://catalog.ldc.upenn.edu/LDC2005T09 (ACE2004) and https://catalog.ldc.upenn.edu/LDC2006T06 (ACE2005)

Proceedings of NAACL-HLT 2018, pages 861–871
New Orleans, Louisiana, June 1 - 6, 2018.

(2007), for example, proposed a cascaded CRF model but it does not identify nested named entities of the same type. Finkel and Manning (2009) proposed building a constituency parser with constituents for each named entity in a sentence. Their approach is expensive, i.e., time complexity is cubic in the number of words in the sentence. Lu and Roth (2015) later proposed a mention hypergraph model for nested entity detection with linear time complexity. And recently, Muis and Lu (2017) introduced a multigraph representation based on mention separators for this task. All of these models depend on manually crafted features. In addition, they cannot be directly applied to extend current state-of-the-art recurrent neural network-based models — for flat named entity recognition (Lample et al., 2016) or the joint extraction of entities and relations (Katiyar and Cardie, 2016) — to handle nested entities.

In this paper, we propose a recurrent neural network-based model for nested named entity and nested entity mention recognition. We present a modification to the standard LSTM-based sequence labeling model (Sutskever et al., 2014) that handles both problems and operates linearly in the number of tokens and the number of possible output labels at any token. The proposed neural network approach additionally jointly models entity mention *head*[2] information, a subtask found to be useful for many information extraction applications. Our model significantly outperforms the previously mentioned hypergraph model of Lu and Roth (2015) and Muis and Lu (2017) on entity mention recognition for the ACE2004 and ACE2005 corpora. It also outperforms their model on *joint* extraction of nested entity mentions and their heads. Finally, we evaluate our approach on nested named entity recognition using the GENIA dataset and show that our model outperforms the previous state-of-the-art parser-based approach of Finkel and Manning (2009).

2 Related Work

Several methods have been proposed for named entity recognition in the existing literature as summarized by Nadeau and Sekine (2007) in their survey paper. Early techniques in the supervised domain have been based on hidden markov models (e.g., Zhou and Su (2002)) or, later, conditional random fields (CRFs) (e.g., McDonald and Pereira (2005)).

Many fewer approaches, however, have addressed the problem of nested entities. Alex et al. (2007) presented several techniques based on CRFs for nested named entity recognition for the GENIA dataset. They obtained their best results from a cascaded approach, where they applied CRFs in a specific order on the entity types, such that each CRF utilizes the output derived from previous CRFs. Their approach could not identify nested entities of the same type. Finkel and Manning (2009) proposed a CRF-based constituency parser for nested named entities such that each named entity is a constituent in the parse tree. Their model achieved state-of-the-art results on the GENIA dataset. However, the time complexity of their model is $O(n^3)$, where n is the number of tokens in the sentence, making inference slow. As a result, we do not adopt their parse tree-based representation of nested entities and propose instead a linear time directed hypergraph-based model similar to that of Lu and Roth (2015). Directed hypergraphs were also introduced for parsing by Klein and Manning (2001).

While most previous efforts for nested entity recognition were limited to named entities, Lu and Roth (2015) addressed the problem of nested entity mention detection where mentions can either be named, nominal or pronominal. Their hypergraph-based approach is able to represent the potentially exponentially many combinations of nested mentions of different types. They adopted a CRF-like log-linear approach to learn these mention hypergraphs and employed several hand-crafted features defined over the input sentence and the output hypergraph structure. Our approach also learns a similar hypergraph representation with differences in the types of nodes and edges in the hypergraph. It does not depend on any manually crafted features. Also, our model learns the hypergraph greedily and significantly outperforms their approach.

Recently, Muis and Lu (2017) introduced the notion of mention separators for nested entity mention detection. In contrast to the hypergraph representation that we and Lu and Roth (2015) adopt, they learn a multigraph representation and are able to perform exact inference on their structure. It is an interesting orthogonal possible approach for nested entity mention detection. How-

[2]This involves identifying the headword of a named entity or entity mention.

ever, we will show that our model also outperforms their approach on all tasks.

Recently, recurrent neural networks (RNNs) have been widely applied to several sequence labeling tasks achieving state-of-the-art results. Lample et al. (2016) proposed neural models based on long short term memory networks (LSTMs) and CRFs for named entity recognition and another transition-based approach inspired by shift-reduce parsers. Both models achieve performance comparable to a state-of-the-art model (Luo et al., 2015), but neither handles nested named entities.

3 Encoding Scheme

Figure 1 shows the desired sequence tagging output for each of three overlapping PER entities ("his", "his fellow pilot" and "his fellow pilot David Williams") according to the standard BILOU tag scheme. Our approach relies on the fact that we can (1) represent these three tag sequences in the single hypergraph structure of Figure 2 and then (2) design an LSTM-based neural network that produces the correct nested entity hypergraph for a given input sentence. In the paragraphs just below we provide a general description of hypergraphs and our task-specific use of them. Sections 3.1 and 3.2 describe the hypergraph construction process; Section 4 presents the LSTM-based sequence tagging method for automating hypergraph construction.

We express our structured prediction problem such that it corresponds to building a hypergraph that encodes the token-level gold labels for all entities in the input sentence.[3] In particular, we represent the problem as a directed hypergraph. For those new to this formalism, directed hypergraphs are very much like standard directed graphs except that nodes are connected by hyperarcs that connect a *set* of tail nodes to a *set* of head nodes. To better explain our desired output structure, we further distinguish between two types of hyperarcs — *normal edges* (or arcs) that connect a single tail node to a single head node, and *hyperarcs* that contain more than one node either as the head or as the tail. The former are shown as straight lines in Figure 2; the latter as curved edges.

In our encoding of nested entities, a hyperarc is introduced when two or more entity mentions requiring different label types are present at the same position. In Figure 2, for example, the nodes "O" (corresponding to the input token "that") and the nodes "U_PER" and "B_PER" (corresponding to the input token "his") are connected by a hyperarc because three entity mentions start at this time step from the tail "O" node (two of which share the "B_PER" tag).[4]

3.1 Hypergraph Construction

Let us first discuss how the problem of nested entity recognition can be expressed as finding a hypergraph. Our goal is to represent the BILOU tag sequences associated with "his", "his fellow pilot" and "his fellow pilot David Williams" as the single hypergraph structure of Figure 2. This is accomplished by collapsing the shared states (labels) in the output entity label sequences into a single state as shown in Figure 2: e.g., the three "O" labels for "that" become a single "O"; the two "B_PER" labels at "his" are collapsed into one "B_PER" node that joins "U_PER", the latter of which represents the entity mention "his". Thus at any time step, the representation size is bounded by the number of possible output states instead of the potentially exponential number of output sequences. We then also adjust the directed edges such that they have the same type of head node and the same type of tail node as before in Figure 1.

If we look closely at Figure 2 then we realise that there is an extra "O" node in the hypergraph corresponding to the token "his" which did not appear in any entity output sequence in Figure 1: in our task-specific hypergraph construction we make sure that there is an "O" node at every timestep to model the possibility of beginning of a new entity. The need for this will become more clear in Section 4.

Note that the hypergraph representation of our model is similar to Lu and Roth (2015). Also, the expressiveness of our model is exactly the same as Lu and Roth (2015); Muis and Lu (2017). The major difference in the two approaches is in learning.

[3] We note that the complete hypergraph for the example in Figure 1 would include nodes for all possible label types at each timestep and all possible hyperarcs between them. In this work, however, we only greedily build a sub-hypergraph for the gold labels when training.

[4] In contrast, note that the nodes "L_PER" and "O" corresponding to the input token "pilot" and the node "O" corresponding to the token "David" are connected by normal edges. Hence, our hypergraph structure contains only one special kind of hyperarc which connects a single tail node to multiple head nodes. We do not have hyperarcs that connect multiple tail nodes to a single head node.

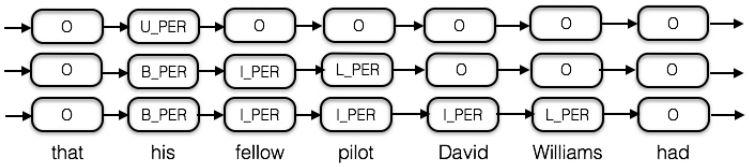


Figure 1: Nested entity mentions in an unfolded hypergraph. Each row corresponds to an entity mention sequence using the well known B_ (beginning of mention), I_ (inside a mention), L_ (last token of an entity mention), O (outside any entity mention), U_ (a single-token entity mention) tagging scheme.

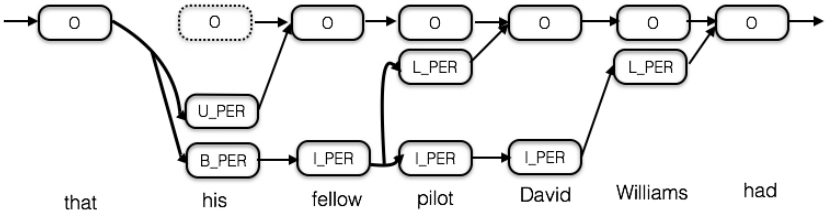


Figure 2: Directed hypergraph constructed for the example shown in Figure 1. Curved edges represent hyperarcs and straight edges are normal edges.

3.2 Edge Probability

In this section, we discuss our assignment of probabilities to all the possible edges from a tail node which helps in the greedy construction of the hypergraph. Thus at any timestep t, let g_{t-1} be the tail node and x be the current word of the sentence; then we model probability distribution over all the possible types of head nodes (different output tag types) conditioned on the tail node and the current word token. In our work we use hidden representations learned from an LSTM model as features to learn these probability distributions using a cross-entropy objective.

It is important to note that there are two types of directed edges in this hypergraph – simple edges for which there is only one head node for every tail node which can be learned as in a traditional sequence labeling task, or hyperarcs that connect more than one head node to a tail node. We learn the set of head nodes connected to a tail node by expressing it as a multi-label learning problem as described in Section 5.

3.3 Extracting Entity Mentions

As described in Section 3.2, we can assign probabilities to the different types of edges in the hypergraph and at the time of decoding we choose for each token the (normal) edge(s) with maximum probability and the hyperarcs with probability above a predefined threshold. Thus, we can extract edges at the time of decoding. Ultimately, however, we are interested in extracting nested entities from the hypergraph. For this, we construct an adjacency matrix from the edges discovered and perform depth-first search from the sentence-initial token to discover the entity mentions. This is described in detail in Section 5.1.

4 Method

We use a standard LSTM-based sequence labeling model to learn the nested entity hypergraph structure for an input sentence. Figure 3 shows part of the network structure. It is a standard bi-directional LSTM network except for a difference in the top hidden layer. When computing the representation of the top hidden layer L at any time step t, in addition to making use of the hidden unit representation from the previous time step $t-1$ and hidden unit representation from the preceding layer $L-1$, we also input the label embedding of the gold labels from the previous time step. For the token "fellow" in Figure 3, for example, we compute three different top hidden layer representations, conditioned respectively on the three labels "U_PER", "B_PER" and "O" from the previous time step $t-1$. Thus, we can model complex interactions between the input and the output. Before passing the learned hidden representation to the next time step, we average the three different top hidden layer representations. In this manner, we can model the interactions between the different overlapping labels and also it is computation-

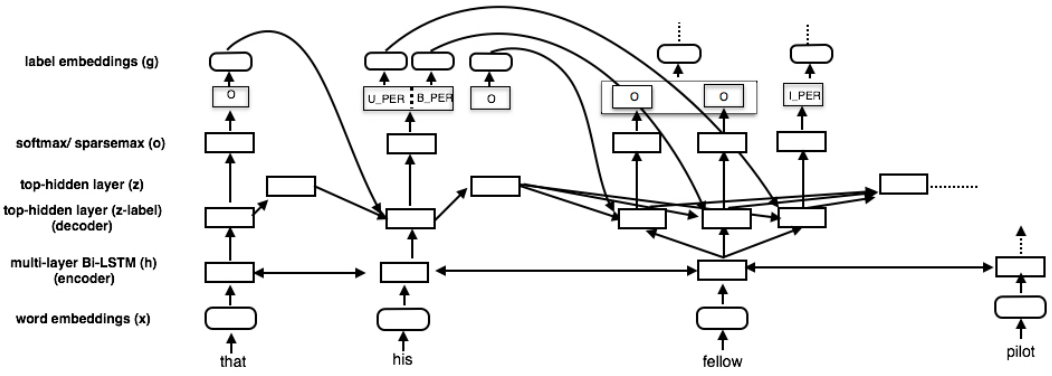


Figure 3: Dynamically computed network structure based on bi-LSTMs for nested entity mention extraction. We show part of the structure for the entity mentions in the running example in Figure 1.

ally less expensive than storing the hidden layer representations for each label sequence.

4.1 Multi-layer Bi-LSTM

We use a multi-layer bi-directional LSTM encoder, for its strength in capturing long-range dependencies between tokens, a useful property for information extraction tasks.

Using LSTMs, we can compute the hidden state $\overrightarrow{\boldsymbol{h}}_t$ in the forward direction and $\overleftarrow{\boldsymbol{h}}_t$ in the backward direction for every token, and use a linear combination of them as the token representation:

$$\overrightarrow{\boldsymbol{h}}_t^{(l)} = \text{LSTM}(\boldsymbol{x}_t, \overrightarrow{\boldsymbol{h}}_{t-1})$$
$$\overleftarrow{\boldsymbol{h}}_t^{(l)} = \text{LSTM}(\boldsymbol{x}_t, \overleftarrow{\boldsymbol{h}}_{t+1})$$
$$\boldsymbol{z}_t^{(l)} = \overrightarrow{\boldsymbol{V}}\overrightarrow{\boldsymbol{h}}_t^{(l)} + \overleftarrow{\boldsymbol{V}}\overleftarrow{\boldsymbol{h}}_t^{(l)} + \boldsymbol{b}^l$$

4.2 Top Hidden Layer

At the top hidden layer, we have a decoder-style model, with a crucial twist to accommodate the hypergraph structure, which may have multiple gold labels at the previous step. At each token t and *for each* gold label at the previous step g_{t-1}^k, our network takes the hidden representation from the previous layer $\boldsymbol{z}_t^{(L-1)}$, the hidden decoder state $\boldsymbol{h}_{t-1}^{(L)}$, as well as the gold label embedding $\boldsymbol{g}_{t-1}^k$ from the previous time step, and computes:

$$\boldsymbol{h}_t^{(L),k} = \text{LSTM}(\boldsymbol{z}_t^{(L-1)}, \boldsymbol{h}_{t-1}^{(L)}, \boldsymbol{g}_{t-1}^k)$$

Unlike the encoder LSTM, this decoder LSTM is single-directional and bifurcates when multiple gold labels are present. We use the decoder hidden states $\boldsymbol{h}_t^{(L),k}$ in the output layer for prediction, as explained in Section 4.3. However, before passing the hidden representation to the next time step we average $\boldsymbol{h}_t^{(L),k}$ over all the gold labels k:

$$\boldsymbol{h}_t^{(L)} = \frac{1}{|G_{t-1}|}\sum_k \boldsymbol{h}_t^{(L),k}$$

Thus, $\boldsymbol{h}_t^{(L)}$ summarizes the information for all the gold labels from the previous time step.

4.3 Entity Extraction

For each token t and previous gold label g_{t-1}^k, we use the decoder state $\boldsymbol{h}_t^{(L),k}$ to predict a *probability distribution* over the possible candidate labels using a linear layer followed by a normalizing transform (illustrated below with softmax). The outputs can be interpreted as conditional probabilities for the next label given the current gold label:

$$\boldsymbol{o}_t^k = \boldsymbol{U}\boldsymbol{h}_t^{(L),k} + \boldsymbol{b}$$
$$\hat{\boldsymbol{e}}_t^k = \text{softmax}(\boldsymbol{o}_t^k)$$
$$p(y_t = c | y_{t-1} = g_{t-1}^k) = (\boldsymbol{e}_t^k)_c$$

Special care is required, however, since the desired output has hyperarcs. As shown in Figure 2, there is an hyperarc between "I_PER" corresponding to the token "fellow" and the label set "L_PER" and "I_PER" corresponding to the token "pilot". Thus, in our network structure conditioned on the previous label "I_PER" in this case, we would like to predict both "L_PER" and "I_PER" as the next labels. To accommodate this, we use a multi-label training objective, as described in Section 5.

5 Training

We train our model using two different multi-label learning objectives. The idea is to represent the

	ACE2004			ACE2005		
Method	P	R	F1	P	R	F1
MH-F (Lu and Roth, 2015)	70.0	59.2	63.8	70.0	56.9	62.8
Muis and Lu (2017)	72.7	58.0	64.5	69.1	58.1	63.1
LSTM-flat	70.3	48.4	57.3	62.4	49.4	55.1
LSTM-output_layer	72.0	63.3	67.4	66.3	68.2	67.2
Our model (softmax)	72.2	65.2	68.5	70.1	67.9	69.0
Our model (sparsemax)	**73.6**	**71.8**	**72.7**	**70.6**	**70.4**	**70.5**

Table 1: Performance on ACE2004 and ACE2005 test set on mention extraction and classification.

gold labels as a distribution over all possible labels, encoded as a vector $\boldsymbol{e}$. Hence, for simple edges, the distribution has a probability of 1 for the unique gold label ($\boldsymbol{e}_g = 1$), and 0 everywhere else. For hyperarcs, we distribute the probability mass uniformly over all the gold labels in the gold label set ($\boldsymbol{e}_g^k = \frac{1}{|G|}$ for all k). Thus, for the example described earlier in Section 4.3, both the labels "L_PER" and "I_PER" receive a probability of 0.5 in the gold label distribution $\boldsymbol{e}_t^k$, conditioned on the label "I_PER" from the previous time step.

Softmax. Our first training method uses softmax to estimate the predicted probabilities, and the KL-divergence multi-label loss between the true distribution $\boldsymbol{e}_t^k$ and the predicted distribution $\hat{\boldsymbol{e}}_t^k = \text{softmax}(\boldsymbol{o}_t^k)$:

$$\ell_{t(\text{softmax})}^k = -\sum_c \left(\boldsymbol{e}_t^k\right)_c \log\left(\hat{\boldsymbol{e}}_t^k\right)_c$$

Sparsemax. Our second training method makes use of *sparsemax*, recently introduced by Martins and Astudillo (2016) as a sparse drop-in replacement to softmax, as well as a loss function. Unlike softmax, which always outputs a nonzero probability for any output, sparsemax outputs zero probability for most of the unlikely classes, leading to good empirical results on multi-label tasks. For our problem, there are only a few nested entities at any timestep in the gold labels thus using a training objective that learns a sparse distribution is more appropriate. Sparsemax can be used to filter part of the output space as in the case for multi-label problems thus leaving non-zero probability on the desired output labels.

Formally, sparsemax returns the euclidean projection of its input $\boldsymbol{o}$ onto the probability simplex:

$$\hat{\boldsymbol{e}} = \text{sparsemax}(\boldsymbol{o}) := \underset{\hat{\boldsymbol{e}} \in \Delta}{\text{argmin}} \|\boldsymbol{o} - \hat{\boldsymbol{e}}\|^2$$

The corresponding loss, a sparse version of the KL divergence, is (up to a constant):

$$\ell_{t(\text{sparsemax})}^k = -2\boldsymbol{e}_t^{k\top}\boldsymbol{o}_t^k + \sum_{c:(\hat{\boldsymbol{e}}_t^k)_c \neq 0} \left((\boldsymbol{o}_t^k)_c^2 - \tau^2\right)$$

This function is convex and differentiable, and the quantity τ is a biproduct of the simplex projection, as described in Martins and Astudillo (2016).

For either choice of probability estimation, the total loss of a training sample is the sum of losses for each token and for each previous gold label:

$$\mathcal{L} = \sum_t \sum_{k \in G_{t-1}} \ell_t^k.$$

5.1 Decoding

At the time of inference, we greedily decode our hypergraph from left-to-right to find the most likely sub-hypergraph. During training, at each timestep the most likely label set is learned conditioned on a gold label from the previous timestep. However, gold labels are not available at test time. Thus, we use the predicted labels from the previous time step as an input to the current time step to find the most likely label set. We use a hard threshold T to determine the predicted label set $P_t^k = \{c : \left(\hat{\boldsymbol{e}}_t^k\right)_c > T\}$

We can get the most likely label set P_t^c for any predicted label at the previous time step $c \in P_{t-1}^k$ using the above decoding strategy. We now combine these inferences to find the most likely entity mention sequences. We construct an adjacency matrix $\boldsymbol{A}$ for each time step, such that $\boldsymbol{A}[\hat{\boldsymbol{e}}_{t-1}^c][\hat{\boldsymbol{e}}_t^k]$ += 1 for every c in the predicted label set P_t^k at timestep t conditioned on $\hat{\boldsymbol{e}}_t^k$ and for every k in predicted labels P_{t-1} at time step $t-1$. This can be viewed as a directed hypergraph with several connected components. We then perform a depth-first search on this directed hypergraph to find all the entity mentions in the sentence.

5.2 Modeling Entity Heads for ACE datasets

The ACE datasets also have annotations for mention heads along with the entity mentions. For example, a sentence with the entity mention "the U.S. embassy" also contains an annotation for its head word which is "embassy" in this case. Thus, we modify our model to also extract the head of the entity mentions for ACE dataset. We jointly model the entity mentions and their heads. To do this, we propose a simple extension to our model by only changing the output label sequence. We introduce new labels starting with "H" to indicate that the current token in the entity mention is part of its head. Thus, we only change the output label sequence for the entity mentions to include the head label: We train with the label sequence "B_ORG I_ORG H_ORG" instead of "B_ORG I_ORG L_ORG". Also, for all our entity sequences we predict the "O" tag at the end, hence we can still extract the entity mentions. At decoding time, we output the sequence of words with the "H" tag as the head words for a mention.

6 Experiments

We evaluate our model on two tasks – nested entity mention detection for the ACE corpora and nested named entity recognition for the GENIA dataset.

6.1 ACE Experiments

6.1.1 Data

We perform experiments on the English section of the ACE2004 and ACE2005 corpora. There are 7 main entity types — Person (PER), Organization (ORG), Geographical Entities (GPE), Location (LOC), Facility (FAC), Weapon (WEA) and Vehicle (VEH). For each entity type, there are annotations for the entity mention and mention heads.

6.1.2 Evaluation Metrics

We use a strict evaluation metric similar to Lu and Roth (2015): an entity mention is considered correct if both the mention span and the mention type are exactly correct. Similarly, for the task of joint extraction of entity mentions and mention heads, the mention span, head span and the entity type should all exactly match the gold label.

6.1.3 Baselines and Previous Models

We compare our model with the feature-based model (MH-F) on hypergraph structure (Lu and Roth, 2015) on both entity mention detection as well as the joint mention and mention heads extraction. We also compare with Muis and Lu (2017) on entity mention detection only as their model cannot detect head phrases of the entity mentions. Lu and Roth (2015) compare their approach with CRF-based approaches such as a linear-chain CRF, semi-markov CRF and a cascaded approach (Alex et al., 2007) and show that their model outperforms them. Hence, we do not include those results in our paper.

We also implement several LSTM-based baselines for comparison. Our first baseline is a standard sequence labeling LSTM model (LSTM-flat). A sequence model is not capable of handling the nested mentions, so we remove the embedded entity mention and keep the mention longer in length. Our second baseline is a hypergraph model (LSTM-output_layer) except that the dependencies are only modeled at the output layer and hence there are no connections to the top-hidden layer from the label embeddings from the previous timestep; instead, these connections are limited to the output layer.

6.1.4 Hyperparameters and Training Details

We use Adadelta (Zeiler, 2012) for training our models. We initialize our word vectors with 300-dimensional word2vec (Mikolov et al., 2013) word embeddings. These word embeddings are tuned during training. We regularize our network using dropout (Srivastava et al., 2014), with the dropout rate tuned on the development set. There are 3 hidden layers in our network and the dimensionality of hidden units is 100 in all our experiments. And we set the threshold T as 0.3.

6.1.5 Results

We show the performance of our approaches in Table 1 compared to the previous state-of-the-art system (Lu and Roth, 2015; Muis and Lu, 2017) on both the ACE2004 and ACE2005 datasets. We find that our LSTM-flat baseline that ignores embedded entity mentions during training performs worse than Lu and Roth (2015); however, our other neural network-based approaches all outperform the previous feature-based approach. Among the neural network-based models, we find that our models that construct a hypergraph perform better than the LSTM-flat models. Also, our approach that models dependencies between the input and the output by passing the prediction from the pre-

	ACE2004			ACE2005		
Method	P	R	F1	P	R	F1
MH-F (Lu and Roth, 2015)	74.4	50.0	59.8	63.4	53.8	58.3
Our model(softmax)	68.2	60.5	64.2	67.5	62.3	64.8
Our model(sparsemax)	**72.3**	**66.8**	**69.7**	**70.6**	**69.8**	**70.2**

Table 2: Performance on ACE2004 and ACE2005 test set on joint entity mention and its head prediction. Muis and Lu (2017) do not predict head of the nested entity mentions.

vious timestep as shown in Figure 3 performs better than the LSTM-output_layer model which only models dependencies at the output layer. Also, as expected, the sparsemax method that produces a sparse probability distribution performs better than the softmax approach for modeling hyperedges. In summary, our sparsemax model is the best performing model.

Joint Modeling of Heads We report the performance of our best performing models on the joint modeling of entity mentions and its head in Table 2. We show that our sparsemax model is still the best performing model. We also find that as the total number of possible labels at any timestep increases because of the way we implemented the entity heads, the gains that we get after incorporating sparsemax are significantly higher compared to the results shown in Table 1.

6.2 GENIA Experiments

6.2.1 Data

We also evaluate our model on the GENIA dataset (Ohta et al., 2002) for nested named entity recognition. We follow the same dataset split as Finkel and Manning (2009); Lu and Roth (2015); Muis and Lu (2017). Thus, the first 90% of the sentences were used in training and the remaining 10% were used for evaluation. We also consider five entity types – DNA, RNA, protein, cell line and cell type.

6.2.2 Baselines and Previous Models

We compare our model with Finkel and Manning (2009) based on a constituency CRF-based parser and the mention hypergraph model by Lu and Roth (2015) and a recent multigraph model by Muis and Lu (2017).

6.2.3 Results

Table 3 shows the performance of our different models compared to the previous models. Interestingly, our LSTM-flat model outperforms Lu and Roth (2015). We suspect that it is because we use pretrained word embeddings[5] trained on PubMed data (Pyysalo et al., 2013) whereas Lu and Roth (2015) did not have access to them. We again find that our neural network model outperforms the previous state-of-the-art (Finkel and Manning, 2009; Muis and Lu, 2017) system. However, we see that both softmax and sparsemax models perform comparably on this dataset.

Method	P	R	F1
Finkel and Manning (2009)	75.4	65.9	70.3
MH-F (Lu and Roth, 2015)	72.5	65.2	68.7
Muis and Lu (2017)	75.4	66.8	70.8
LSTM-flat	75.5	63.5	68.9
LSTM-output_layer	78.4	67.9	72.8
Our model (softmax)	76.7	**71.1**	**73.8**
Our model (sparsemax)	**79.8**	68.2	73.6

Table 3: Performance on the GENIA dataset on nested named entity recognition.

7 Error Analysis

Consistent with existing results on the joint modeling of related tasks in NLP, we find that joint modeling of heads and their entity mentions leads to an increase in F-score by 1pt (i.e., 71.4 for the sparsemax model on the ACE2005 dataset) on the performance of the entity mentions. The precision on extracting entity mentions is 72.1 (vs. 70.6 in Table 1) for our sparsemax model for the ACE2005 dataset.

Example S1 below compares the output from a softmax vs. a sparsemax model on the joint modeling of an entity mention and its head on the ACE2005 dataset. Gold-standard annotations are shown in *red*.

(S1) [[[They]]]$_{\text{PERSON}}$ don't abandon [[[[[their]]]$_{\text{PERSON}}$ patients]]$_{\text{PERSON}}$, except

[5] Word vectors trained on PubMed data are available at http://bio.nlplab.org/#source-data.

for the high premiums of a few specialities?

Based on the gold standard, the models are required to extract "their" — an entity mention of type PER as well as its head — and "their patients", which overlaps with the previous entity mention "their" and has the head word "patients". This means that the models are required to predict a hyperedge from "O" to "H_PER; B_PER". We find that the softmax model shown in *blue* can only predict the entity mention "their" omitting completely the entity mention "their patients" whereas the sparsemax model shown in *green* can predict both nested entities. Overall then, sparsemax seems to allow the modeling of hyperedges more efficiently compared to the softmax model and performance gains are due to extracting more nested entities with the help of sparsemax model.

7.1 Limitations and Future Directions

We also manually scanned the test set predictions on ACE dataset for our sparsemax model to understand its current limitations.

Document Context. Given the following sentence

(S2) [They]$_{\text{VEHICLE}}$ roar, [they]$_{\text{VEHICLE}}$ screech.

the sparsemax model predicts both entity mentions of "they" as PER entity type. Only if the previous sentence in the corpus is accessible — "And if you ride inside that tank, it is like riding in the bowels of a dragon" — can we understand that "they" in S2 refers to the tank and hence is a VEH. Thus, our model can be improved by providing additional context for each sentence rather than making predictions on each sentence in the corpus independently.

Pronominal Entity Mention (It). Next, consider examples S3 and S4:

(S3) [It]$_{\text{FACILITY}}$ also seemed to be [some kind of monitoring station]$_{\text{FACILITY}}$.

(S4) It does not matter to [these people]$_{\text{PERSON}}$ that crime has skyrocketed . . .

In the example sentences, "It" refers to a facility and an event, respectively. Our model does not distinguish between the two cases and always predicts the token "It" as a non-entity. We found this true for all occurrences of the token "It" in our test set. The incorporation of coreference information can potentially overcome this limitation.

Inconsistency in Gold-standard Annotations. We also identified potential inconsistencies in the gold-standard annotations.

(S5) . . . results may affect what happens to [both of these teams]$_{\text{ORG}}$, but in just . . .

For S5, the gold-standard annotation for "both of these teams" is an ORG entity mention with the token "teams" as its head word. Our sparsemax model identifies the entity mention correctly but instead predicts the token "both" as the head. It also identifies "these teams" as another nested entity mention with the head word "teams". In contrast, however, we also found entity mentions such as "all of the victims that get a little money" for which the gold-standard has "all" annotated as its head and another nested mention "the victims that get a little money" with "victims" as the head. We recognize this as an inconsistency in the gold-standard annotation.

8 Conclusion and Future Work

In this paper, we present a novel recurrent network-based model for nested named entity recognition and nested entity mention detection. We propose a hypergraph representation for this problem and learn the structure using an LSTM network in a greedy manner. We show that our model significantly outperforms a feature based mention hypergraph model (Lu and Roth, 2015) and a recent multigraph model (Muis and Lu, 2017) on the ACE dataset. Our model also outperforms the constituency parser-based approach of Finkel and Manning (2009) on the GENIA dataset.

In future work, it would be interesting to learn global dependencies between the output labels for such a hypergraph structure and training the model globally. We can also experiment with different representations such as the one in Finkel and Manning (2009) and use the recent advances in neural network approaches (Vinyals et al., 2015) to learn the constituency parse tree efficiently.

Acknowledgments

We thank Wei Lu for help with the datasets. We also thank Jack Hessel, Vlad Niculae and the reviewers for their helpful comments and feedback. This work was supported in part by NSF grant SES-1741441 and DARPA DEFT Grant FA8750-13-2-0015. The views and conclusions contained herein are those of the authors and should not be

interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF, DARPA or the U.S. Government.

References

Beatrice Alex, Barry Haddow, and Claire Grover. 2007. Recognising nested named entities in biomedical text. In *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, BioNLP '07, pages 65–72. http://dl.acm.org/citation.cfm?id=1572392.1572404.

Kai-Wei Chang, Rajhans Samdani, and Dan Roth. 2013. A constrained latent variable model for coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 601–612. http://www.aclweb.org/anthology/D13-1057.

Yejin Choi, Eric Breck, and Claire Cardie. 2006. Joint extraction of entities and relations for opinion recognition. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Sydney, Australia, pages 431–439. http://www.aclweb.org/anthology/W/W06/W06-1651.

Jenny Rose Finkel and Christopher D. Manning. 2009. Nested named entity recognition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Singapore, pages 141–150. http://www.aclweb.org/anthology/D/D09/D09-1015.

R Florian, H Hassan, A Ittycheriah, H Jing, N Kambhatla, X Luo, N Nicolov, and S Roukos. 2004. A statistical model for multilingual entity detection and tracking. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*. Association for Computational Linguistics, Boston, Massachusetts, USA, pages 1–8.

Arzoo Katiyar and Claire Cardie. 2016. Investigating lstms for joint extraction of opinion entities and relations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. http://aclweb.org/anthology/P/P16/P16-1087.pdf.

Dan Klein and Christopher D. Manning. 2001. Parsing and hypergraphs. In *Proceedings of the Seventh International Workshop on Parsing Technologies (IWPT-2001), 17-19 October 2001, Beijing, China*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, ICML '01, pages 282–289. http://dl.acm.org/citation.cfm?id=645530.655813.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 260–270. http://www.aclweb.org/anthology/N16-1030.

Wei Lu and Dan Roth. 2015. Joint mention extraction and classification with mention hypergraphs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 857–867. http://aclweb.org/anthology/D15-1102.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint entity recognition and disambiguation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 879–888. https://aclweb.org/anthology/D/D15/D15-1104.

André F. T. Martins and Ramón F. Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. JMLR.org, ICML'16, pages 1614–1623. http://dl.acm.org/citation.cfm?id=3045390.3045561.

Ryan McDonald and Fernando Pereira. 2005. Identifying gene and protein mentions in text using conditional random fields. *BMC Bioinformatics* 6(1):S6. https://doi.org/10.1186/1471-2105-6-S1-S6.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., pages 3111–3119. http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Association for Computational Linguistics, Suntec, Singapore, pages 1003–1011. http://www.aclweb.org/anthology/P/P09/P09-1113.

Aldrian Obaja Muis and Wei Lu. 2017. Labeling gaps between words: Recognizing overlapping mentions with mention separators. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 2598–2608. https://www.aclweb.org/anthology/D17-1275.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Linguisticae Investigationes* 30(1):3–26. Publisher: John Benjamins Publishing Company.

Tomoko Ohta, Yuka Tateisi, and Jin-Dong Kim. 2002. The genia corpus: An annotated research abstract corpus in molecular biology domain. In *Proceedings of the Second International Conference on Human Language Technology Research*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, HLT '02, pages 82–86. http://dl.acm.org/citation.cfm?id=1289189.1289260.

S Pyysalo, F Ginter, H Moen, T Salakoski, and S Ananiadou. 2013. *Distributional Semantics Resources for Biomedical Text Processing*, pages 39–44.

Sunita Sarawagi and William W Cohen. 2005. Semi-markov conditional random fields for information extraction. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, MIT Press, pages 1185–1192. http://papers.nips.cc/paper/2648-semi-markov-conditional-random-fields-for-information-extraction.pdf.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15:1929–1958. http://jmlr.org/papers/v15/srivastava14a.html.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*. MIT Press, Cambridge, MA, USA, NIPS'14, pages 3104–3112. http://dl.acm.org/citation.cfm?id=2969033.2969173.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. MIT Press, Cambridge, MA, USA, NIPS'15, pages 2773–2781. http://dl.acm.org/citation.cfm?id=2969442.2969550.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR* abs/1212.5701. http://arxiv.org/abs/1212.5701.

GuoDong Zhou and Jian Su. 2002. Named entity recognition using an hmm-based chunk tagger. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '02, pages 473–480. https://doi.org/10.3115/1073083.1073163.

Simultaneously Self-Attending to All Mentions for Full-Abstract Biological Relation Extraction

Patrick Verga, Emma Strubell, Andrew McCallum

College of Information and Computer Sciences
University of Massachusetts Amherst
{pat, strubell, mccallum}@cs.umass.edu

Abstract

Most work in relation extraction forms a prediction by looking at a short span of text within a single sentence containing a single entity pair mention. This approach often does not consider interactions across mentions, requires redundant computation for each mention pair, and ignores relationships expressed across sentence boundaries. These problems are exacerbated by the document- (rather than sentence-) level annotation common in biological text. In response, we propose a model which simultaneously predicts relationships between all mention pairs in a document. We form pairwise predictions over entire paper abstracts using an efficient self-attention encoder. All-pairs mention scores allow us to perform multi-instance learning by aggregating over mentions to form entity pair representations. We further adapt to settings without mention-level annotation by jointly training to predict named entities and adding a corpus of weakly labeled data. In experiments on two Biocreative benchmark datasets, we achieve state of the art performance on the Biocreative V Chemical Disease Relation dataset for models without external KB resources. We also introduce a new dataset an order of magnitude larger than existing human-annotated biological information extraction datasets and more accurate than distantly supervised alternatives.

1 Introduction

With few exceptions (Swampillai and Stevenson, 2011; Quirk and Poon, 2017; Peng et al., 2017), nearly all work in relation extraction focuses on classifying a short span of text within a single sentence containing a single entity pair mention. However, relationships between entities are often expressed across sentence boundaries or otherwise require a larger context to disambiguate. For example, 30% of relations in the Biocreative V CDR dataset (§3.1) are expressed across sentence boundaries, such as in the following excerpt expressing a relationship between the chemical **azathioprine** and the disease **fibrosis**:

> *Treatment of psoriasis with **azathioprine**.* **Azathioprine** treatment benefited 19 (66%) out of 29 patients suffering from severe psoriasis. Haematological complications were not troublesome and results of biochemical liver function tests remained normal. Minimal cholestasis was seen in two cases and portal **fibrosis** of a reversible degree in eight. Liver biopsies should be undertaken at regular intervals if **azathioprine** therapy is continued so that structural liver damage may be detected at an early and reversible stage.

Though the entities' mentions never occur in the same sentence, the above example expresses that the chemical entity *azathioprine* can cause the side effect *fibrosis*. Relation extraction models which consider only within-sentence relation pairs cannot extract this fact without knowledge of the complicated coreference relationship between *eight* and *azathioprine treatment*, which, without features from a complicated pre-processing pipeline, cannot be learned by a model which considers entity pairs in isolation. Making separate predictions for each mention pair also obstructs multi-instance learning (Riedel et al., 2010; Surdeanu et al., 2012), a technique which aggregates entity representations from mentions in order to improve robustness to noise in the data. Like the majority of relation extraction data, most annotation for biological relations is distantly supervised, and so we could benefit from a model which is amenable to multi-instance learning.

In addition to this loss of cross-sentence and cross-mention reasoning capability, traditional mention pair relation extraction models typically introduce computational inefficiencies by independently extracting features for and scoring every pair of mentions, even when those mentions occur in the same sentence and thus could share representations. In the CDR training set, this requires separately encoding and classifying each of the 5,318 candidate mention pairs independently, versus encoding each of the 500 abstracts once. Though abstracts

Proceedings of NAACL-HLT 2018, pages 872–884
New Orleans, Louisiana, June 1 - 6, 2018.

are longer than e.g. the text between mentions, many sentences contain multiple mentions, leading to redundant computation.

However, encoding long sequences in a way which effectively incorporates long-distance context can be prohibitively expensive. Long Short Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997) are among the most popular token encoders due to their capacity to learn high-quality representations of text, but their ability to leverage the fastest computing hardware is thwarted due to their computational dependence on the length of the sequence — each token's representation requires as input the representation of the previous token, limiting the extent to which computation can be parallelized. Convolutional neural networks (CNNs), in contrast, can be executed entirely in parallel across the sequence, but the amount of context incorporated into a single token's representation is limited by the depth of the network, and very deep networks can be difficult to learn (Hochreiter, 1998). These problems are exacerbated by longer sequences, limiting the extent to which previous work explored full-abstract relation extraction.

To facilitate efficient full-abstract relation extraction from biological text, we propose Bi-affine Relation Attention Networks (BRANs), a combination of network architecture, multi-instance and multi-task learning designed to extract relations between entities in biological text without requiring explicit mention-level annotation. We synthesize convolutions and self-attention, a modification of the Transformer encoder introduced by Vaswani et al. (2017), over sub-word tokens to efficiently incorporate into token representations rich context between distant mention pairs across the entire abstract. We score all pairs of mentions in parallel using a bi-affine operator, and aggregate over mention pairs using a soft approximation of the max function in order to perform multi-instance learning. We jointly train the model to predict relations and entities, further improving robustness to noise and lack of gold annotation at the mention level.

In extensive experiments on two benchmark biological relation extraction datasets, we achieve state of the art performance for a model using no external knowledge base resources in experiments on the Biocreative V CDR dataset, and outperform comparable baselines on the Biocreative VI ChemProt dataset. We also introduce a new dataset which is an order of magnitude larger than existing gold-annotated biological relation extraction datasets while covering a wider range of entity and relation types and with higher accuracy than distantly supervised datasets of the same size. We provide a strong baseline on this new dataset, and encourage its use as a benchmark for future biological relation extraction systems.[1]

2 Model

We designed our model to efficiently encode long contexts spanning multiple sentences while forming pairwise predictions without the need for mention pair-specific features. To do this, our model first encodes input token embeddings using self-attention. These embeddings are used to predict both entities and relations. The relation extraction module converts each token to a *head* and *tail* representation. These representations are used to form mention pair predictions using a bi-affine operation with respect to learned relation embeddings. Finally, these mention pair predictions are pooled to form entity pair predictions, expressing whether each relation type is expressed by each relation pair.

2.1 Inputs

Our model takes in a sequence of N token embeddings in $\mathbb{R}^d$. Because the Transformer has no innate notion of token position, the model relies on positional embeddings which are added to the input token embeddings.[2] We learn the position embedding matrix $P^{m \times d}$ which contains a separate d dimensional embedding for each position, limited to m possible positions. Our final input representation for token x_i is:

$$x_i = s_i + p_i$$

where s_i is the token embedding for x_i and p_i is the positional embedding for the ith position. If i exceeds m, we use a randomly initialized vector in place of p_i.

We tokenize the text using byte pair encoding (BPE) (Gage, 1994; Sennrich et al., 2015). The BPE algorithm constructs a vocabulary of sub-word pieces, beginning with single characters. Then, the algorithm iteratively merges the most frequent co-occurring tokens into a new token, which is added to the vocabulary. This procedure continues until a pre-defined vocabulary size is met.

BPE is well suited for biological data for the following reasons. First, biological entities often have unique mentions made up of meaningful subcomponents, such as *1,2-dimethylhydrazine*. Additionally, tokenization of chemical entities is challenging, lacking a universally agreed upon algorithm (Krallinger et al., 2015). As we demonstrate in §3.3.2, the sub-word representations produced by BPE allow the model to formulate better predictions, likely due to better modeling of rare and unknown words.

[1] Our code and data are publicly available at: https://github.com/patverga/bran.

[2] Though our final model incorporates some convolutions, we retain the position embeddings.

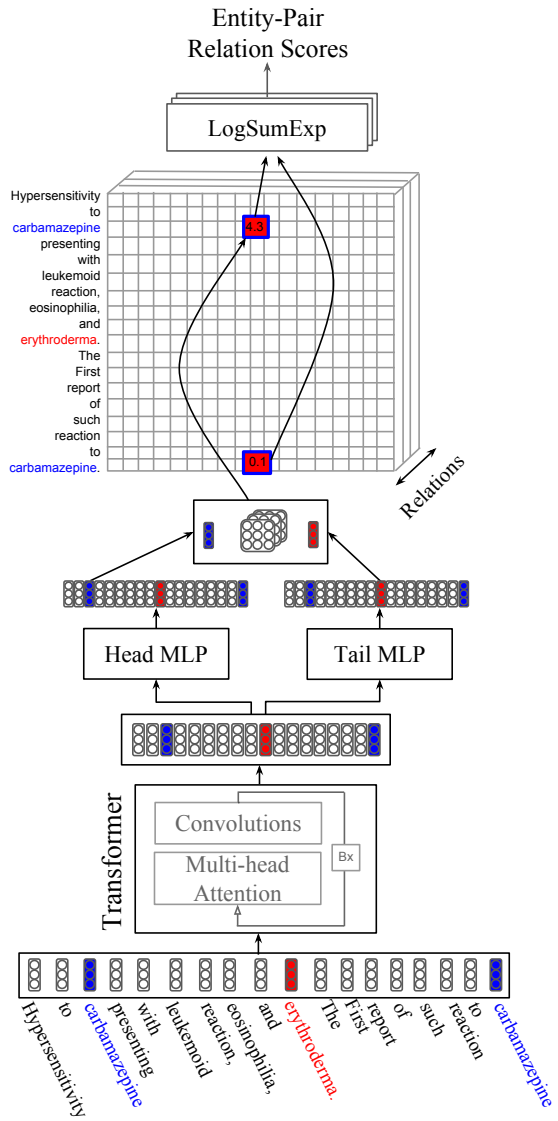


Figure 1: The relation extraction architecture. Inputs are contextually encoded using the Transformer(Vaswani et al., 2017), made up of B layers of multi-head attention and convolution subcomponents. Each transformed token is then passed through a *head* and *tail* MLP to produce two position-specific representations. A bi-affine operation is performed between each *head* and *tail* representation with respect to each relation's embedding matrix, producing a pair-wise relation affinity tensor. Finally, the scores for cells corresponding to the same entity pair are pooled with a separate LogSumExp operation for each relation to get a final score. The colored tokens illustrate calculating the score for a given pair of entities; the model is only given entity information when pooling over mentions.

2.2 Transformer

We base our token encoder on the Transformer self-attention model (Vaswani et al., 2017). The Transformer is made up of B blocks. Each Transformer block, which we denote Transformer_k, has its own set of parameters and is made up of two subcomponents: multi-head attention and a series of convolutions[3]. The output for token i of block k, $b_i^{(k)}$, is connected to its input $b_i^{(k-1)}$ with a residual connection (He et al., 2016). Starting with $b_i^{(0)} = x_i$:

$$b_i^{(k)} = b_i^{(k-1)} + \text{Transformer}_k(b_i^{(k-1)})$$

2.2.1 Multi-head Attention

Multi-head attention applies self-attention multiple times over the same inputs using separately normalized parameters (attention heads) and combines the results, as an alternative to applying one pass of attention with more parameters. The intuition behind this modeling decision is that dividing the attention into multiple heads make it easier for the model to learn to attend to different types of relevant information with each head. The self-attention updates input $b_i^{(k-1)}$ by performing a weighted sum over all tokens in the sequence, weighted by their importance for modeling token i.

Each input is projected to a key k, value v, and query q, using separate affine transformations with ReLU activations (Glorot et al., 2011). Here, k, v, and q are each in $\mathbb{R}^{\frac{d}{H}}$ where H is the number of heads. The attention weights a_{ijh} for head h between tokens i and j are computed using scaled dot-product attention:

$$a_{ijh} = \sigma\left(\frac{q_{ih}^T k_{jh}}{\sqrt{d}}\right)$$
$$o_{ih} = \sum_j v_{jh} \odot a_{ijh}$$

with $\odot$ denoting element-wise multiplication and σ indicating a softmax along the jth dimension. The scaled attention is meant to aid optimization by flattening the softmax and better distributing the gradients (Vaswani et al., 2017).

The outputs of the individual attention heads are concatenated, denoted $[\cdot;\cdot]$, into o_i. All layers in the network use residual connections between the output of the multi-headed attention and its input. Layer normalization (Ba et al., 2016), denoted $\text{LN}(\cdot)$, is then applied to the output.

$$o_i = [o_1; ...; o_h]$$
$$m_i = \text{LN}(b_i^{(k-1)} + o_i)$$

2.2.2 Convolutions

The second part of our Transformer block is a stack of convolutional layers. The sub-network used in

[3]The original Transformer uses feed-forward connections, i.e. width-1 convolutions, whereas we use convolutions with width > 1.

Vaswani et al. (2017) uses two width-1 convolutions. We add a third middle layer with kernel width 5, which we found to perform better. Many relations are expressed concisely by the immediate local context, e.g. *Michele's husband Barack*, or *labetalol-induced hypotension*. Adding this explicit n-gram modeling is meant to ease the burden on the model to learn to attend to local features. We use $C_w(\cdot)$ to denote a convolutional operator with kernel width w. Then the convolutional portion of the transformer block is given by:

$$t_i^{(0)} = \text{ReLU}(C_1(m_i))$$
$$t_i^{(1)} = \text{ReLU}(C_5(t_i^{(0)}))$$
$$t_i^{(2)} = C_1(t_i^{(1)})$$

Where the dimensions of $t_i^{(0)}$ and $t_i^{(1)}$ are in $\mathbb{R}^{4d}$ and that of $t_i^{(2)}$ is in $\mathbb{R}^d$.

2.3 Bi-affine Pairwise Scores

We project each contextually encoded token $b_i^{(B)}$ through two separate MLPs to generate two new versions of each token corresponding to whether it will serve as the first (head) or second (tail) argument of a relation:

$$e_i^{head} = W_{head}^{(1)}(\text{ReLU}(W_{head}^{(0)} b_i^{(B)}))$$
$$e_i^{tail} = W_{tail}^{(1)}(\text{ReLU}(W_{tail}^{(0)} b_i^{(B)}))$$

We use a bi-affine operator to calculate an $N \times L \times N$ tensor A of pairwise affinity scores, scoring each (head, relation, tail) triple:

$$A_{ilj} = (e_i^{head} L) e_j^{tail}$$

where L is a $d \times L \times d$ tensor, a learned embedding matrix for each of the L relations. In subsequent sections we will assume we have transposed the dimensions of A as $d \times d \times L$ for ease of indexing.

2.4 Entity Level Prediction

Our data is weakly labeled in that there are labels at the entity level but not the mention level, making the problem a form of strong-distant supervision (Mintz et al., 2009). In distant supervision, edges in a knowledge graph are heuristically applied to sentences in an auxiliary unstructured text corpus — often applying the edge label to all sentences containing the subject and object of the relation. Because this process is imprecise and introduces noise into the training data, methods like multi-instance learning were introduced (Riedel et al., 2010; Surdeanu et al., 2012). In multi-instance learning, rather than looking at each distantly labeled mention pair in isolation, the model is trained over the aggregate of these mentions and a single update is made. More recently, the weighting function of the instances has been expressed as neural network attention (Verga and McCallum, 2016; Lin et al., 2016; Yaghoobzadeh et al., 2017).

We aggregate over all representations for each mention pair in order to produce per-relation scores for each entity pair. For each entity pair (p^{head}, p^{tail}), let P^{head} denote the set of indices of mentions of the entity p^{head}, and let P^{tail} denote the indices of mentions of the entity p^{tail}. Then we use the LogSumExp function to aggregate the relation scores from A across all pairs of mentions of p^{head} and p^{tail}:

$$scores(p^{head}, p^{tail}) = \log \sum_{\substack{i \in P^{head} \\ j \in P^{tail}}} \exp(A_{ij})$$

The LogSumExp scoring function is a smooth approximation to the max function and has the benefits of aggregating information from multiple predictions and propagating dense gradients as opposed to the sparse gradient updates of the max (Das et al., 2017).

2.5 Named Entity Recognition

In addition to pairwise relation predictions, we use the Transformer output $b_i^{(B)}$ to make entity type predictions. We feed $b_i^{(B)}$ as input to a linear classifier which predicts the entity label for each token with per-class scores c_i:

$$c_i = W^{(3)} b_i^{(B)}$$

We augment the entity type labels with the BIO encoding to denote entity spans. We apply tags to the byte-pair tokenization by treating each subword within a mention span as an additional token with a corresponding B- or I- label.

2.6 Training

We train both the NER and relation extraction components of our network to perform multi-class classification using maximum likelihood, where NER classes y_i or relation classes r_i are conditionally independent given deep features produced by our model with probabilities given by the softmax function. In the case of NER, features are given by the per-token output of the transformer:

$$\frac{1}{N} \sum_{i=1}^{N} \log P(y_i \mid b_i^{(B)})$$

In the case of relation extraction, the features for each entity pair are given by the LogSumExp over pairwise scores described in § 2.4. For E entity pairs, the relation r_i is given by:

$$\frac{1}{E} \sum_{i=1}^{E} \log P(r_i \mid scores(p^{head}, p^{tail}))$$

We train the NER and relation objectives jointly, sharing all embeddings and Transformer parameters. To trade off the two objectives, we penalize the named entity updates with a hyperparameter λ.

3 Results

We evaluate our model on three datasets: The Biocreative V Chemical Disease Relation benchmark (CDR), which models relations between chemicals and diseases (§3.1); the Biocreative VI ChemProt benchmark (CPR), which models relations between chemicals and proteins (§3.2); and a new, large and accurate dataset we describe in §3.3 based on the human curation in the Chemical Toxicology Database (CTD), which models relationships between chemicals, proteins and genes.

The CDR dataset is annotated at the level of paper abstracts, requiring consideration of long-range, cross sentence relationships, thus evaluation on this dataset demonstrates that our model is capable of such reasoning. We also evaluate our model's performance in the more traditional setting which does not require cross-sentence modeling by performing experiments on the CPR dataset, for which all annotations are between two entity mentions in a single sentence. Finally, we present a new dataset constructed using strong-distant supervision (§2.4), with annotations at the document level. This dataset is significantly larger than the others, contains more relation types, and requires reasoning across sentences.

3.1 Chemical Disease Relations Dataset

The Biocreative V chemical disease relation extraction (CDR) dataset[4] (Li et al., 2016a; Wei et al., 2016) was derived from the Comparative Toxicogenomics Database (CTD), which curates interactions between genes, chemicals, and diseases (Davis et al., 2008). CTD annotations are only at the document level and do not contain mention annotations. The CDR dataset is a subset of these original annotations, supplemented with human annotated, entity linked mention annotations. The relation annotations in this dataset are also at the document level only.

3.1.1 Data Preprocessing

The CDR dataset is concerned with extracting only chemically-induced disease relationships (drug-related side effects and adverse reactions) concerning the most specific entity in the document. For example *tobacco causes cancer* could be marked as false if the document contained the more specific *lung cancer*. This can cause true relations to be labeled as false, harming evaluation performance. To address this we follow (Gu et al., 2016, 2017) and filter hypernyms according to the hierarchy in the MESH controlled vocabulary[5]. All entity pairs within the same abstract that do not have an annotated relation are assigned the NULL label.

In addition to the gold CDR data, Peng et al. (2016) add 15,448 PubMed abstracts annotated in the CTD dataset. We consider this same set of abstracts as additional training data (which we subsequently denote +Data). Since this data does not contain entity annotations, we take the annotations from Pubtator (Wei et al., 2013), a state of the art biological named entity tagger and entity linker. See §A.1 for additional data processing details. In our experiments we only evaluate our relation extraction performance and all models (including baselines) use gold entity annotations for predictions.

The byte pair vocabulary is generated over the training dataset — we use a budget of 2500 tokens when training on the gold CDR data, and a larger budget of 10,000 tokens when including extra data described above Additional implementation details are included in Appendix A.

Data split	Docs	Pos	Neg
Train	500	1,038	4,280
Development	500	1,012	4,136
Test	500	1,066	4,270
CTD	15,448	26,657	146,057

Table 1: Data statistics for the CDR Dataset and additional data from CTD. Shows the total number of abstracts, positive examples, and negative examples for each of the data set splits.

3.1.2 Baselines

We compare against the previous best reported results on this dataset not using knowledge base features.[6] Each of the baselines are ensemble methods for within- and cross-sentence relations that make use of additional linguistic features (syntactic parse and part-of-speech). Gu et al. (2017) encode mention pairs using a CNN while Zhou et al. (2016a) use an LSTM. Both make cross-sentence predictions with featurized classifiers.

3.1.3 Results

In Table 2 we show results outperforming the baselines despite using no linguistic features. We show performance averaged over 20 runs with 20 random seeds as well as an ensemble of their averaged predictions. We see a further boost in performance by adding weakly labeled data. Table 3 shows the

[4] http://www.biocreative.org/

[5] https://www.nlm.nih.gov/mesh/download/2017MeshTree.txt

[6] The highest reported score is from (Peng et al., 2016), but they use explicit lookups into the CTD knowledge base for the existence of the test entity pair.

Model	P	R	F1
Gu et al. (2016)	62.0	55.1	58.3
Zhou et al. (2016a)	55.6	68.4	61.3
Gu et al. (2017)	55.7	68.1	61.3
BRAN	55.6	70.8	**62.1** ± 0.8
+ Data	64.0	69.2	**66.2** ± 0.8
BRAN(ensemble)	63.3	67.1	65.1
+ Data	65.4	71.8	**68.4**

Table 2: Precision, recall, and F1 results on the Biocreative V CDR Dataset.

Model	P	R	F1
BRAN (Full)	55.6	70.8	**62.1** ± 0.8
– CNN only	43.9	65.5	52.4 ± 1.3
– no width-5	48.2	67.2	55.7 ± 0.9
– no NER	49.9	63.8	55.5 ± 1.8

Table 3: Results on the Biocreative V CDR Dataset showing precision, recall, and F1 for various model ablations.

effects of ablating pieces of our model. 'CNN only' removes the multi-head attention component from the transformer block, 'no width-5' replaces the width-5 convolution of the feed-forward component of the transformer with a width-1 convolution and 'no NER' removes the named entity recognition multi-task objective (§2.5).

3.2 Chemical Protein Relations Dataset

To assess our model's performance in settings where cross-sentence relationships are not explicitly evaluated, we perform experiments on the Biocreative VI ChemProt dataset (CDR) (Krallinger et al., 2017). This dataset is concerned with classifying into six relation types between chemicals and proteins, with nearly all annotated relationships occurring within the same sentence.

3.2.1 Baselines

We compare our models against those competing in the official Biocreative VI competition (Liu et al., 2017). We compare to the top performing team whose model is directly comparable with ours — i.e. used a single (non-ensemble) model trained only on the training data (many teams use the development set as additional training data). The baseline models are standard state of the art relation extraction models: CNNs and Gated RNNs with attention. Each of these baselines uses mention-specific features encoding relative position of each token to the two target entities being classified, whereas our model aggregates over all mention pairs in each sentence. It is also worth noting that these models use a large vocabulary of pre-trained word embeddings, giving their models the advantage of far more model parameters, as well as additional information from unsupervised pre-training.

Model	P	R	F1
CNN†	50.7	43.0	46.5
GRU+Attention†	53.0	46.3	49.5
BRAN	48.0	54.1	**50.8** ± .01

Table 4: Precision, recall, and F1 results on the Biocreative VI Chem-Prot Dataset. † denotes results from Liu et al. (2017)

3.2.2 Results

In Table 4 we see that even though our model forms all predictions simultaneously between all pairs of entities within the sentence, we are able to outperform state of the art models classifying each mention pair independently. The scores shown are averaged across 10 runs with 10 random seeds. Interestingly, our model appears to have higher recall and lower precision, while the baseline models are both precision-biased, with lower recall. This suggests that combining these styles of model could lead to further gains on this task.

3.3 New CTD Dataset

3.3.1 Data

Existing biological relation extraction datasets including both CDR (§3.1) and CPR (§3.2) are relatively small, typically consisting of hundreds or a few thousand annotated examples. Distant supervision datasets apply document-independent, entity-level annotations to all sentences leading to a large proportion of incorrect labels. Evaluations on this data involve either very small (a few hundred) gold annotated examples or cross validation to predict the noisy, distantly applied labels (Mallory et al., 2015; Quirk and Poon, 2017; Peng et al., 2017).

We address these issues by constructing a new dataset using strong-distant supervision containing document-level annotations. The Comparative Toxicogenomics Database (CTD) curates interactions between genes, chemicals, and diseases. Each relation in the CTD is associated with a disambiguated entity pair and a PubMed article where the relation was observed.

To construct this dataset, we collect the abstracts for each of the PubMed articles with at least one curated relation in the CTD database. As in §3.1, we use PubTator to automatically tag and disambiguate the entities in each of these abstracts. If both entities in the relation are found in the abstract, we take the (abstract, relation) pair as a positive example. The evidence for the curated relation could occur anywhere in the full text article, not just the abstract. Abstracts with no recovered relations are discarded. All other entity pairs with valid types and without an annotated relation that

Types	Docs	Pos	Neg
Total	68,400	166,474	1198,493
Chemical/Disease	64,139	93,940	571,932
Chemical/Gene	34,883	63,463	360,100
Gene/Disease	32,286	9,071	266,461

Table 5: Data statistics for the new CTD dataset.

occur in the remaining abstracts are considered negative examples and assigned the NULL label. We additionally remove abstracts containing greater than 500 tokens[7]. This limit removed about 10% of the total data including numerous extremely long abstracts. The average token length of the remaining data was $\tilde{2}30$ tokens. With this procedure, we are able to collect 166,474 positive examples over 13 relation types, with more detailed statistics of the dataset listed in Table 5.

We consider relations between chemical-disease, chemical-gene, and gene-disease entity pairs downloaded from CTD[8]. We remove inferred relations (those without an associated PubMed ID) and consider only human curated relationships. Some chemical-gene entity pairs were associated with multiple relation types in the same document. We consider each of these relation types as a separate positive example.

The chemical-gene relation data contains over 100 types organized in a shallow hierarchy. Many of these types are extremely infrequent, so we map all relations to the highest parent in the hierarchy, resulting in 13 relation types. Most of these chemical-gene relations have an increase and decrease version such as increase_expression and decrease_expression. In some cases, there is also an affects relation (affects_expression) which is used when the directionality is unknown. If the affects version is more common, we map decrease and increase to affects. If affects is less common, we drop the affects examples and keep the increase and decrease examples as distinct relations, resulting in the final set of 10 chemical-gene relation types.

3.3.2 Results

In Table 7 we list precision, recall and F1 achieved by our model on the CTD dataset, both overall and by relation type. Our model predicts each of the relation types effectively, with higher performance on relations with more support.

In Table 8 we see that our sub-word BPE model out-performs the model using the Genia tokenizer (Kulick et al., 2012) even though our vocabulary size is one-fifth as large. We see a 1.7 F1 point boost in predicting Pubtator NER labels for BPE. This could be explained by the increased out-of-vocabulary (OOV) rate for named entities. Word training data has 3.01 percent OOV rate for tokens with an entity. The byte pair-encoded data has an OOV rate of 2.48 percent. Note that in both the word-tokenized and byte pair-tokenized data, we replace tokens that occur less than five times with a learned UNK token.

[7]We include scripts to generate the unfiltered set of data as well to encourage future research

[8]http://ctdbase.org/downloads/

	Train	Dev	Test
Total	120k	15k	15k
Chemical/Disease			
marker/mechanism	41,562	5,126	5,167
therapeutic	24,151	2,929	3,059
Gene/Disease			
marker/mechanism	5,930	825	819
therapeutic	560	77	75
Chemical/Gene			
increase_expression	15,851	1,958	2,137
increase_MP	5,986	740	638
decrease_expression	5,870	698	783
increase_activity	4,154	467	497
affects_response	3,834	475	508
decrease_activity	3,124	396	434
affects_transport	3,009	333	361
increase_reaction	2,881	367	353
decrease_reaction	2,221	247	269
decrease_MP	798	100	120

Table 6: Data statistics for the new CTD dataset broken down by relation type. The first column lists relation types separated by the types of the entities. Columns 2–4 show the number of positive examples of that relation type. MP stands for metabolic processing.

Figure 2 depicts the model's performance on relation extraction as a function of distance between entities. For example, the blue bar depicts performance when removing all entity pair candidates (positive and negative) whose closest mentions are more than 11 tokens apart. We consider removing entity pair candidates with distances of 11, 25, 50, 100 and 500 (the maximum document length). The average sentence length is 22 tokens. We see that the model is not simply relying on short range relationships, but is leveraging information about distant entity pairs, with accuracy increasing as the maximum distance considered increases. Note that all results are taken from the same model trained on the full unfiltered training set.

4 Related work

Relation extraction is a heavily studied area in the NLP community. Most work focuses on news and web data (Doddington et al., 2004; Riedel et al., 2010; Hendrickx et al., 2009).[9] Recent neural net-

[9]And TAC KBP: https://tac.nist.gov

	P	R	F1
Total			
Micro F1	44.8	50.2	47.3
Macro F1	34.0	29.8	31.7
Chemical/Disease			
marker/mechanism	46.2	57.9	51.3
therapeutic	55.7	67.1	60.8
Gene/Disease			
marker/mechanism	42.2	44.4	43.0
therapeutic	52.6	10.1	15.8
Chemical/Gene			
increases_expression	39.7	48.0	43.3
increases_MP	26.3	35.5	29.9
decreases_expression	34.4	32.9	33.4
increases_activity	24.5	24.7	24.4
affects_response	40.9	35.5	37.4
decreases_activity	30.8	19.4	23.5
affects_transport	28.7	23.8	25.8
increases_reaction	12.8	5.6	7.4
decreases_reaction	12.3	5.7	7.4
decreases_MP	28.9	7.0	11.0

Table 7: BRAN precision, recall and F1 results for the full CTD dataset by relation type. The model is optimized for micro F1 score across all types.

Model	P	R	F1
Relation extraction			
Words	44.9	48.8	46.7 ± 0.39
BPE	44.8	50.2	**47.3** ± 0.19
NER			
Words	91.0	90.7	90.9 ± 0.13
BPE	91.5	93.6	**92.6** ± 0.12

Table 8: Precision, recall, and F1 results for CTD named entity recognition and relation extraction, comparing BPE to word-level tokenization.

work approaches to relation extraction have focused on CNNs (dos Santos et al., 2015; Zeng et al., 2015) or LSTMs (Miwa and Bansal, 2016; Verga et al., 2016a; Zhou et al., 2016b) and replacing stage-wise information extraction pipelines with a single end-to-end model (Miwa and Bansal, 2016; Ammar et al., 2017; Li et al., 2017). These models all consider mention pairs separately.

There is also a considerable body of work specifically geared towards supervised biological relation extraction including protein-protein (Pyysalo et al., 2007; Poon et al., 2014; Mallory et al., 2015), drug-drug (Segura-Bedmar et al., 2013), and chemical-disease (Gurulingappa et al., 2012; Li et al., 2016a) interactions, and more complex events (Kim et al., 2008; Riedel et al., 2011). Our work focuses on modeling relations between chemicals, diseases, genes and proteins, where available annotation is often at the document- or abstract-level, rather than the

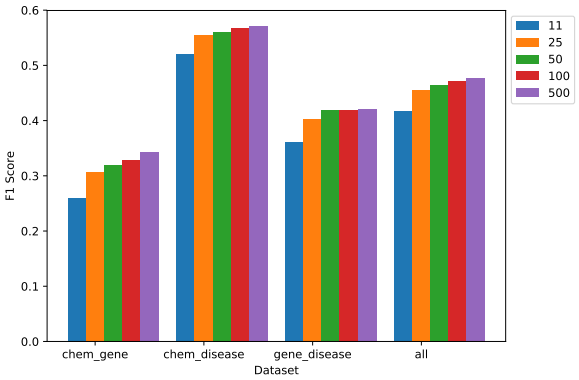


Figure 2: Performance on the CTD dataset when restricting candidate entity pairs by distance. The x-axis shows the coarse-grained relation type. The y-axis shows F1 score. Different colors denote maximum distance cutoffs.

sentence level.

Some previous work exists on cross-sentence relation extraction. Swampillai and Stevenson (2011) and Quirk and Poon (2017) consider featurized classifiers over cross-sentence syntactic parses. Most similar to our work is that of Peng et al. (2017), which uses a variant of an LSTM to encode document-level syntactic parse trees. Our work differs in three key ways. First, we operate over raw tokens negating the need for part-of-speech or syntactic parse features which can lead to cascading errors. We also use a feed-forward neural architecture which encodes long sequences far more efficiently compared to the graph LSTM network of Peng et al. (2017). Finally, our model considers all mention pairs simultaneously rather than a single mention pair at a time.

We employ a bi-affine function to form pairwise predictions between mentions. Such models have also been used for knowledge graph link prediction (Nickel et al., 2011; Li et al., 2016b), with variations such as restricting the bilinear relation matrix to be diagonal (Yang et al., 2015) or diagonal and complex (Trouillon et al., 2016). Our model is similar to recent approaches to graph-based dependency parsing, where bilinear parameters are used to score head-dependent compatibility (Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017).

5 Conclusion

We present a bi-affine relation attention network that simultaneously scores all mention pairs within a document. Our model performs well on three datasets, including two standard benchmark biological relation extraction datasets and a new, large and high-quality dataset introduced in this work. Our model out-performs the previous state of the art on the Biocreative V CDR dataset despite us-

ing no additional linguistic resources or mention pair-specific features.

Our current model predicts only into a fixed schema of relations given by the data. However, this could be ameliorated by integrating our model into open relation extraction architectures such as Universal Schema (Riedel et al., 2013; Verga et al., 2016b). Our model also lends itself to other pairwise scoring tasks such as hypernym prediction, co-reference resolution, and entity resolution. We will investigate these directions in future work.

Acknowledgments

We thank Ofer Shai and the Chan Zuckerberg Initiative / Meta data science team for helpful discussions. We also thank Timothy Dozat and Kyubyong Park for releasing their code.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. http://tensorflow.org/.

Waleed Ammar, Matthew E. Peters, Chandra Bhagavatula, and Russell Power. 2017. The ai2 system at semeval-2017 task 10 (scienceie): semi-supervised end-to-end entity and relation extraction. *nucleus* 2(e2):e2.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* .

Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. 2017. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, Valencia, Spain, pages 132–141. http://www.aclweb.org/anthology/E17-1013.

Allan Peter Davis, Cynthia G Murphy, Cynthia A Saraceni-Richards, Michael C Rosenstein, Thomas C Wiegers, and Carolyn J Mattingly. 2008. Comparative toxicogenomics database: a knowledgebase and discovery tool for chemical–gene–disease networks. *Nucleic acids research* 37(suppl_1):D786–D792.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ace) program tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation*.

Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 626–634. http://www.aclweb.org/anthology/P15-1061.

Timothy Dozat and Christopher D Manning. 2017. Deep biaffine attention for neural dependency parsing. *5th International Conference on Learning Representations* .

Philip Gage. 1994. A new algorithm for data compression. *The C Users Journal* 12(2):23–38.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. pages 315–323.

Jinghang Gu, Longhua Qian, and Guodong Zhou. 2016. Chemical-induced disease relation extraction with various linguistic features. *Database* 2016.

Jinghang Gu, Fuqing Sun, Longhua Qian, and Guodong Zhou. 2017. Chemical-induced disease relation extraction via convolutional neural network. *Database* 2017.

Harsha Gurulingappa, Abdul Mateen Rajput, Angus Roberts, Juliane Fluck, Martin Hofmann-Apitius, and Luca Toldo. 2012. Development of a benchmark corpus to support the automatic extraction of drug-related adverse effects from medical case reports. *Journal of biomedical informatics* 45(5):885–892.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 770–778.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of*

the Workshop on Semantic Evaluations: Recent Achievements and Future Directions. Association for Computational Linguistics, pages 94–99.

Sepp Hochreiter. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6(2):107–116.

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

Jin-Dong Kim, Tomoko Ohta, and Jun'ichi Tsujii. 2008. Corpus annotation for mining biomedical events from literature. *BMC bioinformatics* 9(1):10.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference for Learning Representations (ICLR)*. San Diego, California, USA.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics* 4:313–327. https://transacl.org/ojs/index.php/tacl/article/view/885.

Martin Krallinger, Obdulia Rabal, Saber A. Akhondi, Martín Pérez Pérez, Jesús Santamaría, Pérez Gael Rodríguez, Georgios Tsatsaronis, Ander Intxaurrondo, José Antonio López, Umesh Nandal, Erin Van Buel, Akileshwari Chandrasekhar, Marleen Rodenburg, Astrid Laegreid, Marius Doornenbal, Julen Oyarzabal, Analia Lourenço, and Alfonso Valencia. 2017. Overview of the biocreative vi chemical-protein interaction track. *Proceedings of the BioCreative VI Workshop* page 140.

Martin Krallinger, Obdulia Rabal, Florian Leitner, Miguel Vazquez, David Salgado, Zhiyong Lu, Robert Leaman, Yanan Lu, Donghong Ji, Daniel M Lowe, et al. 2015. The chemdner corpus of chemicals and drugs and its annotation principles. *Journal of cheminformatics* 7(S1):S2.

Seth Kulick, Ann Bies, Mark Liberman, Mark Mandel, Scott Winters, and Pete White. 2012. Integrated annotation for biomedical information extraction. *HLT/NAACL Workshop: Biolink* .

Fei Li, Meishan Zhang, Guohong Fu, and Donghong Ji. 2017. A neural joint model for entity and relation extraction from biomedical text. *BMC bioinformatics* 18(1):198.

Jiao Li, Yueping Sun, Robin J Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J Mattingly, Thomas C Wiegers, and Zhiyong Lu. 2016a. Biocreative v cdr task corpus: a resource for chemical disease relation extraction. *Database* 2016.

Xiang Li, Aynaz Taheri, Lifu Tu, and Kevin Gimpel. 2016b. Commonsense knowledge base completion. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1445–1455. http://www.aclweb.org/anthology/P16-1137.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 2124–2133. http://www.aclweb.org/anthology/P16-1200.

Sijia Liu, Feichen Shen, Yanshan Wang, Majid Rastegar-Mojarad, Ravikumar Komandur Elayavilli, Vipin Chaundary, and Hongfang Liu. 2017. Attention-based neural networks for chemical protein relation extraction. *Proceedings of the BioCreative VI Workshop* .

Emily K Mallory, Ce Zhang, Christopher Ré, and Russ B Altman. 2015. Large-scale extraction of gene interactions from full-text literature using deepdive. *Bioinformatics* 32(1):106–113.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Association for Computational Linguistics, Suntec, Singapore, pages 1003–1011. http://www.aclweb.org/anthology/P/P09/P09-1113.

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

Arvind Neelakantan, Luke Vilnis, Quoc V Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, and James Martens. 2015. Adding gradient noise improves learning for very deep networks. *arXiv preprint arXiv:1511.06807* .

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th international conference on machine*

learning (ICML-11). Bellevue, Washington, USA, pages 809–816.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph lstms. *Transactions of the Association for Computational Linguistics* 5:101–115.

Yifan Peng, Chih-Hsuan Wei, and Zhiyong Lu. 2016. Improving chemical disease relation extraction with rich features and weakly labeled data. *Journal of cheminformatics* 8(1):53.

Hoifung Poon, Kristina Toutanova, and Chris Quirk. 2014. Distant supervision for cancer pathway extraction from text. In *Pacific Symposium on Biocomputing Co-Chairs*. pages 120–131.

Sampo Pyysalo, Filip Ginter, Juho Heimonen, Jari Björne, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. 2007. Bioinfer: a corpus for information extraction in the biomedical domain. *BMC bioinformatics* 8(1):50.

Chris Quirk and Hoifung Poon. 2017. Distant supervision for relation extraction beyond the sentence boundary. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, Valencia, Spain, pages 1171–1182.

Sebastian Riedel, David McClosky, Mihai Surdeanu, Andrew McCallum, and Christopher D. Manning. 2011. Model combination for event extraction in bionlp 2011. In *Proceedings of BioNLP Shared Task 2011 Workshop*. Association for Computational Linguistics, Portland, Oregon, USA, pages 51–55. http://www.aclweb.org/anthology/W11-1808.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. *Machine learning and knowledge discovery in databases* pages 148–163.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of NAACL-HLT*. pages 74–84.

Isabel Segura-Bedmar, Paloma Martínez, and María Herrero Zazo. 2013. Semeval-2013 task 9 : Extraction of drug-drug interactions from biomedical texts (ddiextraction 2013). In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. Association for Computational Linguistics, Atlanta, Georgia, USA, pages 341–350. http://www.aclweb.org/anthology/S13-2056.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909* .

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research* 15(1):1929–1958.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, Jeju Island, Korea, pages 455–465. http://www.aclweb.org/anthology/D12-1042.

Kumutha Swampillai and Mark Stevenson. 2011. Extracting relations within and across sentences. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*. RANLP 2011 Organising Committee, Hissar, Bulgaria, pages 25–32. http://aclweb.org/anthology/R11-1004.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*. pages 2071–2080.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* .

Patrick Verga, David Belanger, Emma Strubell, Benjamin Roth, and Andrew McCallum. 2016a. Multilingual relation extraction using compositional universal schema. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 886–896. http://www.aclweb.org/anthology/N16-1103.

Patrick Verga, David Belanger, Emma Strubell, Benjamin Roth, and Andrew McCallum. 2016b. Multilingual relation extraction using compositional universal schema. In *Proceedings of NAACL-HLT*. pages 886–896.

Patrick Verga and Andrew McCallum. 2016. Row-less universal schema. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*. Association for Computational Linguistics, San Diego, CA, pages 63–68. http://www.aclweb.org/anthology/W16-1312.

Chih-Hsuan Wei, Hung-Yu Kao, and Zhiyong Lu. 2013. Pubtator: a web-based text mining tool for assisting biocuration. *Nucleic Acids Research* 41. https://doi.org/10.1093/nar/gkt441.

Chih-Hsuan Wei, Yifan Peng, Robert Leaman, Allan Peter Davis, Carolyn J Mattingly, Jiao Li, Thomas C Wiegers, and Zhiyong Lu. 2016. Assessing the state of the art in biomedical relation extraction: overview of the biocreative v chemical-disease relation (cdr) task. *Database* 2016.

Yadollah Yaghoobzadeh, Heike Adel, and Hinrich Schütze. 2017. Noise mitigation for neural entity typing and relation extraction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, Valencia, Spain, pages 1183–1194. http://www.aclweb.org/anthology/E17-1111.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *3rd International Conference for Learning Representations (ICLR)*. San Diego, California, USA.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1753–1762. http://aclweb.org/anthology/D15-1203.

Huiwei Zhou, Huijie Deng, Long Chen, Yunlong Yang, Chen Jia, and Degen Huang. 2016a. Exploiting syntactic and semantics information for chemical–disease relation extraction. *Database* 2016.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016b. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 207–212. http://anthology.aclweb.org/P16-2034.

A Implementation Details

The model is implemented in Tensorflow (Abadi et al., 2015) and trained on a single TitanX gpu. The number of transformer block repeats is $B = 2$. We optimize the model using Adam (Kingma and Ba, 2015) with best parameters chosen for ϵ, β_1, β_2 chosen from the development set. The learning rate is set to 0.0005 and batch size 32. In all of our experiments we set the number of attention heads to $h = 4$.

We clip the gradients to norm 10 and apply noise to the gradients (Neelakantan et al., 2015). We tune the decision threshold for each relation type separately and perform early stopping on the development set. We apply dropout (Srivastava et al., 2014) to the input layer randomly replacing words with a special UNK token with keep probability .85. We additionally apply dropout to the input T (word embedding + position embedding), interior layers, and final state. At each step, we randomly sample a positive or negative (NULL class) minibatch with probability 0.5.

A.1 Chemical Disease Relations Dataset

Token embeddings are pre-trained using skipgram (Mikolov et al., 2013) over a random subset of 10% of all PubMed abstracts with window size 10 and 20 negative samples. We merge the train and development sets and randomly take 850 abstracts for training and 150 for early stopping. Our reported results are averaged over 10 runs and using different splits. All baselines train on both the train and development set. Models took between 4 and 8 hours to train.

ϵ was set to 1e-4, β_1 to .1, and β_2 to 0.9. Gradient noise $\eta = .1$. Dropout was applied to the word embeddings with keep probability 0.85, internal layers with 0.95 and final bilinear projection with 0.35 for the standard CRD dataset experiments. When adding the additional weakly labeled data: word embeddings with keep probability 0.95, internal layers with 0.95 and final bilinear projection with 0.5.

A.2 Chemical Protein Relations Dataset

We construct our byte-pair encoding vocabulary using a budget of 7500. The dataset contains annotations for a larger set of relation types than are used in evaluation. We train on only the relation types in the evaluation set and set the remaining types to the Null relation. The embedding dimension is set to 200 and all embeddings are randomly initialized. ϵ was set to 1e-8, β_1 to .1, and β_2 to 0.9. Gradient noise $\eta = 1.0$. Dropout was applied to the word embeddings with keep probability 0.5, internal layers with 1.0 and final bilinear projection with 0.85 for the standard CRD dataset experiments.

A.3 Full CTD Dataset

We tune separate decision boundaries for each relation type on the development set. For each prediction, the relation type with the maximum probability is assigned. If the probability is below the relation specific threshold, the prediction is set to NULL. We use embedding dimension 128 with all embeddings randomly initialized. Our byte pair encoding vocabulary is constructed with a budget of 50,000. Models took 1 to 2 days to train.

ϵ was set to 1e-4, β_1 to .1, and β_2 to 0.9. Gradient noise $\eta = .1$.Dropout was applied to the word embeddings with keep probability 0.95, internal layers with 0.95 and final bilinear projection with 0.5

Supervised Open Information Extraction

Gabriel Stanovsky[*2,3], **Julian Michael**[2], **Luke Zettlemoyer**[2], **and Ido Dagan**[1]

[1]Bar-Ilan University Computer Science Department, Ramat Gan, Israel
[2]Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, WA
[3]Allen Institute for Artificial Intelligence, Seattle, WA
{gabis,julianjm,lsz}@cs.washington.edu
dagan@cs.biu.ac.il

Abstract

We present data and methods that enable a supervised learning approach to Open Information Extraction (Open IE). Central to the approach is a novel formulation of Open IE as a sequence tagging problem, addressing challenges such as encoding multiple extractions for a predicate. We also develop a bi-LSTM transducer, extending recent deep Semantic Role Labeling models to extract Open IE tuples and provide confidence scores for tuning their precision-recall tradeoff. Furthermore, we show that the recently released Question-Answer Meaning Representation dataset can be automatically converted into an Open IE corpus which significantly increases the amount of available training data. Our supervised model, made publicly available,[1] outperforms the state-of-the-art in Open IE on benchmark datasets.

1 Introduction

Open Information Extraction (Open IE) systems extract tuples of natural language expressions that represent the basic propositions asserted by a sentence (see Figure 1). They have been used for a wide variety of tasks, such as textual entailment (Berant et al., 2011), question answering (Fader et al., 2014), and knowledge base population (Angeli et al., 2015). However, perhaps due to limited data, existing methods use semi-supervised approaches (Banko et al., 2007; Wu and Weld, 2010), or rule-based algorithms (Fader et al., 2011; Mausam et al., 2012; Del Corro and Gemulla, 2013). In this paper, we present new data and methods for Open IE, showing that supervised learning can greatly improve performance.

Mercury filling, particularly prevalent in the USA, was banned in the EU, partly because it causes antibiotic resistance.

(mercury filling; **particularly prevalent**; in the USA)
(mercury filling; **causes**; antibiotic resistance)
(mercury filling; **was banned**; in the EU; partly because it causes antibiotic resistance)

Figure 1: Open IE extractions from an example sentence. Each proposition is composed of a tuple with a single predicate position (in bold), and an ordered list of arguments, separated by semicolons.

We build on recent work that studies other natural-language driven representations of predicate argument structure, which can be annotated by non-experts. Recently, Stanovsky and Dagan (2016) created the first labeled corpus for evaluation of Open IE by an automatic translation from question-answer driven semantic role labeling (QA-SRL) annotations (He et al., 2015). We extend these techniques and apply them to the QAMR corpus (Michael et al., 2018), an open variant of QA-SRL that covers a wider range of predicate-argument structures (Section 5). The combined dataset is the first corpus that is large and diverse enough to train an accurate extractor.

To train on this data, we formulate Open IE as a sequence labeling problem. We introduce a novel approach that can extract multiple, overlapping tuples for each sentence (Section 3), extending recent deep BIO taggers used for semantic role labeling (Zhou and Xu, 2015; He et al., 2017). We also introduce a method to calculate extraction confidence, allowing us to effectively trade off precision and recall (Section 4).

Experiments demonstrate that our approach out-

[*]Work performed while at Bar-Ilan University.

[1]Our code and models are made publicly available at https://github.com/gabrielStanovsky/supervised-oie

Proceedings of NAACL-HLT 2018, pages 885–895
New Orleans, Louisiana, June 1 - 6, 2018.

performs state-of-the-art Open IE systems on several benchmarks (Section 6), including three that were collected independently of our work (Xu et al., 2013; de Sá Mesquita et al., 2013; Schneider et al., 2017). This shows that for Open IE, careful data curation and model design can push the state of the art using supervised learning.

2 Background

In this section we survey existing Open IE systems, against which we compare our system, and available data for the task, that we will use for training and testing our model.

2.1 Different Open IE Systems and Flavors

Open IE's original goal (Banko et al., 2007) was to extend traditional (closed) information extraction, such that *a*ll of the propositions asserted by a given input sentence are extracted (see Figure 1 for examples). The broadness of this definition, along with the lack of a standard benchmark dataset for the task, prompted the development of various Open IE systems tackling different facets of the task.

While most Open IE systems aim to extract the common case of verbal binary propositions (i.e, subject-verb-object tuples), some systems specialize in other syntactic constructions, including noun-mediated relations (Yahya et al., 2014; Pal and Mausam, 2016), n-ary relations (Akbik and Löser, 2012), or nested propositions (Bhutani et al., 2016).

Many different modeling approaches have also been developed for Open IE. Some of the early systems made use of distant supervision (Banko et al., 2007; Wu and Weld, 2010), while the current best systems use rule-based techniques to extract predicate-argument structures as a post-processing step over an intermediate representation. ReVerb (Fader et al., 2011) extracts Open IE propositions from part of speech tags, OLLIE (Mausam et al., 2012), ClausIE (Del Corro and Gemulla, 2013) and PropS (Stanovsky et al., 2016) post-process dependency trees, and Open IE4[2] extracts tuples from Semantic Role Labeling (SRL) structures. These systems typically associate a confidence metric with each extraction, which allows end applications to trade off precision and recall.

[2] https://github.com/dair-iitd/OpenIE-standalone

2.2 Open IE Corpora

Recent work addressed the lack of labeled reference Open IE datasets for comparatively evaluating extractors. Stanovsky and Dagan (2016) created a large Open IE corpus (OIE2016) for verbal predicates by automatic conversion from QA-SRL (He et al., 2015), a variant of traditional SRL that labels arguments of verbs with simple, template-based natural language questions. Schneider et al. (2017) aggregated datasets annotated independently in previous Open IE efforts (WEB and NYT (de Sá Mesquita et al., 2013), PENN (Xu et al., 2013), and OIE2016) into a common benchmarking suite.

In addition to these, we create and make available a new Open IE training corpus, All Words Open IE (AW-OIE), derived from Question-Answer Meaning Representation (QAMR) (Michael et al., 2018), a recent extension of the QA-SRL paradigm to free-form questions over a wide range of predicate types (see Section 5). Table 1 presents more details on these datasets.

3 Task Formulation

In this work, we choose to model an Open IE proposition as a tuple consisting of a single predicate operating over a non-empty set of arguments, where the predicate and the arguments are contiguous spans from the sentence. As with traditional (binary) Open IE, every tuple should be asserted by the sentence and the order of the tuple elements should be such that it would be naturally interpretable when reading from left to right (for example, see the third tuple in Figure 1). As we show in following sections, this formulation intuitively lends itself to BIO tagging, while being expressive enough to capture a wide range of propositions.

Formally, given an input sentence S =

Dataset	Domain	#Sent.	#Tuples		
			Train	Dev	Test
AW-OIE*	Wikinews,Wiki	3300	12952	4213	-
OIE2016	News,Wiki	3200	5077	1671	1729
WEB-500	News,Web	500	-	-	461
NYT-222	News,Wiki	222	-	-	222
PENN-100	Mixed	100	-	-	51

Table 1: Datasets used in this work, following (Schneider et al., 2017). *AW-OIE (All Words Open IE) was created in the course of this work, see Section 5 for details.

Open IE Encoding Examples

(a) *The president claimed that he won the majority vote.*
(The president; **claimed that he won**; the majority vote)

The$_{A0-B}$ president$_{A0-I}$ claimed$_{P-B}$ that$_{P-I}$ he$_{P-I}$ won$_{P-I}$ the$_{A1-B}$ majority$_{A1-I}$ vote$_{A1-I}$

(b) *Barack Obama, a former U.S president, was born in Hawaii.*
(Barack Obama; **was born in**; Hawaii)
(a former U.S. president; **was born in**; Hawaii)

Barack$_{A0-B}$ Obama$_{A0-I}$,$_{O}$ a$_{A0-B}$ former$_{A0-I}$ U.S.$_{A0-I}$ president$_{A0-I}$,$_{O}$ was$_{P-B}$ born$_{P-I}$ in$_{P-I}$ Hawaii$_{A1-B}$

(c) *Theresa May plans for Brexit, on which the UK has voted last June.*
(the UK; **has voted on**; Brexit; last June)

Theresa$_{O}$ May$_{O}$ plans$_{O}$ for$_{O}$ Brexit$_{A1-B}$,$_{O}$ on$_{O}$ which$_{O}$ the$_{A0-B}$ UK$_{A0-I}$ has$_{P-B}$ voted$_{P-I}$ on$_{P-I}$ last$_{A2-B}$ June$_{A2-I}$

Table 2: Example sentences and respective Open IE extractions. The first line in each example presents the input pairs $(S,\ p)$, where S is the input sentence, and the predicate head, p, is denoted with an underline. Below the inputs we present the corresponding Open IE extractions. The corresponding encodings are presented below the dashed lines, where subscripts indicate the associated BIO label. Demonstrating: (a) the encoding of a multi-word predicate, (b) several arguments collapsed into the same ***A0*** argument position, (c) argument position deviating from the sentence ordering.

$(w_1, \ldots, w_n)$, a tuple consists of $(\mathbf{x_1}, \ldots, \mathbf{x_m})$, where each x_i is a contiguous subspan of S. One of the x_i is distinguished as the *predicate* (marked in bold in Figure 1), while the other spans are considered its arguments. Following this definition, we reformulate Open IE as a sequence labeling task, using a custom BIO[3] (Ramshaw and Marcus, 1995; Sang and Veenstra, 1999) scheme adapted from recent deep SRL models (He et al., 2017).

In our formulation, the set of Open IE tuples for a sentence S are grouped by predicate head-word p, as shown in Table 2. For instance, example (b) lists two tuples for the predicate head "born", which is underlined in the sentence. Grouping tuples this way allows us to run the model once for each predicate head, and accumulate the predictions across predicates to produce the final set of extractions.

Open IE tuples deviate from SRL predicate-argument structures in two major respects. First, while SRL generally deals with single-word predicates, Open IE uses multi-word predicates that often incorporate modals and embedded predicates. For example, the first tuple in the table includes the embedded predicate **claimed that he won**. Second, Open IE generates multiple extractions from a single predicate in certain syntactic constructions (e.g., apposition, co-ordination or coreference). For instance, example (b) repeats the predicate **was born in** for the two components of the apposition *Barack Obama, a former U.S. president*.

To model these unique challenges, we introduce a custom BIO tagging scheme, shown in Table 2 below the dashed lines. Predicates are encoded using the P label type, while arguments are represented using Ai labels, where i represents the argument's position within the extracted Open IE tuple. While softer than SRL's predicate-specific argument roles (e.g., ARG0), these argument positions also capture semantic information because they are arranged such that the tuple can be naturally read as a standalone statement, regardless of the complications of the source text's syntax (such as reorderings and long-distance dependencies). For instance, in the last example in Table 2, the order of the arguments in the Open IE tuple deviates from the ordering in the original sentence due to a relative clause construction (headed by the word *Brexit*).

Finally, multiple extractions per predicate are encoded by assigning the same argument index to all arguments appearing in that position across all of the predicate's extractions. For example, note that the ***A0*** argument label appears twice for the apposition in example (b). To reconstruct the extractions from the BIO labels, we produce an extraction for every possible way of choosing one argument for each index.

[3]Beginning, Inside, Outside

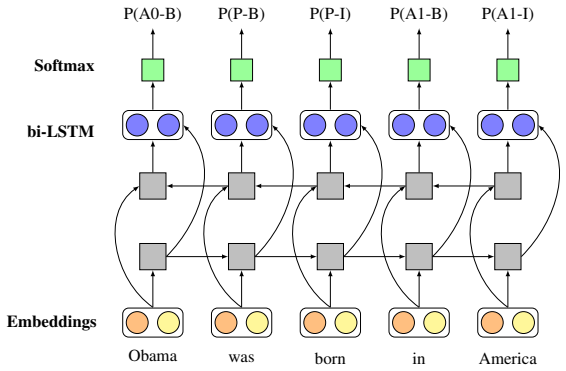


Figure 2: RNN model architecture. Orange circles represent current word features: embedding for word and part of speech. Yellow circles represent predicate features, duplicated and concatenated to all other word features.

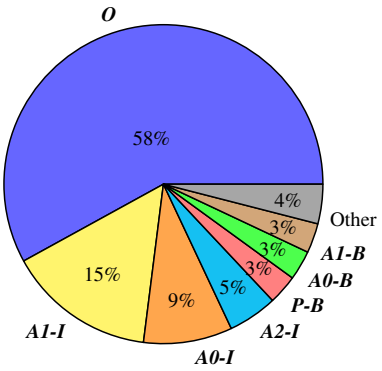


Figure 3: Word label distribution in the training set.

4 Supervised Open IE Model

Our model, named RnnOIE, is a bi-LSTM transducer, inspired by the state of the art deep learning approach for SRL suggested by Zhou and Xu (2015) and He et al. (2017). The architecture is shown in Figure 2.

Given an input instance of the form $(S,\ p)$, where S is the input sentence, and p is the word index of the predicate's syntactic head, we extract a feature vector $feat$ for every word $w_i \in S$:

$$feat(w_i,\ p) = emb(w_i) \oplus emb(pos(w_i)) \oplus \\ \oplus\ emb(w_p) \oplus emb(pos(w_p))$$

Here, $emb(w)$ is a d-dimensional word embedding, $emb(pos(w))$ is a 5-dimensional embedding of w's part of speech, and $\oplus$ denotes concatenation. We duplicate the predicate head's features on all words to allow the model to more directly access this information as it makes predicate-specific word label predictions.

The features are fed into a bi-directional deep LSTM transducer (Graves, 2012) which computes contextualized output embeddings. The outputs are used in softmaxes for each word, producing independent probability distributions over possible BIO tags.

The model is trained with gold predicate heads, using a per-word maximum likelihood objective. Figure 3 depicts the overall word label distribution within the training set. The large percentage of ***O*** labels demonstrates Open IE's tendency to shorten arguments, compared to SRL which considers full syntactic constitutes as arguments.

Inference At inference time, we first identify all verbs and nominal predicates in the sentence as candidate predicate heads. We use a Part Of Speech (POS) tagger to identify verbs, and Catvar's subcategorization frames (Habash and Dorr, 2003) for nominalizations, identifying nouns which share the same frame with a verbal equivalent (e.g., *acquisition* with *acquire*). We then generate an input instance for each candidate predicate head. For each instance, we tag each word with its most likely BIO label under the model, and reconstruct Open IE tuples from the resulting sequence according to the method described in Section 3, with the exception that we ignore malformed spans (i.e., if an ***A0-I*** label is not preceded by ***A0-I*** or ***A0-B***, we treat it as ***O***).

Assigning extraction confidence It is beneficial for an Open IE system to associate a confidence value with each predicted extraction to allow for tuning its precision-recall tradeoff. Our model does not directly produce confidence values for extractions, but it does assign probabilities to each BIO label that it predicts. We experimented with several heuristics to combine these predictions to an extraction-level confidence metric. The best performance on the development set was achieved by multiplying the probabilities of the B and I labels participating in the extraction.[4] This metric prefers shorter extractions, which correlates well with the requirements of Open IE (Bhutani et al., 2016).

Implementation details We implemented the model using the Keras framework (Chollet, 2015) with TensorFlow backend (Abadi et al., 2015). All

[4] We also tried taking the maximum or minimum observed single word-label probability.

Mercury filling, particularly prevalent in the USA, was banned in the EU, partly because it causes antibiotic resistance.			
Predicate	QA-SRL	QAMR	Open IE
made	-	What is the **filling made of**? mercury	-
prevalent	-	What was **particularly prevalent in the USA**? mercury filling	(mercury filling; **particularly prevalent**; in the USA)
banned	What was **banned**? mercury filling Where was something **banned**? the EU Why was something **banned**? partly because it causes antibiotic resistance	What was **banned in the EU partly because it causes antibiotic resistance**? mercury filling	(mercury filling; **was banned**; in the EU; partly because it causes antibiotic resistance)
causes	What **caused** something? mercury filling What did something **cause**? antibiotic resistance	What did **mercury filling cause**? antibiotic resistance	(mercury filling; **caused**; antibiotic resistance)

Table 3: Comparison of QA-SRL, QAMR, and desired Open IE annotations for an example sentence, adapted from the QAMR corpus.

hyperparameters were tuned on the `OIE2016` development set. The bi-LSTM transducer has 3 layers and each LSTM cell uses 128 hidden units and a linear rectifier (ReLU) (Nair and Hinton, 2010) activation function. The model was trained for 100 epochs in mini batches of 50 samples, with 10% word-level dropout. The word-embeddings were initialized using the GloVe 300-dimensions pre-trained embeddings (Pennington et al., 2014) and were kept fixed during training. The part of speech embeddings were randomly initialized and updated during training. Finally, we use the average perceptron part-of-speech tagger (as implemented in spaCy[5]) to predict parts of speech for input features and verb predicate identification.

5 Open IE from QAMR

This section describes our approach for automatically extracting Open IE tuples from QAMR (Michael et al., 2018), a recent extension of QA-SRL. While QA-SRL uses question templates centered on verbs, QAMR annotates free-form questions over arbitrary predicate types. The QAMR corpus consists of annotations over $5,000$ sentences. By extending the `OIE2016` training set with extractions from QAMR, we more than triple the available amount of training data.

The extraction algorithm, as well as the resulting corpus, are made publicly available at `https://github.com/gabrielStanovsky/supervised-oie`.

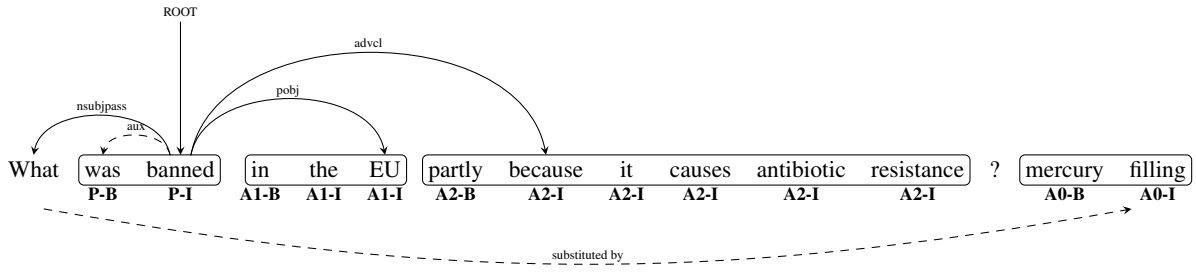
[5] `https://spacy.io`

5.1 The QAMR Corpus

Question-Answer Meaning Representation, or QAMR (Michael et al., 2018), was recently proposed as an extension of QA-SRL. Like QA-SRL, QAMR represents predicate-argument structure with a set of question-answer pairs about a sentence, where each answer is a span from the sentence. However, while QA-SRL restricts questions to fit into a particular verb-centric template, QAMR is more general, allowing any natural language question that begins with a *wh*-word and contains at least one word from the sentence. This allows QAMR to express richer, more complex relations. Consider, for example, the first two entries for QAMR in Table 3. The first explicates the implicit relation **made of** from the noun compound *mercury filling*, and the second identifies the adjectival predicate **prevalent**. Neither of these can be represented in QA-SRL.

5.2 Extraction Algorithm

While QAMR's broader scope presents an opportunity to vastly increase the number and coverage of annotated Open IE tuples, it also poses additional challenges for the extraction algorithm. The free-form nature of QAMR questions means that some are over-expressive for Open IE, while in many other cases it is less obvious how to extract a predicate and a list of arguments from a question-answer pair.



(mercury filling; **was banned**; in the EU; partly because it causes antibiotic resistance)

Figure 4: A QAMR (top) to Open IE (bottom) conversion example. The BIO labels for our encoding of the Open IE tuple appear below the text. The root of the question's dependency tree is the predicate, while its syntactic constituents are the arguments. The answer appears as the first argument of the Open IE tuple due to the passive construction.

Over-expressiveness The QAMR formalism allows many constructions that diverge from Open IE extractions, which generally are drawn verbatim from the source text. For example, the predicate **made** is introduced in the QAMR for the sentence in Table 3, despite not appearing in the sentence. To circumvent this issue, we filter out questions which: (1) introduce new content words,[6] (2) have more than one *wh*-word, (3) do not start with *who*, *what*, *when* or *where*, or (4) ask *what did X do?*, delegating the predicate to the answer.

Detecting predicates and arguments While a QA-SRL question has a designated predicate and a single argument as the answer, in QAMR, the predicate can appear anywhere in the question and its arguments are spread between the question and answer. For example, extracting an Open IE tuple for the predicate **banned** in Table 3 requires decoupling the predicate and its arguments *in the EU* and *partly because it causes antibiotic resistance*. Our solution to this problem is illustrated in Figure 4. We first run each question through a syntactic dependency parser. We then identify the predicate as the head of the question's dependency tree extended to include all dependents with an auxiliary relation (e.g., *aux*, *neg*, or *prt*). The predicted arguments are the predicate's constituent argument subtrees, while the answer to the question replaces the subtree headed by the wh-word. Finally, we employ similar heuristics to those used converting verbal QA-SRL to Open IE to find the correct argument position for the answer (Stanovsky and Dagan, 2016). For example, the passive construction in Figure 4 implies that the answer should be placed in the first argument position, while the existence of a prepositional object in, e.g., *What did he put **on the table**?* signals that the answer should be placed in the second argument position.

[6]We do not count inflected forms of verbs from the sentence, such as *caused* in the last entry of the table, as new words.

QAMR Open IE Tuples
(The treaty of Brussels; **was signed**; on 17 March 1948; by Belgium, the Netherlands, Luxembourg, France, and the UK)
(The treaty of Brussels; **is the precursor to**; the NATO agreement)
(The scope of publishing; **has expanded to include**; websites, blogs, and the like.)

Table 4: Tuples from the All Words Open IE Corpus, exemplifying n-ary extractions (top example), non-verbal predicates (middle), and multi-word predicates (bottom).

5.3 The All Words Open IE Corpus

As described by Michael et al. (2018), QAMR annotations were gathered via crowdsourcing in a two-stage pipeline over Wikipedia and Wikinews text. We use the training partition of the QAMR dataset, which consists of 51,063 QA pairs over 3,938 sentences. Our filtering and conversion from the QAMR corpus yields 12,952 Open IE tuples (2.5 times the size of `OIE2016`'s training corpus), composed of 7,470 (58%) verbal predicates, 4,952 (38%) nominal predicates, and 530 (4%) adjectival predicates. See Table 4 for example tuples, taken from the converted corpus.

Examining the results, we found that they are not accurate enough to constitute a gold test cor-

pus, partly because some relations were missed by the annotators of QAMR and partly because of noise introduced in the automatic extraction process. Instead, we use this corpus to extend the train partition of OIE2016. In the following section, we show its usefulness in significantly improving the precision and recall of our Open IE model.

6 Evaluation

We evaluate the performance of our model on the four test sets discussed in Section 2.

6.1 Experimental Setup

Metrics We evaluate each system according to three metrics. First, as is typical for Open IE, we compute a *precision-recall (PR) curve* by evaluating the systems' performance at different extraction confidence thresholds. This curve is useful for downstream applications which can set the threshold according to their specific needs (i.e., recall oriented versus precision oriented). Second, we compute the *area under the PR curve (AUC)* as a scalar measurement of the overall system performance. Finally, for each system, we report a single F1 score using a confidence threshold optimized on the development set. This can serve as a preset threshold for out-of-the-box use.

Matching function Similar to other cases in NLP, we would like to allow some variability in the predicted tuples. For example, for the sentence *The sheriff standing against the wall spoke in a very soft voice* we would want to treat both (The Sheriff; **spoke**; in a soft voice) and (The sheriff standing against the wall; **spoke**; in a very soft voice) as acceptable extractions. To that end, we follow He et al. (2015) which judge an argument as correct if and only if it includes the syntactic head of the gold argument (and similarly for predicates). For OIE2016, we use the available Penn Treebank gold syntactic trees (Marcus et al., 1993), while for the other test sets, we use predicted trees instead. While this metric may sometimes be too lenient, it does allow a more balanced and fair comparison between systems which can make different, but equally valid, span boundary decisions.

Baselines We compare our model (RnnOIE) against the top-performing systems of those evaluated most recently in Stanovsky and Dagan (2016) and in Schneider et al. (2017): Open IE4,[7] ClausIE (Del Corro and Gemulla, 2013), and PropS (Stanovsky et al., 2016).

6.2 Results

Table 5 reports the AUC and F1 scores of all of the systems on the 4 test sets. In addition, the PR curves for the two largest test sets (OIE2016 and WEB) are depicted in Figures 5a and 5b. We report results for two versions of our model: one trained on the OIE2016 training set containing only verbal predicates (*RnnOIE-verb*), and another on the extended training set that includes the automatic conversion of QAMR outlined in Section 5 (*RnnOIE-aw*).

Overall, RnnOIE-aw outperforms the other systems across the datasets. On the larger test sets (OIE2016 and WEB) it provides the best performance in terms of AUC and F1, with a superior precision-recall curve. On each of the smaller test sets, it performs best on one metric and competitively on the other.

Furthermore, on all of the test sets, extending the training set significantly improves our model's performance, showing that it benefits from the additional data and types of predicates available in the QAMR dataset. While this is most notable in the test sets which include nominalizations (WEB, NYT, and PENN), it also improves the performance on OIE2016, which is composed solely of verb predicates.

6.3 Performance Analysis

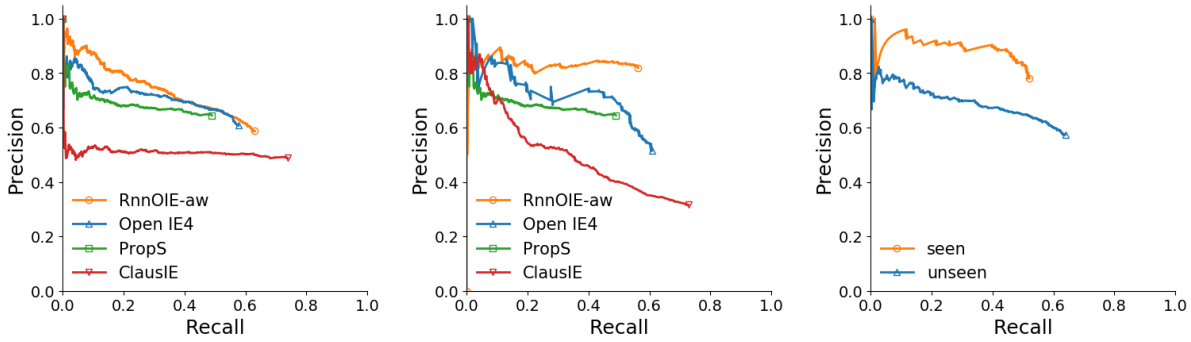
In our analysis, we find that RnnOIE generalizes to unseen predicates, produces more and shorter arguments on average than are in the gold extractions, and, like all of the systems we tested, struggles with nominal predicates.

Unseen predicates We split the propositions in the gold and predicted OIE2016 test set into two partitions, *seen* and *unseen*, based on whether the predicate head's lemma appears in the training set. The *unseen* part contains 145 unique predicate lemmas in 148 extractions, making up 24% out of the 590 unique predicate lemmas and 7% out of the 1993 total extractions in the test set. We then evaluated RnnOIE-aw on each part separately. The resulting PR curves (Figure 5c) depict overall good performance also on the unseen part, competitive with previous Open IE systems.

[7] https://github.com/dair-iitd/OpenIE-standalone

	OIE2016		WEB		NYT		PENN	
	AUC	F1 (P, R)	AUC	F1 (P, R)	AUC	F1 (P, R)	AUC	F1 (P, R)
ClausIE	.38	.59 (.49, .74)	.40	.45 (.39,.53)	.23	.30 (.24, .39)	**.28**	.34 (.24, .61)
PropS	.34	.56 (.64, .49)	.45	.59 (.44, .89)	.22	.37 (.25, .77)	**.28**	.39 (.26, .81)
Open IE4	.42	.60 (.64, .56)	.45	.56 (.63, .50)	.24	**.38 (.26, .74)**	**.28**	.43 (.37, .50)
RnnOIE-verb	.45	.59 (.57, .62)	.23	.46 (.38, .58)	.09	.25 (.20,.33)	.21	.38 (.35, .40)
RnnOIE-aw	**.48**	**.62 (.61, .64)**	**.47**	**.67 (.83, .56)**	**.25**	.35 (.24,.67)	.26	**.44 (.31,.75)**

Table 5: Performance of the OIE extractors on our test sets. Each system is tested in terms of Area Under the PR Curve (AUC), and F1 (precision and recall in parenthesis).



(a) OIE2016 (b) WEB (c) Seen vs. unseen

Figure 5: Precision-recall curves of the different OIE systems on OIE2016 (5a), WEB (5b) and seen vs. unseen predicates in RnnOIE-aw on OIE2016 (5c). See details in Section 6.

System	# Tuples	Args/Prop	Words/Arg
Gold	1730	2.45	5.38
ClausIE	2768	2.00	5.78
PropS	1551	2.68	5.8
Open IE4	1793	3.07	4.55
RnnOIE-aw	1993	3.19	4.68

Table 6: Output statistics of the different systems on OIE2016, versus the gold data.

This indicates that our model generalizes beyond memorization of specific predicate templates.

Argument length and number In Table 6 we compare statistics on the the outputs of the Open IE systems on OIE2016 and the gold data. The best performing systems, RnnOIE and OpenIE4, tend to produce more arguments, and each argument tends to be shorter on average, in comparison to other systems and gold.

Runtime analysis Since Open IE is intended to be usable at web scale, we timed the different Open IE systems on a batch of 3200 sentences from OIE2016, running on a Xeon 2.3GHz CPU. The results are presented in Table 8.[8] We find that our system fares well, processing only 12% fewer sentences per second than the fastest system, Open IE 4.0. Further, while these numbers are reported on CPU for the sake of fair comparison, running our neural model on a GPU (NVIDIA GeForce GTX 1080 Ti) boosts speed by a factor of more than 10 (149.25 sentences per second, on average).

Error analysis All of the systems still lack in recall across all tested corpora. We examined a random sample of 100 recall errors shared by all of the extractors across the tested datasets and found several common error types, shown in Table 7. Notably, noun and nominalized predicates still pose a challenge, appearing in 51% of the recall errors (whereas they make up 24% of all extractions). 19% of the examined errors required some

[8]PropS and ClausIE's relatively slow performance is in part due to their hard-coded use of the Stanford parser, which took on average 0.2 seconds per sentence. Using a faster parser (e.g., spaCy) may improve this performance.

Phenomenon	%	Example (sentence / gold tuple)
Noun	38	*Andre Agassi did a similar thing in his **hometown** of Las Vegas a few years ago.* (Andre Agassi; **hometown**; Las Vegas)
Sent.-level Inference	19	*John Steinbeck also **earned** alot of awards, one being the Pulitzer Prize in 1940.* (John Steinbeck; **earned**; Pulitzer Prize)
Long sentence	14	*"I don't see any radical change for our company", said David Westin, the **president** of production for Capital CitiesABC Inc. "But in the next year or so, I would expect you might see all sorts of new deals between networks and studios, joint ventures and creative financing of programs."* (David Westin; **president**; Capital Cities/ABC Inc.)
Nominalization	13	*We first heard about this when the Google-Youtube **acquisition** news broke, and wrote briefly about it here* (Google; **acquisition**; Youtube)
Noisy Informal	13	*But who knows, with Google's "**owning**" of YouTube now ..they are now in the 'media' department with that deal... so who knows if they will move on to music stuff next :P* (Google; **owning**; YouTube)
PP-attachment	10	*The novelist Franz Kafka was **born** of Jewish parentage in Prague in 1883.* (Franz Kafka; **born**; Prague)

Table 7: Analysis of frequently-occuring recall errors for all tested systems on a random sample of 100 sentences. For each phenomenon we list the percentage of sentences in which it occurs (possibly overlapping with other phenomena), and a protoypical example, taken from the WEB corpus.

	ClausIE	PropS	Open IE4	RnnOIE
CPU	4.07	4.59	**15.38**	13.51
GPU	—	—	—	**149.25**

Table 8: Runtime analysis, measured in sentences per second, of the different systems on 3200 sentences from the `OIE2016` corpus on Xeon 2.3GHz CPU (top) and on an NVIDIA GeForce GTX 1080 Ti (bottom). Baselines were only run on CPU as they are currently not optimized for GPU.

form of sentence level inference, such as determining event factuality or pronoun resolution. 14% of the errors involved long sentences with over 40 words (where the average word count per sentence is 29.4).

7 Conclusions and Future Work

We present a supervised model for Open IE, formulating it as a sequence tagging problem and applying a bi-LSTM transducer to produce a state-of-the-art Open IE system. Along the way, we address several task-specific challenges, including the BIO encoding of predicates with multiple extractions and confidence estimation in our sequence tagging model. To train the system, we leverage a recently published large scale corpus for Open IE (Stanovsky and Dagan, 2016), and further extend it using a novel conversion of the QAMR corpus (Michael et al., 2018), which covers a wider range of predicates.

In addition to these contributions, this work shows that Open IE can greatly benefit from future research into the QA-SRL paradigm. For example, Open IE would directly benefit from an automatic QA-SRL extractor, while a more exhaustive or extensive annotation of QAMR would improve Open IE's performance on a wider range of predicates.

Acknowledgments

This work was supported in part by grants from the MAGNET program of the Israeli Office of the Chief Scientist (OCS); the German Research Foundation through the German-Israeli Project Cooperation (DIP, grant DA 1600/1-1); the Israel Science Foundation (grant No. 1157/16); the US NSF (IIS1252835,IIS-1562364); and an Allen Distinguished Investigator Award.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. `http://tensorflow.org/`.

Alan Akbik and Alexander Löser. 2012. Kraken: N-ary facts in open information extraction. In *NAACL-HLT 2012: Proceedings of the The Knowledge Extraction Workshop*.

Gabor Angeli, Melvin Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015)*.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*. pages 2670–2676.

Jonathan Berant, Ido Dagan, and Jacob Goldberger. 2011. Global learning of typed entailment rules. In *Proceedings of ACL*. Portland, OR.

Nikita Bhutani, HV Jagadish, and Dragomir Radev. 2016. Nested propositions in open information extraction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Austin, Texas, pages 55–64.

Franois Chollet. 2015. Keras. `https://github.com/fchollet/keras`.

Filipe de Sá Mesquita, Jordan Schmidek, and Denilson Barbosa. 2013. Effectiveness and efficiency of open relation extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 447–457.

Luciano Del Corro and Rainer Gemulla. 2013. Clausie: clause-based open information extraction. In *Proceedings of the 22nd international conference on World Wide Web*. International World Wide Web Conferences Steering Committee, pages 355–366.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1535–1545.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*. pages 1156–1165.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711* .

Nizar Habash and Bonnie J. Dorr. 2003. A categorial variation database for english. In *HLT-NAACL*.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and whats next. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

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

Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, Jeju Island, Korea, pages 523–534. `http://www.aclweb.org/anthology/D12-1048`.

Julian Michael, Gabriel Stanovsky, Luheng He, Ido Dagan, and Luke Zettlemoyer. 2018. Crowdsourcing question-answer meaning representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*. pages 807–814.

Harinder Pal and Mausam. 2016. Demonyms and compound relational nouns in nominal open ie. In *AKBC@NAACL-HLT*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. volume 14, pages 1532–1543.

Lance A Ramshaw and Mitchell P Marcus. 1995. Text chunking using transformation-based learning. *arXiv preprint cmp-lg/9505040* .

Erik F Sang and Jorn Veenstra. 1999. Representing text chunks. In *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 173–179.

Rudolf Schneider, Tom Oberhauser, Tobias Klatt, Felix A. Gers, and Alexander Loser. 2017. Analysing errors of open information extraction systems. *CoRR* abs/1707.07499.

Gabriel Stanovsky and Ido Dagan. 2016. Creating a large benchmark for open information extraction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Austin, Texas.

Gabriel Stanovsky, Jessica Ficler, Ido Dagan, and Yoav Goldberg. 2016. Getting more out of syntax with props. *arXiv preprint* .

Fei Wu and Daniel S. Weld. 2010. Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Uppsala, Sweden, pages 118–127. `http://www.aclweb.org/anthology/P10-1013`.

Ying Xu, Mi-Young Kim, Kevin Quinn, Randy Goebel, and Denilson Barbosa. 2013. Open information extraction with tree kernels. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Atlanta, Georgia, pages 868–877. `http://www.aclweb.org/anthology/N13-1107`.

Mohamed Yahya, Steven Euijong Whang, Rahul Gupta, and Alon Y. Halevy. 2014. Renoun: Fact extraction for nominal attributes. In *EMNLP*.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*. pages 1127–1137.

Embedding Syntax and Semantics of Prepositions via Tensor Decomposition

Hongyu Gong, Suma Bhat, Pramod Viswanath
{hgong6, spbhat2, pramodv}@illinois.edu
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign, USA

Abstract

Prepositions are among the most frequent words in English and play complex roles in the syntax and semantics of sentences. Not surprisingly, they pose well-known difficulties in automatic processing of sentences (prepositional attachment ambiguities and idiosyncratic uses in phrases). Existing methods on preposition representation treat prepositions no different from content words (e.g., word2vec and GloVe). In addition, recent studies aiming at solving prepositional attachment and preposition selection problems depend heavily on external linguistic resources and use dataset-specific word representations. In this paper we use *word-triple* counts (one of the triples being a preposition) to capture a preposition's interaction with its attachment and complement. We then derive preposition embeddings via tensor decomposition on a large unlabeled corpus. We reveal a new geometry involving Hadamard products and empirically demonstrate its utility in paraphrasing phrasal verbs. Furthermore, our preposition embeddings are used as simple features in two challenging downstream tasks: preposition selection and prepositional attachment disambiguation. We achieve results comparable to or better than the state-of-the-art on multiple standardized datasets.

1 Introduction

Prepositions are a linguistically closed class comprising some of the most frequent words; they play an important role in the English language since they encode rich syntactic and semantic information. Many preposition-related tasks are challenging in computational linguistics because of their polysemous nature and flexible usage patterns. An accurate understanding and representation of prepositions' linguistic role is key to several important NLP tasks such as grammatical error correction and prepositional phrase attachment. A first-order approach is to represent prepositions as real-valued vectors via word embeddings such as word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014).

Word embeddings have brought a renaissance in NLP research; they have been very successful in capturing word similarities as well as analogies (both syntactic and semantic) and are now mainstream in nearly all downstream NLP tasks (such as question-answering (Chen et al., 2017)). Despite this success, available literature does not highlight any specific properties of word embeddings of prepositions. Indeed, many of the common prepositions have very similar vector representations as shown in Table 1 for preposition vectors trained using word2vec and GloVe (Tensor embedding is our proposed representation for prepositions). While this suggests that using available representations for prepositions diminishes the distinguishing aspect between prepositions, one could hypothesize that this is primarily because standard word embedding algorithms treat prepositions no different from other content words such as verbs and nouns, i.e., embeddings are created based on co-occurrences with other words. However, prepositions are very frequent and co-occur with nearly all words, which means that their co-occurrence ought to be treated differently.

Preposition pair	Word2vec	GloVe	Tensor
(above, below)	0.85	0.78	0.22
(above, beneath)	0.40	0.45	0.15
(after, before)	0.83	0.70	0.44
(after, during)	0.56	0.42	0.16
(amid, despite)	0.47	0.37	0.12
(amongst, besides)	0.46	0.37	0.21
(beneath, inside)	0.55	0.47	0.29

Table 1: Cosine similarity between pairs of centered prepositions using some word embeddings

Modern descriptive linguistic theory proposes

Proceedings of NAACL-HLT 2018, pages 896–906
New Orleans, Louisiana, June 1 - 6, 2018.

to understand a preposition via its interactions with *both* the head it attaches to (termed *head*) and its complement (Huddleston, 1984; DeCarrico, 2000). This theory naturally suggests that one should count co-occurrences of a given preposition with *pairs* of neighboring words. One way of achieving this would be by considering a *tensor* of triples ($word_1$, $word_2$, preposition), where we do not restrict $word_1$ and $word_2$ to be the head and complement words; instead we model a preposition's interaction with *all* pairs of neighboring words via a *slice* of a tensor X, where the slice is populated by word co-occurrences restricted to a context window of the specific preposition. Thus, the tensor dimension is $N \times N \times K$ where N is the vocabulary size and K is the number of prepositions; since $K \approx 50$, we note that $N \gg K$.

Using such a representation, we notice that the resulting tensor is low rank and use it to extract embeddings for both preposition and non-preposition words. In doing so, we use a combination of standard ideas from word representations (such as weighted spectral decomposition as in GloVe (Pennington et al., 2014)) and tensor decompositions (alternating least squares (ALS) methods (Sharan and Valiant, 2017)). We find that the preposition embeddings extracted in this manner are discriminative (see the preposition similarity of the tensor embedding in Table 1). Note that the smaller the cosine similarity is, the more distinct the representations are from each other. We demonstrate that the resulting preposition representation captures the core linguistic properties of prepositions–the attachment and the complement properties. Using both intrinsic evaluations and downstream tasks, we show this by providing new state-of-the-art results on well-known NLP tasks involving prepositions.

Intrinsic evaluations: We show that the *Hadamard* product of the embeddings of a verb and a preposition that together make a phrasal verb, closely approximates the representation of this phrasal verb's paraphrase as a single verb. Example: $v_{\text{made}} \odot v_{\text{from}} \approx v_{\text{produced}} \odot v$, where $\odot$ represents the Hadamard product (i.e., element-wise multiplication) of two vectors and v is a constant vector (not associated with a specific word and is defined later); this approximation validates that prepositional semantics are appropriately encoded into their trained embeddings. We provide a mathematical interpretation for this new geometry while empirically demonstrating the paraphrasing of compositional phrasal verbs.

Extrinsic evaluations: Our preposition embeddings are used as features for a simple classifier in two well-known challenging downstream NLP classification tasks. In both tasks, we perform as well as or strictly better than the state-of-the-art on multiple standardized datasets.

Preposition selection: While the context in which a preposition occurs governs the choice of the preposition, the specific preposition by itself significantly influences the semantics of the context in which it occurs. Furthermore, the choice of the right preposition for a given context can be very subtle. This idiosyncratic behavior of prepositions is the reason behind preposition errors being one of the most frequent error types made by second language English speakers (Leacock et al., 2010)). We demonstrate the utility of the preposition embeddings in the preposition selection task, which is to choose the correct preposition to a given sentence. We show this for a large set of contexts–$7,000$ combined instances from the CoNLL-2013 and the SE datasets (Prokofyev et al., 2014). Our approach achieves 6% and 2% absolute improvement over the previous state-of-the-art results on the respective datasets.

Prepositional phrase attachment disambiguation: Prepositional phrase attachment is a common cause of structural ambiguity in natural language. In the sentence "Pierre Vinken *joined the board as a voting member*", the prepositional phrase "as a voting member" can attach to either "joined" (the VP) or "the board" (the NP); in this case the VP attachment is correct. Despite being extensively studied over decades, prepositional attachment continues to be a major source of syntactic parsing errors (Brill and Resnik, 1994; Kummerfeld et al., 2012; de Kok and Hinrichs, 2016). We use our prepositional representations as simple features to a standard classifier on this task. Our approach tested on a widely studied standard dataset (Belinkov et al., 2015) achieves 89% accuracy and compares favorably with the state-of-the art. It is noteworthy that while the state-of-the-art results are obtained with significant linguistic resources, including syntactic parsers and the WordNet, our approach achieves a comparable performance without relying on such resources.

We emphasize two aspects of our contributions: (1) Word representations trained via *pairwise*

word counts are previously shown to capture much of the benefits of the unlabeled sentence-data; example: (Sharan and Valiant, 2017) reports that their word representations via word-triple counts are better than others, but still significantly worse than regular word2vec representations. One of our main observations is that considering word-triple counts makes most (linguistic) sense when one of the words is a preposition. Furthermore, the sparsity of the corresponding tensor is no worse than the sparsity of the regular word co-occurrence matrix (since prepositions are so frequent and co-occur with essentially every word). Taken together, these two points strongly suggest the benefits of tensor representations in the context for prepositions.

(2) The word and preposition representations via tensor decomposition are simple features leading to a standard classifier. In particular, we do not use dependency parsing (which many prior methods have relied on) or handcrafted features (Prokofyev et al., 2014) or train task-specific representations on the annotated training dataset (Belinkov et al., 2015). The simplicity of our approach, combined with the strong empirical results, lends credence to the strength of the prepositional representations found via tensor decompositions.

2 Method

We begin with a description of how the tensor with triples (word, word, preposition) is formed and empirically show that its slices are low-rank. Next, we derive low dimensional vector representations for words and prepositions via appropriate tensor decomposition methods.

Tensor creation: Suppose that K prepositions are in the preposition set $P = \{p_1, \ldots, p_K\}$; here K is 49 in our preposition selection task, and 76 in the attachment disambiguation task. We limited the number of prepositions to what was needed in the dataset. The vocabulary, the set of all words excluding the prepositions, contains N words, $V = \{w_1, \ldots, w_N\}$, and $N \approx 1M$. We generate a third order tensor $\mathbf{X}_{N \times N \times (K+1)}$ from the WikiCorpus (Al-Rfou et al., 2013) as follows. We say two words co-occur if they appear within a distance t of each other in a sentence. For $k \leq K$, the entry $\mathbf{X}_{ijk}$ is the number of occurrences where word w_i co-occurs with preposition p_k, and w_j also co-occurs with preposition p_k in the same sentence, and this is counted across all sentences in the WikiCorpus. For $0 \leq k \leq K$, $\mathbf{X}[:, :, k]$ is a matrix of the count of the word pairs that co-occur with the preposition k, and we call such a matrix a slice.

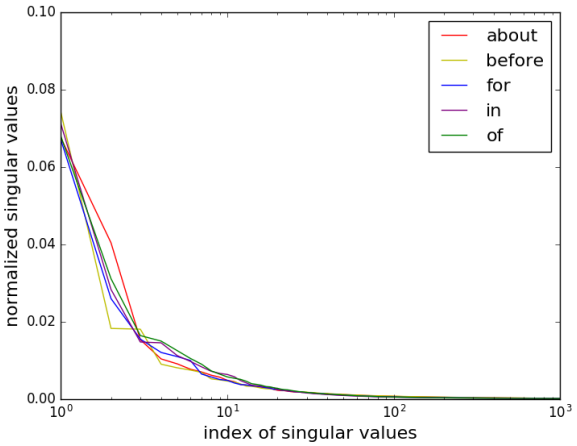


Figure 1: Decaying normalized singular values of slices.

Here we use a window of size $t = 3$. While prepositions co-occur with many words, there are also a number of other words which do not occur in the context of any preposition. In order to make the maximal use of the data, we add an extra slice $\mathbf{X}[:, :, K+1]$, where the entry $\mathbf{X}_{ij(K+1)}$ is the number of occurrences where w_i co-occurs with w_j (within distance $2t = 6$) but at least one of them is not within a distance of t of any preposition. Note that the preposition window of 3 is smaller than the word window of 6, since it is known that the interaction between prepositions and neighboring words usually weakens more sharply with distance when compared to that of content words (Hassani and Lee, 2017).

Empirical properties of X: We find that the tensor X is very sparse – only 1% of the tensor elements are non-zero. Furthermore, $\log(1 + \mathbf{X}[:, :, k])$ is low-rank (here the logarithm is applied component-wise to every entry of the tensor slice). Towards seeing this, we choose slices corresponding to the prepositions "about", "before","for", "in" and "of", and plot their normalized singular values in Figure 1. We see that the singular values decay dramatically, suggesting the low-rank structure in each slice.

Tensor decomposition: We combine standard ideas from word embedding algorithms and tensor decomposition algorithms to arrive at the low-rank approximation to the tensor $\log(1 + \mathbf{X})$. In

particular, we consider two separate methods:

1. *Alternating Least Squares (ALS)*. A generic method to decompose a tensor into its modes is via the CANDECOMP/PARAFAC (CP) decomposition (Kolda and Bader, 2009). The tensor $\log(1 + \mathbf{X})$ is decomposed into three modes: $\mathbf{U}_{d\times N}$, $\mathbf{W}_{d\times N}$ and $\mathbf{Q}_{d\times(K+1)}$, based on the solutions to the optimization problem (1). Here $\mathbf{u}_i$, $\mathbf{w}_i$ and $\mathbf{q}_i$ are the i-th column of U, W and Q, respectively.

$$L = \min_{\mathbf{U},\mathbf{W},\mathbf{Q}} \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{K+1} \big(\langle \mathbf{u}_i, \mathbf{w}_j, \mathbf{q}_k \rangle - \log(1 + \mathbf{X}_{ijk}) \big)^2, \quad (1)$$

where $\langle \mathbf{a}, \mathbf{b}, \mathbf{c} \rangle = \mathbf{1}^t(\mathbf{a} \odot \mathbf{b} \odot \mathbf{c})$ is the inner product of three vectors $\mathbf{a}, \mathbf{b}$ and $\mathbf{c}$. Here $\mathbf{1}$ is the column vector of all ones and $\odot$ refers to the Hadamard product. We can interpret the columns of U as the word representations and the columns of Q as the preposition representations, each of dimension d (equal to 200 in this paper). There are several algorithmic solutions to this optimization problem in the literature, most of which are based on alternating least squares methods (Kolda and Bader, 2009; Comon et al., 2009; Anandkumar et al., 2014) and we employ a recent one named Orth-ALS (Sharan and Valiant, 2017) in this paper. Orth-ALS periodically orthogonalizes the decomposed components while fixing two modes and updating the remaining one. It is supported by theoretical guarantees and empirically outperforms standard ALS methods in different applications.

2. *Weighted Decomposition (WD)*: Based on ideas from the literature on word embedding algorithms, we also consider weighting different elements of the tensors differently in order to reduce the effect of the large dynamic range of the tensor values. Specifically, we employ the GloVe objective function to our tensor model and minimize the objective function (2):

$$L_{\text{weighted}} = \min_{\mathbf{U},\mathbf{W},\mathbf{Q}} \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{K+1} \omega_{ijk} \big(\langle \mathbf{u}_i, \mathbf{w}_j, \mathbf{q}_k \rangle + b_{Ui} + b_{Wj} + b_{Qk} - \log(\mathbf{X}_{ijk} + 1) \big)^2, \quad (2)$$

where b_{Ui} is the scalar bias for the word i in the matrix U. Similarly, b_{Wj} is the bias for the word j in the matrix W, and b_{Qk} for preposition k in the matrix Q. Bias terms are learned in such a way as to minimize the loss function. Here ω_{ijk} is the weight assigned to each tensor element X_{ijk}, and we use the weighting proposed by GloVe:

$$\omega_{ijk} = \min\left(\left(\frac{\mathbf{X}_{ijk}}{x_{\max}} \right)^{\alpha}, 1 \right).$$

We set the hyperparameters to be $x_{\max} = 10$, and $\alpha = 0.75$ in this work. We solve this optimization problem via standard gradient descent, arriving at word representations $\mathbf{U}$ and tensor representations $\mathbf{Q}$.

3 Geometry of Phrasal Verbs

Representation Interpretation Suppose that we have a phrase (h, p_i, c) where h, p_i and c are the head word, the preposition $i (i \leq K)$ and the complement respectively. The inner product of the word vectors of h, p_i and c reflects how frequently h and c co-occur in the context of p. It also reflects how cohesive the triple is.

Recall that there is an extra $(K + 1)-$th slice that describes the word co-occurrences outside the preposition window, which considers cases such as the verb phrase (v, c) where v and c are the verb and its complement without a preposition in their shared context. Now consider a phrasal verb *sparked off* and a verb phrase with head *prompted*. For any complement word c that fits these two phrases–the phrasal verb having h as its head verb and p_i as its preposition, and the other, the verb phrase with v as its head–we can expect that

$$\langle \mathbf{u}_h, \mathbf{q}_i, \mathbf{w}_c \rangle \approx \langle \mathbf{u}_v, \mathbf{q}_{K+1}, \mathbf{w}_c \rangle.$$

In other words $\mathbf{u}_h \odot \mathbf{q}_i \approx \mathbf{u}_v \odot \mathbf{q}_{K+1}$, where $\mathbf{a} \odot \mathbf{b}$ denotes the pointwise multiplication (Hadamard product) of vectors $\mathbf{a}$ and $\mathbf{b}$. This suggests that: (1) The vector $\mathbf{q}_{K+1}$ is a constant vector for all (v, c) pairs, and that (2) we could paraphrase the verb phrase (h, p_i) by finding a verb v such that $\mathbf{u}_v \odot \mathbf{q}_{K+1}$ is closest to $\mathbf{u}_h \odot \mathbf{q}_i$.

$$\text{paraphrase} = \arg\min_v \|\mathbf{u}_v \odot \mathbf{q}_{K+1} - \mathbf{u}_h \odot \mathbf{q}_i\|. \quad (3)$$

This shows that well-trained embeddings are able to capture the relation between phrasal verbs and their equivalent single verb forms.

In Table 2, we list paraphrases of some verb phrases, which are generated from the weighted tensor decomposition. As can be seen, the tensor embedding gives reasonable paraphrasing, which

Phrase	replied to	blocked off	put in	pray for	dreamed of	sparked off
Paraphrase	answered	intercepted	place	hope	wanted	prompted
Phrase	stuck with	derived from	switched over	asked for	passed down	blend in
Paraphrase	stalled	generated	transferred	requested	delivered	mix

Table 2: Paraphrasing of prepositional phrases.

validates that the trained embedding is interpretable in terms of lexical semantics.

In the next two sections, we evaluate the proposed tensor-based preposition embeddings in the context of two important NLP downstream tasks: preposition selection and preposition attachment disambiguation. In this work, we use the English WikiCorpus (around 9 GB) as the training corpus for different sets of embeddings. We train tensor embeddings with both Orth-ALS and weighted decomposition. The implementation of Orth-ALS is built upon the SPLATT toolkit (Smith and Karypis, 2016). We perform orthogonalization in the first 5 iterations in Orth-ALS decomposition, and the training is completed when its performance stabilizes. As for the weighted decomposition, we train for 20 iterations, and its hyperparameters are set as $x_{\max} = 10$, and $\alpha = 0.75$.

We also include two baselines for comparison–word2vec's CBOW model and GloVe. We set 20 training iterations for both the models. The hyperparameters in word2vec are set as: window size=6, negative sampling=25 and downsampling=1e-4. The hyperparameters in GloVe are set as: window size=6, x_{max}=10, α=0.75 and minimum word count=5. We note that all the representations in this study–word2vec, GloVe and our tensor embedding–are of dimension 200.

4 Downstream Application: Preposition Selection

Grammatical error detection and correction constitute important tasks in NLP. Among grammatical errors, prepositional errors constitute about 13% of all errors, ranking second among the most common error types (Leacock et al., 2010). This is due to the fact that prepositions are highly polysemous and have idiosyncratic usage. Selecting a preposition depends on how well we can capture the interaction between a preposition and its context. Hence we choose this task to evaluate how well the lexical interactions are captured by different methods.

Task. Given a sentence in English containing a preposition, we either replace the preposition with the correct one or retain it. For example, in the sentence "It can save the effort to carrying a lot of cards," "to" should be corrected as "of." Formally, there is a closed set of preposition candidates $P = \{p_1, \ldots, p_m\}$. A preposition p is used in a sentence s consisting of words $s = \{\ldots, w_{-2}, w_{-1}, p, w_1, w_2, \ldots\}$. If used incorrectly, we need to replace p by another preposition $\hat{p} \in P$ based on the context.

FCE	# of sent	27119
	# of prep	60279
	Error ratio	4.8
CoNLL	# of sent	1375
	# of prep	3241
	Error ratio	4.7
SE	# of sent	5917
	# of prep	15814
	Error ratio	38.2

Table 3: Dataset statistics.

Dataset. For training, we use the data from the Cambridge First Certificate in English (FCE) exam, just as used by the state-of-the-art on preposition error correction (Prokofyev et al., 2014). As for test data, we use two the CoNLL-2013 and the Stack Exchange (SE) datasets. The CoNLL dataset on preposition error correction was published by the CoNLL 2013 shared task (Ng et al., 2014), collected from 50 essays written by 25 non-native English learners at a university. The SE dataset consists of texts generated by non-native speakers on the Stack Exchange website. Detailed statistics are shown in Table 3. We focus on the most frequent 49 prepositions listed in Appendix A.

Evaluation metric. Three metrics–precision, recall and F1 score–are used to evaluate the preposition selection performance.

Our algorithm. We first preprocess the dataset by removing articles, determiners and pronouns, and take a context window of 3. We divide the task into two steps: error detection and error cor-

Dataset	Method	Precision	Recall	F1 score
CoNLL	State-of-the-art	0.2592	0.3611	0.3017
	Word2vec	0.1558	0.1579	0.1569
	GloVe	0.1538	0.1578	0.1558
	Our method (ALS)	0.3355	0.3355	0.3355
	Our method (WD)	0.3590	0.3684	**0.3636**
SE	State-of-the-art	0.2704	0.2961	0.2824
	Word2vec	0.2450	0.2585	0.2516
	GloVe	0.2454	0.2589	0.2520
	Our method (ALS)	0.2958	0.3146	**0.3049**
	Our method (WD)	0.2899	0.3055	0.2975

Table 4: Performance on preposition selection.

rection. Firstly, we decide whether a preposition is used correctly in the context. If not, we suggest another preposition as replacement in the second step. The detection step uses only three features: the cosine similarity between the the current preposition embedding and the average context embedding, the rank of the preposition in terms of this cosine similarity, and the probability that this preposition is not changed in the training corpus. We build a decision tree classifier with these three features and find that we can identify errors with 98% F1 score in the CoNLL dataset and 96% in the SE dataset.

For the error correction part, we only focus on the errors detected in the first stage. Suppose that the original preposition is q, and the candidate preposition is p with the embedding v_p. The word vectors in the left context window are averaged as the left context embedding v_ℓ, and the right vectors are averaged to give the right context embedding v_r. We have the following features:

1. Embedding features: $\mathbf{v}_\ell, \mathbf{v}_p$ and $\mathbf{v}_r$;
2. Pair similarity between the preposition and the context: maximum of the similarity of the preposition between the left and the right context, i.e., pair sim $= \max\left(\frac{\mathbf{v}_\ell^T \mathbf{v}_p}{\|\mathbf{v}_\ell\|_2 \cdot \|\mathbf{v}_p\|_2}, \frac{\mathbf{v}_r^T \mathbf{v}_p}{\|\mathbf{v}_r\|_2 \cdot \|\mathbf{v}_p\|_2}\right)$;
3. Triple similarity $= \frac{\langle \mathbf{v}_\ell, \mathbf{v}_p, \mathbf{v}_r \rangle}{\|\mathbf{v}_\ell\|_3 \cdot \|\mathbf{v}_p\|_3 \cdot \|\mathbf{v}_r\|_3}$;
4. Confusion probability: the probability that q is replaced by p in the training data.

A two-layer feed-forward neural network (FNN) with hidden layer sizes of 500 and 10 is trained with these features to score prepositions in each sentence. The preposition with the highest score is the suggested edit.

Baseline. The state-of-the-art on preposition selection uses n-gram statistics from a large corpus (Prokofyev et al., 2014). Features such as point-wise mutual information (PMI) and part-of-speech tags are fed into a supervised scoring system. Given a sentence with a preposition to either replace or retain, the preposition with the highest score is chosen.

The performance of the baseline is affected by both the system architecture and the features. To evaluate the benefits brought about by our tensor embedding-based features, we also consider other baselines which have the same two-step architecture whereas the features are generated from word2vec and GloVe embeddings. These baselines allow us to compare the representation power independent of the classifier.

Result. We compare our proposed embedding-based method against baselines mentioned in Table 4. We note that the proposed tensor embeddings achieve the best performance among all approaches. In particular, the tensor with weighted decomposition has the highest F1 score on the CoNLL dataset–a 6% improvement over the state-of-the-art. However, the tensor with ALS decomposition performs the best on the SE dataset, achieving a 2% improvement over the state-of-the art. We also note that with the same architecture, tensor embeddings perform much better than word2vec and GloVe embeddings on both the datasets. This validates the representation power of tensor embeddings of prepositions.

To get a deeper insight into the importance of the features in the preposition selection task, we also performed an ablation analysis of the tensor

Removed feature		Left context embedding	Prep embedding	Right context embedding	Pair similarity	Triple similarity	Confusion score
CoNLL	Precision	0.1558	0.2662	0.3117	0.3247	0.3247	0.3506
	Recall	0.1579	0.2697	0.3158	0.3289	0.3289	0.3553
	F1 score	0.1569	0.2680	0.3137	0.3268	0.3268	0.3529
SE	Precision	0.2587	0.2796	0.2649	0.2658	0.2647	0.1993
	Recall	0.2743	0.2964	0.2801	0.2818	0.2807	0.2114
	F1 score	0.2663	0.2877	0.2726	0.2735	0.2725	0.2052

Table 5: Ablation analysis in preposition selection.

method with weighted decomposition as shown in Table 5. We find that the left context is the most important feature in for the CoNLL dataset, whereas the confusion score is the most important for the SE dataset. Pair similarity and triple similarity are less important when compared with the other features. This is because the neural network was able to learn the lexical similarity from the embedding features, thus reducing the importance of the similarity features.

Discussion. Now we analyze different cases where our approach selects the wrong preposition. (1) *Limited context window*. We focus on the local context within a preposition's window. In some cases, we find that head words might be out of the context window. An instance of this is found in the sentence "prevent more of this kind of tragedy *to* happening" *to* should be corrected as *from*. Given the context window of 3, we cannot get the lexical clues provided by *prevent*, which leads to the selection error. (2) *Preposition selection requires more context*. Even when the context window contains all the words on which the preposition depends, it still may not be sufficient to select the right one. For example, in the sentence "it is controlled by some men *in* a bad purpose" where our approach replaces the preposition *in* with the preposition *on* given the high frequency of the phrase "on purpose". The correct preposition should be *for* based on the whole sentence.

5 Downstream Application: Prepositional Attachment

In this section, we discuss the task of prepositional phrase (PP) attachment disambiguation, a well-studied, but hard task in syntactic parsing. The PP attachment disambiguation inherently requires an accurate description of the interactions among the head, the preposition and the complement, which becomes an ideal task to evaluate our tensor-based embeddings.

Task. The English dataset used in this work is collected from a linguistic treebank by (Belinkov et al., 2015). It provides $35,359$ training and $1,951$ test instances. Each instance consists of several head candidates, a preposition and a complement word. The task is to pick the head to which the preposition attaches. In the example "he saw an elephant with long tusks", the words "saw" and "elephant" are the candidate head words.

Our algorithm. Let $\mathbf{v}_h, \mathbf{v}_p$ and $\mathbf{v}_c$ be embeddings for the head candidate h, preposition p and child c respectively. We then use the following features:

1. Embedding feature: candidate head, preposition and complement embedding;
2. Triple similarity: $\frac{\langle \mathbf{v}_h, \mathbf{v}_p, \mathbf{v}_c \rangle}{\|\mathbf{v}_h\|_3 \cdot \|\mathbf{v}_p\|_3 \cdot \|\mathbf{v}_c\|_3}$;
3. Head-preposition similarity: $\frac{\mathbf{v}_h^T \mathbf{v}_p}{\|\mathbf{v}_h\|_2 \cdot \|\mathbf{v}_p\|_2}$;
4. Head-child similarity: $\frac{\mathbf{v}_h^T \mathbf{v}_c}{\|\mathbf{v}_h\|_2 \cdot \|\mathbf{v}_c\|_2}$;
5. Part-of-speech (pos) tag of candidates and next words;
6. Distance between h and p.

We use a basic neural network, a two-layer feed-forward network (FNN) with hidden-layers of size 1000 and 20, to take the input features and predict the probability that a candidate is the head. The candidate with the highest likelihood is chosen as the head.

Baselines. For comparison, we include the following state-of-the-art approaches in preposition attachment disambiguation. The linguistic resources they used to enrich their features are listed in Table 6.

Classifier	HPCD (enriching)	LRFR	OntoLSTM	FNN	FNN	FNN	FNN
Embedding method	GloVe	Word2vec	Glove-extended	Word2vec	GloVe	Our method (ALS)	Our method (WD)
Resources	POS tag, WordNet, VerbNet	POS tag, WordNet, VerbNet	POS tag, WordNet	POS tag	POS tag	POS tag	POS tag
Accuracy	0.887	**0.903**	0.897	0.866	0.858	0.883	0.892

Table 6: Accuracy in prepositional attachment disambiguation.

(1) Head-Prep-Child-Dist (HPCD) Model (Belinkov et al., 2015): this compositional neural network is used to train task-specific representations of prepositions.
(2) Low-Rank Feature Representation (LRFR) (Yu et al., 2016): this method incorporates word parts, contexts and labels into a tensor, and uses decomposed vectors as features for disambiguation.
(3) Ontology LSTM (OntoLSTM) (Dasigi et al., 2017): the vectors are initialized with GloVe, extended by AutoExtend (Rothe and Schütze, 2015), and trained via LSTMs for head selection.

Similar to the experiments in the preposition selection task (see Section 4), we also include baselines which have the same feed-forward network architecture but generate features with vectors trained by word2vec and GloVe. They are denoted as FNN with different initializations in Table 6. Since the attachment disambiguation is a selection task, accuracy is a natural evaluation metric.

Result. We compare the results of the different approaches and the linguistic resources used in Table 6, where we see that our simple classifier built on the tensor representation is comparable in performance to the state-of-the-art (within 1% of the result). This result is notable considering that prior competitive approaches have used significant linguistic resources such as VerbNet and WordNet, whereas we use none. With the same feed-forward neural network as the classifier, our tensor-based approaches (both ALS and WD) achieve better performance than word2vec and GloVe.

An ablation analysis that is provided in Table 7 shows that the head vector feature affects the performance the most (indicating that heads interact more closely with prepositions), and the POS tag feature comes second. The similarity features appear less important since the classifier has access to the lexical relatedness via the embedding features. Prior works have reported the importance of the distance feature since 81.7% sentences take the word closest to the preposition as the head. In our experiments, the distance feature was found to be less important compared to the embedding features.

Discussion. We found that one source of attachment disambiguation error is the lack of a broader context in our features. A broader context is critical in examples such as "worked" and "system," which are head candidates of "for trades" in the sentence "worked on a system for trading". They are reasonable heads in the expressions "worked for trades" and "system for trades" and further disambiguation requires a context larger than what we considered.

6 Related Work

Word representation. Word embeddings have been successfully used in many NLP applications. Word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014) show that embeddings can capture lexical semantics very well. Zhang et al. (2014) studied embeddings which can generalize different similarity perspectives when combined with corresponding linear transformations. Unlike other words, the crucial syntactic roles of prepositions in addition to their rich semantic meanings have been highlighted in prior works (Hovy et al., 2010; Schneider et al., 2015). Nevertheless, word representations specifically focused on prepositions are not available and to the best of our knowledge, ours is the first work exploring this intriguing direction.

Tensor Decomposition. Tensors embed higher order interaction among different modes, and the tensor decomposition captures this interaction via lower dimensional representations. There are several decomposition methods such as Alternating Least Square (ALS) (Kolda and Bader, 2009), Si-

Removed feature	Head vector	Prep vector	Child vector	Head-prep similarity	Head-child similarity	Triple similarity	POS	Distance
Accuracy	0.843	0.871	0.880	0.877	0.885	0.873	0.850	0.872

Table 7: Ablation analysis in preposition attachment disambiguation.

multaneous Diagonalization (SD) (Kuleshov et al., 2015) and optimization-based methods (Liu and Nocedal, 1989; Moré, 1978). Orthogonalized Alternating Least Square (Orth-ALS) adds the step of component orthogonalization to each update step in the ALS method (Sharan and Valiant, 2017). Owing to its theoretical guarantees and, more relevantly due to its good empirical performance, Orth-ALS is the algorithm of choice in this paper.

Preposition Selection. Preposition selection, an important area of study in computational linguistics, is also a very practical topic in the context of grammar correction and second language learning. Prior works have used hand-crafted heuristic rules (Xiang et al., 2013), n-gram features (Prokofyev et al., 2014; Rozovskaya et al., 2013), and by the use of POS tags and dependency relations to enrich other features (Kao et al., 2013)–all toward addressing preposition error correction.

Prepositional Attachment Disambiguation. There is a storied literature on prepositional attachment disambiguation, long recognized as an important part of syntactic parsing (Kiperwasser and Goldberg, 2016). Recent works, based on word embeddings have pushed the boundary of state of the art empirical results. A seminal work in this direction is the Head-Prep-Child-Dist Model, which trained embeddings in a compositional network to maximize the accuracy of head prediction (Belinkov et al., 2015). The performance has been further improved in conjunction with semantic and syntactic features. A recent work has proposed an initialization with semantics-enriched GloVe embeddings, and retrained representations with LSTM-RNNs (Dasigi et al., 2017). Another recent work has used tensor decompositions to capture the relation between word representations and their labels (Yu et al., 2016).

7 Conclusion

Co-occurrence counts of word pairs in sentences and the resulting word vector representations (embeddings) have revolutionalized NLP research. A natural generalization is to consider co-occurrence counts of word triples, resulting in a third order tensor. Partly due to the size of the tensor (a vocabulary of 1M, leads to a tensor with 10^{18} entries!) and partly due to the extreme dynamic range of entries (including sparsity), word vector representations via tensor decompositions have largely been inferior to their lower order cousins (i.e., regular word embeddings).

In this work, we trek this well-trodden but arduous terrain by restricting word triples to the scenario when one of the words is a preposition. This is linguistically justified, since prepositions are understood to model interactions between pairs of words. Numerically, this is also very well justified since the sparsity and dynamic range of the resulting tensor is no worse than the original matrix of pairwise co-occurrence counts; this is because prepositions are very frequent and co-occur with essentially every word in the vocabulary.

Our intrinsic evaluations and new state-of-the-art results in downstream evaluations lend strong credence to the tensor-based approach to prepositional representation. We expect our vector representations of prepositions to be widely used in more complicated downstream NLP tasks where prepositional role is crucial, including "text to programs" (Guu et al., 2017).

Acknowledgements

This work is supported by IBM-ILLINOIS Center for Cognitive Computing Systems Research (C3SR) - a research collaboration as part of the IBM AI Horizons Network.

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Sofia, Bulgaria, pages 183–192. http://www.aclweb.org/anthology/W13-3520.

Animashree Anandkumar, Rong Ge, and Majid Janzamin. 2014. Guaranteed non-orthogonal tensor decomposition via alternating rank-1 updates. *arXiv preprint arXiv:1402.5180* .

Yonatan Belinkov, Tao Lei, Regina Barzilay, and Amir Globerson. 2015. Erratum: "exploring compositional architectures and word vector representations for prepositional phrase attachment". *TACL* 3:101. https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/593.

Eric Brill and Philip Resnik. 1994. A rule-based approach to prepositional phrase attachment disambiguation. In *Proceedings of the 15th conference on Computational linguistics-Volume 2*. Association for Computational Linguistics, pages 1198–1204.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. pages 1870–1879. https://doi.org/10.18653/v1/P17-1171.

Pierre Comon, Xavier Luciani, and André LF De Almeida. 2009. Tensor decompositions, alternating least squares and other tales. *Journal of chemometrics* 23(7-8):393–405.

Pradeep Dasigi, Waleed Ammar, Chris Dyer, and Eduard Hovy. 2017. Ontology-aware token embeddings for prepositional phrase attachment. *arXiv preprint arXiv:1705.02925* .

Daniël de Kok and Erhard Hinrichs. 2016. Transition-based dependency parsing with topological fields. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. volume 2, pages 1–7.

Jeanette S DeCarrico. 2000. *The structure of English: Studies in form and function for language teaching*, volume 1. University of Michigan Press/ESL.

Kelvin Guu, Panupong Pasupat, Evan Zheran Liu, and Percy Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. *arXiv preprint arXiv:1704.07926* .

Kaveh Hassani and Won-Sook Lee. 2017. Disambiguating spatial prepositions using deep convolutional networks. In *AAAI*. pages 3209–3215.

Dirk Hovy, Stephen Tratz, and Eduard Hovy. 2010. What's in a preposition?: dimensions of sense disambiguation for an interesting word class. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. Association for Computational Linguistics, pages 454–462.

Rodney Huddleston. 1984. *Introduction to the Grammar of English*. Cambridge University Press.

Ting-Hui Kao, Yu-Wei Chang, Hsun-Wen Chiu, Tzu-Hsi Yen, Joanne Boisson, Jian-Cheng Wu, and Jason S Chang. 2013. Conll-2013 shared task: Grammatical error correction nthu system description. In *CoNLL Shared Task*. pages 20–25.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *arXiv preprint arXiv:1603.04351* .

Tamara G Kolda and Brett W Bader. 2009. Tensor decompositions and applications. *SIAM review* 51(3):455–500.

Volodymyr Kuleshov, Arun Chaganty, and Percy Liang. 2015. Tensor factorization via matrix factorization. In *Artificial Intelligence and Statistics*. pages 507–516.

Jonathan K Kummerfeld, David Hall, James R Curran, and Dan Klein. 2012. Parser showdown at the wall street corral: An empirical investigation of error types in parser output. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, pages 1048–1059.

Claudia Leacock, Martin Chodorow, Michael Gamon, and Joel Tetreault. 2010. Automated grammatical error detection for language learners. *Synthesis lectures on human language technologies* 3(1):1–134.

Dong C Liu and Jorge Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical programming* 45(1):503–528.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Jorge J Moré. 1978. The levenberg-marquardt algorithm: implementation and theory. In *Numerical analysis*, Springer, pages 105–116.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The conll-2014 shared task on grammatical error correction. In *CoNLL Shared Task*. pages 1–14.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pages 1532–1543.

Roman Prokofyev, Ruslan Mavlyutov, Martin Grund, Gianluca Demartini, and Philippe Cudré-Mauroux. 2014. Correct me if i'm wrong: Fixing grammatical errors by preposition ranking. In *Proceedings of the*

23rd ACM International Conference on Conference on Information and Knowledge Management. ACM, pages 331–340.

Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. *arXiv preprint arXiv:1507.01127* .

Alla Rozovskaya, Kai-Wei Chang, Mark Sammons, and Dan Roth. 2013. The university of illinois system in the conll-2013 shared task. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*. pages 13–19.

Nathan Schneider, Vivek Srikumar, Jena D Hwang, and Martha Palmer. 2015. A hierarchy with, of, and for preposition supersenses. In *Proceedings of The 9th Linguistic Annotation Workshop*. pages 112–123.

Vatsal Sharan and Gregory Valiant. 2017. Orthogonalized als: A theoretically principled tensor decomposition algorithm for practical use. *arXiv preprint arXiv:1703.01804* .

Shaden Smith and George Karypis. 2016. SPLATT: The Surprisingly ParalleL spArse Tensor Toolkit. http://cs.umn.edu/~splatt/.

Yang Xiang, Bo Yuan, Yaoyun Zhang, Xiaolong Wang, Wen Zheng, and Chongqiang Wei. 2013. A hybrid model for grammatical error correction. In *CoNLL Shared Task*. pages 115–122.

Mo Yu, Mark Dredze, Raman Arora, and Matthew Gormley. 2016. Embedding lexical features via low-rank tensors. *arXiv preprint arXiv:1604.00461* .

Jingwei Zhang, Jeremy Salwen, Michael Glass, and Alfio Gliozzo. 2014. Word semantic representations using bayesian probabilistic tensor factorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 1522–1531.

A Roster of Prepositions

The list of most frequent 49 Prepositions in the task of preposition selection is shown below:

about, above, absent, across, after, against, along, alongside, amid, among, amongst, around, at, before, behind, below, beneath, beside, besides, between, beyond, but, by, despite, during, except, for, from, in, inside, into, of, off, on, onto, opposite, outside, over, since, than, through, to, toward, towards, under, underneath, until, upon, with.

From Phonology to Syntax: Unsupervised Linguistic Typology at Different Levels with Language Embeddings

Johannes Bjerva
Department of Computer Science
University of Copenhagen
Denmark
bjerva@di.ku.dk

Isabelle Augenstein
Department of Computer Science
University of Copenhagen
Denmark
augenstein@di.ku.dk

Abstract

A core part of linguistic typology is the classification of languages according to linguistic properties, such as those detailed in the World Atlas of Language Structure (WALS). Doing this manually is prohibitively time-consuming, which is in part evidenced by the fact that only 100 out of over 7,000 languages spoken in the world are fully covered in WALS.

We learn distributed language representations, which can be used to predict typological properties on a massively multilingual scale. Additionally, quantitative and qualitative analyses of these language embeddings can tell us how language similarities are encoded in NLP models for tasks at different typological levels. The representations are learned in an unsupervised manner alongside tasks at three typological levels: phonology (grapheme-to-phoneme prediction, and phoneme reconstruction), morphology (morphological inflection), and syntax (part-of-speech tagging).

We consider more than 800 languages and find significant differences in the language representations encoded, depending on the target task. For instance, although Norwegian Bokmål and Danish are typologically close to one another, they are phonologically distant, which is reflected in their language embeddings growing relatively distant in a phonological task. We are also able to predict typological features in WALS with high accuracies, even for unseen language families.

1 Introduction

For more than two and a half centuries, linguistic typologists have studied languages with respect to their structural and functional properties (Haspelmath, 2001; Velupillai, 2012). Although typology has a long history (Herder, 1772; Gabelentz, 1891; Greenberg, 1960, 1974; Dahl, 1985; Comrie, 1989; Haspelmath, 2001; Croft, 2002), computational approaches have only recently gained popularity (Dunn et al., 2011; Wälchli, 2014; Östling, 2015; Cotterell and Eisner, 2017; Asgari and Schütze, 2017; Malaviya et al., 2017; Bjerva and Augenstein, 2018). One part of traditional typological research can be seen as assigning sparse explicit feature vectors to languages, for instance manually encoded in databases such as the World Atlas of Language Structures (WALS, Dryer and Haspelmath, 2013). A recent development which can be seen as analogous to this is the process of learning distributed language representations in the form of dense real-valued vectors, often referred to as *language embeddings* (Tsvetkov et al., 2016; Östling and Tiedemann, 2017; Malaviya et al., 2017). We hypothesise that these language embeddings encode typological properties of language, reminiscent of the features in WALS, or even of parameters in Chomsky's Principles and Parameters framework (Chomsky, 1993).

In this paper, we model languages in deep neural networks using language embeddings, considering three typological levels: phonology, morphology and syntax. We consider four NLP tasks to be representative of these levels: grapheme-to-phoneme prediction and phoneme reconstruction, morphological inflection, and part-of-speech tagging. We pose three research questions (**RQ**s):

RQ 1 Which typological properties are encoded in task-specific distributed language representations, and can we predict phonological, morphological and syntactic properties of languages using such representations?

RQ 2 To what extent do the encoded properties change as the representations are fine tuned for tasks at different linguistic levels?

RQ 3 How are language similarities encoded in fine-tuned language embeddings?

Proceedings of NAACL-HLT 2018, pages 907–916
New Orleans, Louisiana, June 1 - 6, 2018.

One of our key findings is that language representations differ considerably depending on the target task. For instance, for grapheme-to-phoneme mapping, the differences between the representations for Norwegian Bokmål and Danish increase rapidly during training. This is due to the fact that, although the languages are typologically close to one another, they are phonologically distant.

2 Related work

Computational linguistics approaches to typology are now possible on a larger scale than ever before due to advances in neural computational models. Even so, recent work only deals with fragments of typology compared to what we consider here.

Computational typology has to a large extent focused on exploiting word or morpheme alignments on the massively parallel New Testament, in approximately 1,000 languages, in order to infer word order (Östling, 2015) or assign linguistic categories (Asgari and Schütze, 2017). Wälchli (2014) similarly extracts lexical and grammatical markers using New Testament data. Other work has taken a computational perspective on language evolution (Dunn et al., 2011), and phonology (Cotterell and Eisner, 2017; Alishahi et al., 2017).

Language embeddings In this paper, we follow an approach which has seen some attention in the past year, namely the use of distributed language representations, or *language embeddings*. Some typological experiments are carried out by Östling and Tiedemann (2017), who learn language embeddings via multilingual language modelling and show that they can be used to reconstruct genealogical trees. Malaviya et al. (2017) learn language embeddings via neural machine translation, and predict syntactic, morphological, and phonetic features.

Contributions Our work bears the most resemblance to Bjerva and Augenstein (2018), who fine-tune language embeddings on the task of PoS tagging, and investigate how a handful of typological properties are coded in these for four Uralic languages. We expand on this and thus contribute to previous work by: (i) introducing novel qualitative investigations of language embeddings, in addition to thorough quantitative evaluations; (ii) considering four tasks at three different typological levels; (iii) considering a far larger sample of several hundred languages; and (iv) grounding the language representations in linguistic theory.

3 Background

3.1 Distributed language representations

There are several methods for obtaining distributed language representations by training a recurrent neural language model (Mikolov et al., 2010) simultaneously for different languages (Tsvetkov et al., 2016; Östling and Tiedemann, 2017). In these recurrent multilingual language models with long short-term memory cells (LSTM, Hochreiter and Schmidhuber, 1997), languages are embedded into an n-dimensional space. In order for multilingual parameter sharing to be successful in this setting, the neural network is encouraged to use the language embeddings to encode features of language. In this paper, we explore the embeddings trained by Östling and Tiedemann (2017), both in their original state, and by further tuning them for our four tasks. These are trained by training a multilingual language model with language representations on a collection of texts from the New Testament, covering 975 languages. While other work has looked at the types of representations encoded in different layers of deep neural models (Kádár et al., 2017), we choose to look at the representations only in the bottom-most embedding layer. This is motivated by the fact that we look at several tasks using different neural architectures, and want to ensure comparability between these.

3.1.1 Language embeddings as continuous Chomskyan parameter vectors

We now turn to the theoretical motivation of the language representations. The field of NLP is littered with distributional word representations, which are theoretically justified by distributional semantics (Harris, 1954; Firth, 1957), summarised in the catchy phrase *You shall know a word by the company it keeps* (Firth, 1957). We argue that language embeddings, or distributed representations of language, can also be theoretically motivated by Chomsky's Principles and Parameters framework (Chomsky and Lasnik, 1993; Chomsky, 1993, 2014). Language embeddings encode languages as dense real-valued vectors, in which the dimensions are reminiscent of the parameters found in this framework. Briefly put, Chomsky argues that languages can be described in terms of principles

(abstract rules) and parameters (switches) which can be turned either on or off for a given language (Chomsky and Lasnik, 1993). An example of such a switch might represent the positioning of the head of a clause (i.e. either head-initial or head-final). For English, this switch would be set to the 'initial' state, whereas for Japanese it would be set to the 'final' state. Each dimension in an n-dimensional language embedding might also describe such a switch, albeit in a more continuous fashion. The number of dimensions used in our language representations, 64, is a plausible number of parameter vector dimensions (Dunn et al., 2011). If we were able to predict typological features using such representations, this lends support to the argument that languages, at the very least, *can* be represented by theoretically motivated parameter vectors, with the given dimensionality.

3.2 Typological features in the World Atlas of Language Structure

In the experiments for **RQ1** and **RQ2** we predict typological features extracted from WALS (Dryer and Haspelmath, 2013). We choose to investigate three linguistic levels of language: phonology, morphology, and syntax. This is motivated by three factors: (i) these features are related to NLP tasks for which data is available for a large language sample; (ii) the levels cover a range from basic phonological and morphological structure, to syntactic structure, allowing us to approach our research question from several angles; and (iii) the features in these categories are coded in WALS for a relatively large selection of languages. We extract the three feature sets which represent these typological levels of language from WALS.[1]

Phonological features cover 20 features ranging from descriptions of the consonant and vowel inventories of a particular language to presence of tone and stress markers. As an example, consider WALS feature 13A (Tone).[2] This feature takes three feature values: (i) *no tones*, (ii) *simple tone system*, and (iii) *complex tone system*. Most Indo-European languages, such as English, Spanish, and Russian, do not have any tones (i). Norwegian and Swedish are exceptions to this, as they both have simple tone systems (ii) similar to that in Japanese. Finally, complex tone systems (iii) are typically found in several African languages as well as languages in South-East Asia.

Morphological features cover a total of 41 features. We consider the features included in the Morphology chapter as well as those included in the Nominal Categories chapter to be morphological in nature.[3] This includes features such as the number of genders, usage of definite and indefinite articles and reduplication. As an example, consider WALS feature 37A (Definite Articles).[4] This feature takes five values: (i) *Definite word distinct from demonstrative*, (ii) *Demonstrative word used as definite article*, (iii) *Definite affix*, (iv) *No definite, but indefinite article*, and (v) *No definite or indefinite article*. Again, most Indo-European languages fall into category (i), with Norwegian, Swedish, and Danish as relative outliers in category (iii).

Word-order features cover 56 features, encoding properties such as the ordering of subjects, objects and verbs. As an example, consider WALS feature 81A (Order of Subject, Object and Verb).[5] This feature takes all possible combinations of the three word classes as its feature values, with the addition of a special class for *No dominant order*. Most languages in WALS fall into the categories SOV (41.0%) and SVO (35.4%).

4 Method

The general set-up of the experiments in this paper is as follows. We aim at answering our three research questions dealing with typological properties and similarities as encoded in language embeddings. In order to do this, we attempt to predict typological features as they are encoded in WALS, using language embeddings which have been fine-tuned during training on tasks related to different typological properties. The main interest in this paper is therefore not on how well each model performs on a given NLP task, but rather on what the resulting language embeddings encode.

Concretely, we use language embeddings $\vec{l}_i$ from a given training iteration of a given task as input to a k-NN classifier, which outputs the typological class a language belongs to (as coded in WALS). We train separate classifiers for each

[1] These are defined in the chapter structure in WALS: http://wals.info/chapter

[2] http://wals.info/feature/13A

[3] This choice was made as, e.g., feature 37A (Definite Articles) includes as a feature value whether a definite affix is used.

[4] http://wals.info/feature/37A

[5] http://wals.info/feature/81A

typological property and each target task. When $i = 0$, this indicates the pre-trained language embeddings as obtained from Östling and Tiedemann (2017). Increasing i indicates the number of iterations over which the system at hand has been trained. In each experiment, for a given iteration i, we consider each $\vec{l_i} \in L$ where L is the intersection $L_{task} \cap L_{pre}$, where L_{task} is the set of languages for a given task, and L_{pre} is the set of languages for which we have pre-trained embeddings.

All results in the following sections are the mean of three-fold cross-validation, and the mean over the WALS features in each given category.[6] We run the experiments in a total of three settings: (i) evaluating on randomly selected language/feature pairs from a task-related feature set; (ii) evaluating on an *unseen language family from a task-related feature set*; (iii) evaluating on randomly selected language/feature pairs from all WALS feature sets. This allows us to establish how well we can predict task-related features given a random sample of languages (i), and a sample from which a whole language family has been omitted (ii). Finally, (iii) allows us to compare the task-specific feature encoding with a general one.

A baseline reference is also included, which is defined as the most frequently occurring typological trait within each category.[7] For instance, in the morphological experiments, we only consider the 41 WALS features associated with the categories of morphology and nominal categories. The overlap between languages for which we have data for morphological inflection and languages for which these WALS features are coded is relatively small (fewer than 20 languages per feature). This small dataset size is why we have opted for a non-parametric k-Nearest Neighbours classifier for the typological experiments. We use $k = 1$, as several of the features take a large number of class values, and might only have a single instance represented in the training set.

Table 1 shows the datasets we consider (detailed in later sections), the typological class they are related to, the size of the language sample in the task, and the size of the intersection $L_{task} \cap L_{pre}$. The number of pre-trained language embeddings, $|L_{pre}|$, is 975 in all cases. We focus the evaluation for each task-specific language embedding set on the typological property relevant to that dataset. In addition, we also evaluate on a set of all typological properties in WALS. Note that the evaluation on all properties is only comparable to the evaluation on each specific property, as the set of languages under consideration differs between tasks.

Dataset	Class	$\lvert L_{task}\rvert$	$\lvert L_{task} \cap L_{pre}\rvert$
G2P	Phonology	311	102
ASJP	Phonology	4,664	824
SIGMORPHON	Morphology	52	29
UD	Syntax	50	27

Table 1: Overview of tasks and datasets.

5 Phonology

5.1 Grapheme-to-phoneme

We use grapheme-to-phoneme (G2P) as a proxy of a phonological task (Deri and Knight, 2016; Peters et al., 2017). The dataset contains over 650,000 such training instances, for a total of 311 languages (Deri and Knight, 2016). The task is to produce a phonological form of a word, given its orthographic form and the language in which it is written. Crucially, this mapping is highly different depending on the language at hand. For instance, take the word written *variation*, which exists in both English and French:

```
(English, variation) -> ˌvɛəriˈeɪʃən
(French,  variation) -> ˌvaʁjaˈsjɔ̃
```

5.1.1 Experiments and Analysis

We train a sequence-to-sequence model with attention for the task of grapheme-to-phoneme mapping.[8] The model takes as input the characters of each source form together with the language embedding for the language at hand and outputs a predicted phonological form. Input and output alphabets are shared across all languages. The system is trained over 3,000 iterations.

Quantitative results Since we consider Grapheme-to-Phoneme as a phonological task, we focus the quantitative evaluation on phonological features from WALS. We run experiments using the language embeddings as features for a simple

[6] The mean accuracy score is a harsh metric, as some features are very difficult to predict due to them, e.g., being very language specific or taking a large number of different values.

[7] The languages represented in several of the tasks under consideration have a high Indo-European bias. Hence, several of the properties have a relatively skewed distribution, providing us with a strong baseline.

[8] The system is described in detail in Section 8.

k-NN classifier. The results in Table 2 indicate that G2P is a poor proxy for language phonology, however, as typological properties pertaining to phonology are not encoded. That is to say, the k-NN results do not outperform the baseline, and performance is on par even after fine tuning (no significant difference, $p > 0.05$). In the unseen setting, however, we find that pre-trained language embeddings are significantly better ($p < 0.05$) at predicting the phonological features than both fine-tuned ones and the baseline.

System / features	Random phon.	Unseen phon.	All feat.
Most Frequent Class	*75.46%	65.57%	79.90%
k-NN (pre-trained)	71.45%	*86.54%	80.39%
k-NN (fine-tuned)	71.66%	82.36%	79.17%

Table 2: Accuracies on prediction of WALS features with language embeddings fine-tuned on Grapheme-to-Phoneme mapping. Asterisks indicate results significantly higher than both other conditions ($p < 0.05$).

Qualitative results We now turn to why this task is not a good proxy of phonology. The task of grapheme-to-phoneme is more related to the processes in the diachronic development of the writing system of a language than it is to the actual genealogy or phonology of the language. This is evident when considering the Scandinavian languages Norwegian and Danish which are typologically closely related, and have almost exactly the same orthography. In spite of this fact, the phonology of the two languages differs drastically due to changes in Danish phonology, which impacts the mapping from graphemes to phonemes severely. Hence, the written forms of the two languages should be very similar, which makes the language embeddings based on language modelling highly similar to one another. However, when the embeddings are fine-tuned on a task taking orthography as well as phonology into account, this is no longer the case. Figure 1 shows that the language embeddings of Norwegian Bokmål and Danish diverge from each other, which is especially striking when comparing to the converging with the typologically much more distant languages Tagalog and Finnish. However, the absolute difference between Norwegian Bokmål and both Tagalog/Finnish is still greater than that of Norwegian Bokmål and Danish even after 3,000 iterations.

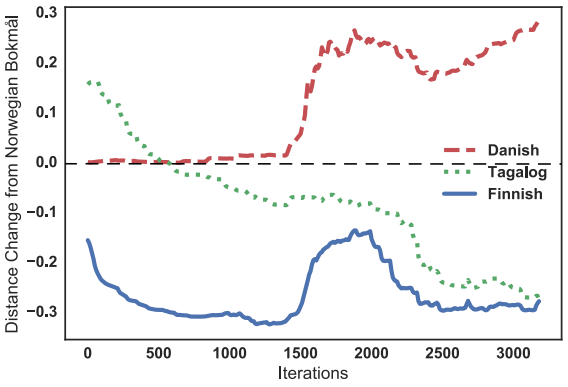


Figure 1: Language similarities between Norwegian, and Danish/Tagalog/Finnish, as G2P-based embeddings are fine-tuned.

5.2 Phonological reconstruction

As a second phonological task, we look at phonological reconstruction using word lists from the Automated Similarity Judgement Program (ASJP, Wichmann et al. (2016)). This resource contains word lists of at least 40 words per language for more than 4,500 languages. The task we consider is to reproduce a given source phonological form, also given the language, for instance:

```
(English, wat3r) -> wat3r
```

The intuition behind these experiments is that languages with similar phonetic inventories will be grouped together, as reflected in changes in the language embeddings.

5.2.1 Experiments and Analysis

We train a sequence-to-sequence model with attention, framed as an auto-encoding problem, using the same sequence-to-sequence architecture and setup as for the grapheme-to-phoneme task. The model takes as input the characters of each source form together with the language embedding for the language at hand and outputs the predicted target form which is identical to the source form.

Quantitative results Since we also consider phonological reconstruction to be a phonological task, we focus the quantitative evaluation on phonological features from WALS. As with the G2P experiments, Table 3 shows that the fine-tuned embeddings do not offer predictive power above the most frequent class baseline ($p > 0.05$). Observing the changes in the language embeddings reveals that the embeddings are updated to

a very small extent, indicating that these are not used by the model to a large extent. This can be explained by the fact that the task is highly similar for each language, and that the model largely only needs to learn to copy the input string.

We do, however find that evaluating on a set with an unseen language family does yield results significantly above baseline levels with the pre-trained embeddings ($p < 0.05$), which together with the G2P results indicate that the language modelling objective does encode features to some extent related to phonology.

System / features	Random phon.	Unseen phon.	All feat.
Most Frequent Class	***59.39%**	63.71%	***58.12%**
k-NN (pre-trained)	53.02%	***77.44%**	51.6%
k-NN (fine-tuned)	53.09%	***77.45%**	51.9%

Table 3: Accuracies on prediction of WALS features with language embeddings fine-tuned on Phonological Reconstruction. Asterisks indicate results significantly higher than non-bold conditions ($p < 0.05$).

6 Morphology

6.1 Morphological inflection

We use data from the Unimorph project, specifically the data released for the SIGMORPHON-2017 shared task (Cotterell et al., 2017).[9] This data covers 52 languages, thereby representing a relatively large typological variety. Whereas the shared task has two subtasks, namely inflection and paradigm cell filling, we only train our system using the inflection task. This was a choice of convenience, as we are not interested in solving the task of morphological paradigm cell filling, but rather observing the language embeddings as they are fine-tuned. Furthermore we focus on the high-resource setting in which we have access to 10,000 training examples per language. The inflection subtask is to generate a target inflected form given a lemma with its part-of-speech as in the following example:

```
(release, V;V.PTCP;PRS) -> releasing
```

6.1.1 Morphological experiments

We train a sequence-to-sequence model with attention over 600 iterations, using the same sequence-to-sequence architecture from the previous tasks.

[9] https://unimorph.github.io/

Quantitative results Since this is considered a morphological task, Table 4 contains results using the language embeddings to predict morphological properties. The fine-tuned language embeddings in this condition are able to predict morphological properties in WALS significantly above baseline levels and pre-trained embeddings ($p < 0.05$). We further also obtain significantly better results in the unseen setting ($p < 0.05$), in which the language family evaluated on is not used in training. This indicates that these properties are important to the model when learning the task at hand.

System / Features	Random morph.	Unseen morph.	All feat.
Most Frequent Class	77.98%	85.68%	84.12%
k-NN (pre-trained)	74.49%	88.83%	**84.97%**
k-NN (fine-tuned)	***82.91%**	***91.92%**	84.95%

Table 4: Accuracies on prediction of WALS features with language embeddings fine-tuned on morphological inflection. Asterisks indicate results significantly higher than both other conditions ($p < 0.05$).

Qualitative results The performance of the fine-tuned embeddings on prediction of morphological features is above baseline for most features. For 18 out of the 35 features under consideration both the baseline and k-NN performances are at 100% from the outset, so these are not considered here.[10] Figure 2 shows two of the remaining 17 features.[11] We can observe two main patterns: For some features such as 49A (Number of cases), fine-tuning on morphological inflection increases the degree to which the features are encoded in the language embeddings. This can be explained by the fact that the number of cases in a language is central to how morphological inflection is treated by the model. For instance, languages with the same number of cases might benefit from sharing certain parameters. On the other hand, the feature 38A (Indefinite Articles) mainly encodes whether the indefinite word is distinct or not from the word for *one*, and it is therefore not surprising that this is not learned in morphological inflection.

7 Word order

7.1 Part-of-speech tagging

We use PoS annotations from version 2 of the Universal Dependencies (Nivre et al., 2016). As

[10] This is partially explained by the fact that certain categories were completely uniform in the small sample as well as by the Indo-European bias in the sample.

[11] The full feature set is included in the Supplements.

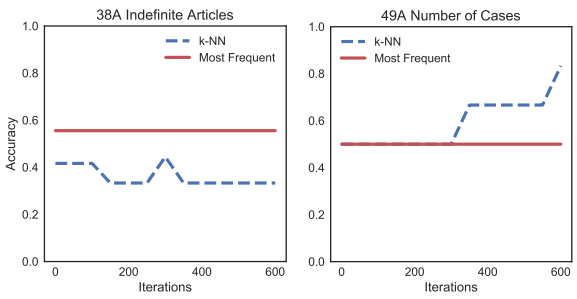


Figure 2: Prediction of two morphological features in WALS with morphological language embeddings, one data point per 50 iterations.

we are mainly interested in observing the language embeddings, we down-sample all training sets to 1,500 sentences (approximate number of sentences of the smallest data sets) so as to minimise any size-related effects.

7.1.1 Word-order experiments

We approach the task of PoS tagging using a fairly standard bi-directional LSTM architecture based on Plank et al. (2016), detailed in Section 8.

Quantitative results Table 5 contains results on WALS feature prediction using language embeddings fine-tuned on PoS tagging. We consider both the set of word order features, which are relevant for the dataset, and a set of all WALS features. Using the fine-tuned embeddings is significantly better than both the baseline and the pre-trained embeddings ($p < 0.05$), in both the random and the unseen conditions, indicating that the model learns something about word order typology. This can be expected, as word order features are highly relevant when assigning a PoS tag to a word.

System / features	Random W-Order	Unseen W-Order	All feat.
Most Frequent Class	76.81%	82.47%	82.93%
k-NN (pre-trained)	76.66%	92.76%	82.69%
k-NN (fine-tuned)	*80.81%	*94.48%	83.55%

Table 5: Accuracies on prediction of WALS features with language embeddings fine-tuned on PoS tagging. Asterisks indicate the result in bold significantly outperforming both other conditions ($p < 0.05$).

Qualitative results We now turn to the syntactic similarities between languages as encoded in the fine-tuned language embeddings. We consider a set of the North-Germanic languages Icelandic, Swedish, Norwegian Nynorsk, Danish, Norwegian Bokmål, the West-Germanic language English, and the Romance languages Spanish, French, and Italian. We apply hierarchical clustering using UPGMA (Michener and Sokal, 1957) to the pre-trained language embeddings of these languages.[12] Striking here is that English is grouped together with the Romance languages. This can be explained by the fact that English has a large amount of vocabulary stemming from Romance loan words, which under the task of language modelling results in a higher similarity with such languages. We then cluster the embeddings fine-tuned on PoS tagging in the same way. In this condition, English has joined the rest of the Germanic languages' cluster. This can be explained by the fact that, in terms of word ordering and morpho-syntax, English is more similar to these languages than it is to the Romance ones.

We can also observe that, whereas the orthographically highly similar Norwegian Bokmål and Danish form the first sub-cluster in the pre-trained condition, Norwegian Nynorsk replaces Danish in this pairing when fine-tuning on PoS tagging. This can be explained by the fact that morpho-syntactic similarities between the two written varieties of Norwegian are more similar to one another.

8 Implementation

8.1 Sequence-to-sequence modelling

The system architecture used in the sequence-to-sequence tasks, i.e., G2P, phonological reconstruction, and morphological inflection is depicted in Figure 3. The system is based on that developed by Östling and Bjerva (2017) and is implemented using Chainer (Tokui et al., 2015). We modify the architecture by concatenating a language embedding, $\vec{l}$, to the character embeddings before encoding. In the grapheme-to-phoneme and phonological reconstruction experiments, the one-hot feature mapping before decoding is irrelevant and therefore omitted. The rest of the hyperparameters are the same as in Östling and Bjerva (2017).

8.2 Sequence labelling

This system is based on Plank et al. (2016), and is implemented using DyNet (Neubig et al., 2017). We train using the Adam optimisation algorithm (Kingma and Ba, 2014) over a maximum of 10 epochs using early stopping. We make two modifications to the bi-LSTM architecture of Plank et al.

[12] Included in the Supplements due to space restrictions.

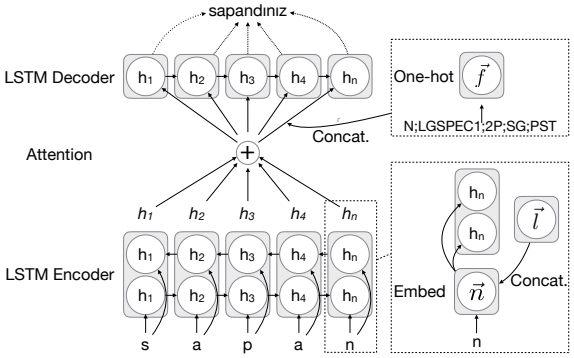


Figure 3: System architecture used in the seq-to-seq tasks (morphological inflection, G2P, and phonological reconstruction). Figure adapted with permission from Östling and Bjerva (2017).

(2016). First of all, we do not use any atomic embedded word representations, but rather use only character-based word representations. This choice was made so as to encourage the model not to rely on language-specific vocabulary. Additionally, we concatenate a pre-trained language embedding to each word representation. In our formulation, each word w is represented as $LSTM_c(w) + \vec{l}$, where $LSTM_c(w)$ is the final states of a character bi-LSTM running over the characters in a word and $\vec{l}$ is an embedded language representation. We use a two-layer deep bi-LSTM with 100 units in each layer, and 100-dimensional character embeddings. The rest of the hyper-parameters are the same as in Plank et al. (2016).[13]

9 Discussion and Conclusions

The language embeddings obtained by fine-tuning on linguistic tasks at various typological levels were found to include typological information somehow related to the task at hand. This lends some support to the theoretical foundations of such representations, in that it shows that it is possible to learn something akin to a vector of continuous Chomskyan parameters (Chomsky, 1993).

9.1 RQ1: Encoding of typological features in task-specific language embeddings

The features which are encoded depend to a large degree on the task at hand. The language embeddings resulting from the phonological tasks did not encode phonological properties in the sense of WALS features, whereas the pre-trained ones did.

[13]Both modified systems are included in the Supplements, and will be made publicly available.

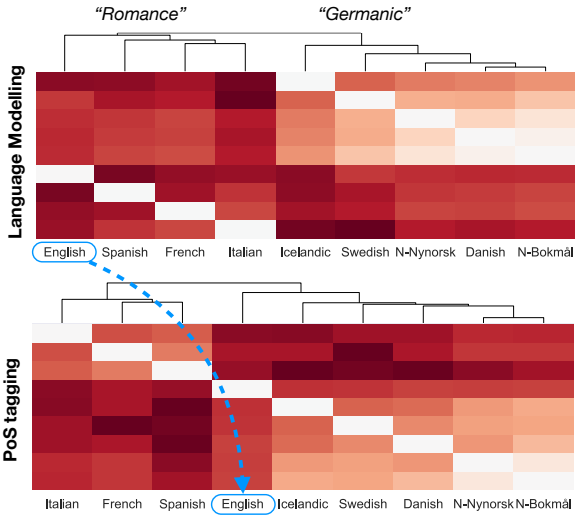


Figure 4: Language similarities changing during fine tuning.

The morphological language embeddings were found to encode morphological features, and the PoS language embeddings were similarly found to encode word order features.

A promising result is the fact that we were able to predict typological features for unseen language families. That is to say, without showing, e.g., a single Austronesian training instance to the k-NN classifier, typological features could still be predicted with high accuracies. This indicates that we can predict typological features with language embeddings, even for languages for which we have no prior typological knowledge.

Table 6 contains a comparison of the top five and bottom five feature prediction accuracies for the ASJP task.[14] In the case of the phonologically oriented ASJP task it is evident that the embeddings still encode something related to phonology, as four out of five best features are phonological.

9.2 RQ2: Change in encoding of typological features

The changes in the features encoded in language embeddings are relatively monotonic. Features are either learnt, forgotten, or remain static throughout training. This indicates that the language representations converge towards a single optimum.

9.3 RQ3: Language similarities

Training language embeddings in the task of multilingual language modelling has been found to reproduce trees which are relatively close matches

[14]See the Supplement for the remaining tasks.

to more traditional genealogical trees (Östling and Tiedemann, 2017). We show a similar analysis considering pre-trained and PoS fine-tuned embeddings, and it is noteworthy that fine-tuning on PoS tagging in this case yielded a tree more faithful to genealogical trees, such as those represented on glottolog.org. Figure 4 shows an example of this, in which a language modelling objective places English with Romance languages. This makes sense, as the English lexicon contains a large amount of Romance vocabulary. When fine-tuning on PoS tagging, however, English is placed among the Germanic languages, as it shares more syntactic similarities with these.

Another striking result in terms of language similarities in fine-tuned language embedding spaces was found in the G2P task. We here found that the phonological differences between some otherwise similar languages, such as Norwegian Bokmål and Danish, were accurately encoded.

Task	Feature	WALS Chapter	Accuracy
ASJP	6A	Phonology	89.45
	18A	Phonology	88.91
	20A	Morphology	82.74
	19A	Phonology	82.58
	7A	Phonology	80.97
	144A	Word Order	14.48
	144L	Word Order	10.07
	62A	Nominal Syntax	10.00
	81B	Word Order	9.52
	133A	Lexicon	8.18

Table 6: Top 5 and bottom 5 accuracies in feature prediction using phonological language embeddings.

Acknowledgements

We would also like to thank Robert Östling for giving us access to the pre-trained language embeddings. Isabelle Augenstein is supported by Eurostars grant Number E10138. We further gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

References

Afra Alishahi, Marie Barking, and Grzegorz Chrupała. 2017. Encoding of phonology in a recurrent neural model of grounded speech. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. Association for Computational Linguistics, pages 368–378.

Ehsaneddin Asgari and Hinrich Schütze. 2017. Past, present, future: A computational investigation of the typology of tense in 1000 languages. In *EMNLP*. Association for Computational Linguistics, pages 113–124.

Johannes Bjerva and Isabelle Augenstein. 2018. Tracking Typological Traits of Uralic Languages in Distributed Language Representations. In *Proceedings of the Fourth International Workshop on Computational Linguistics for Uralic Languages (IWCLUL)*.

Noam Chomsky. 1993. *Lectures on government and binding: The Pisa lectures*. 9. Walter de Gruyter.

Noam Chomsky. 2014. *The minimalist program*. MIT press.

Noam Chomsky and Howard Lasnik. 1993. The theory of principles and parameters.

Bernard Comrie. 1989. *Language universals and linguistic typology: Syntax and morphology*. University of Chicago press.

Ryan Cotterell and Jason Eisner. 2017. Probabilistic typology: Deep generative models of vowel inventories. In *ACL*. Association for Computational Linguistics, pages 1182–1192.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, et al. 2017. Conll-sigmorphon 2017 shared task: Universal morphological reinflection in 52 languages. *arXiv preprint arXiv:1706.09031* .

William Croft. 2002. *Typology and universals*. Cambridge University Press.

Östen Dahl. 1985. *Tense and Aspect Systems*. Basil Blackwell Ltd., NewYork.

Aliya Deri and Kevin Knight. 2016. Grapheme-to-phoneme models for (almost) any language. In *ACL*. Association for Computational Linguistics, pages 399–408.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig. http://wals.info/.

Michael Dunn, Simon J Greenhill, Stephen C Levinson, and Russell D Gray. 2011. Evolved structure of language shows lineage-specific trends in word-order universals. *Nature* 473(7345):79–82.

J. R. Firth. 1957. A synopsis of linguistic theory pages 1930–1955. 1952–1959:1–32.

Georg von der Gabelentz. 1891. *Die Sprachwissenschaft, ihre Aufgaben, Methoden und bisherigen Ergebnisse*. Leipzig.

Joseph Greenberg. 1974. *Language typology: A historical and analytic overview*, volume 184. Walter de Gruyter.

Joseph H Greenberg. 1960. A quantitative approach to the morphological typology of language. *International journal of American linguistics* 26(3):178–194.

Z. Harris. 1954. Distributional structure. *Word* 10:146–162.

Martin Haspelmath. 2001. *Language typology and language universals: An international handbook*, volume 20. Walter de Gruyter.

J. Herder. 1772. *Abhandlung über den Ursprung der Sprache*. Berlin: Christian Friedrich Voß.

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

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Ákos Kádár, Grzegorz Chrupała, and Afra Alishahi. 2017. Representation of linguistic form and function in recurrent neural network. *Computational Linguistics* 43:761–780.

Chaitanya Malaviya, Graham Neubig, and Patrick Littell. 2017. Learning language representations for typology prediction. In *EMNLP*. Association for Computational Linguistics, pages 2519–2525.

Charles D Michener and Robert R Sokal. 1957. A quantitative approach to a problem in classification. *Evolution* 11(2):130–162.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*. volume 2, page 3.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980* .

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*.

Robert Östling. 2015. Word order typology through multilingual word alignment. In *The 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. pages 205–211.

Robert Östling and Johannes Bjerva. 2017. Su-rug at the conll-sigmorphon 2017 shared task: Morphological inflection with attentional sequence-to-sequence models. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*. Association for Computational Linguistics, pages 110–113.

Robert Östling and Jörg Tiedemann. 2017. Continuous multilinguality with language vectors. In *EACL*. Association for Computational Linguistics, pages 644–649.

Ben Peters, Jon Dehdari, and Josef van Genabith. 2017. Massively multilingual neural grapheme-to-phoneme conversion. In *Proceedings of the First Workshop on Building Linguistically Generalizable NLP Systems*. pages 19–26.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *ACL*. Association for Computational Linguistics, pages 412–418.

Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. 2015. Chainer: a next-generation open source framework for deep learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)*.

Yulia Tsvetkov, Sunayana Sitaram, Manaal Faruqui, Guillaume Lample, Patrick Littell, David Mortensen, Alan W Black, Lori Levin, and Chris Dyer. 2016. Polyglot neural language models: A case study in cross-lingual phonetic representation learning. In *NAACL-HLT*. Association for Computational Linguistics, pages 1357–1366.

Viveka Velupillai. 2012. *An introduction to linguistic typology*. John Benjamins Publishing.

Bernhard Wälchli. 2014. Algorithmic typology and going from known to similar unknown categories within and across languages. *Aggregating Dialectology, Typology, and Register Analysis: Linguistic Variation in Text and Speech* 28:355.

Søren Wichmann, Eric W. Holman, and Cecil H. Brown (eds.). 2016. The ASJP Database (version 17).

Monte Carlo Syntax Marginals for Exploring and Using Dependency Parses

Katherine A. Keith, Su Lin Blodgett, and **Brendan O'Connor**
College of Information and Computer Sciences
University of Massachusetts Amherst
{kkeith,blodgett,brenocon}@cs.umass.edu

Abstract

Dependency parsing research, which has made significant gains in recent years, typically focuses on improving the accuracy of single-tree predictions. However, ambiguity is inherent to natural language syntax, and communicating such ambiguity is important for error analysis and better-informed downstream applications. In this work, we propose a *transition sampling* algorithm to sample from the full joint distribution of parse trees defined by a transition-based parsing model, and demonstrate the use of the samples in probabilistic dependency analysis. First, we define the new task of *dependency path prediction*, inferring syntactic substructures over part of a sentence, and provide the first analysis of performance on this task. Second, we demonstrate the usefulness of our *Monte Carlo syntax marginal* method for parser error analysis and calibration. Finally, we use this method to propagate parse uncertainty to two downstream information extraction applications: identifying persons killed by police and semantic role assignment.[1]

1 Introduction

Dependency parsers typically predict a single tree for a sentence to be used in downstream applications, and most work on dependency parsers seeks to improve accuracy of such single-tree predictions. Despite tremendous gains in the last few decades of parsing research, accuracy is far from perfect—but perfect accuracy may be impossible since syntax models by themselves do not incorporate the discourse, pragmatic, or world knowledge necessary to resolve many ambiguities.

In fact, although relatively unexamined, substantial ambiguity already exists within commonly used discriminative probabilistic parsing models,

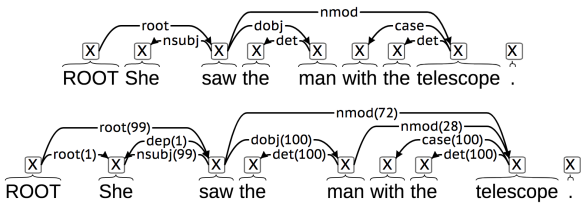


Figure 1: Example of a sentence with inherent ambiguity. **Top:** output from a greedy parser. **Bottom:** edge marginal probabilities from 100 samples in parentheses.

which define a parse forest—a probability distribution $p(y \mid x)$ over possible dependency trees $y \in \mathcal{Y}(x)$ for an input sentence x.

For example, the top of Figure 1 shows the predicted parse $y^{(greedy)}$ from such a parser (Chen and Manning, 2014), which resolves a prepositional (PP) attachment ambiguity in one manner; this prediction was selected by a standard greedy transition-based algorithm (§2.1). However, the bottom of Figure 1 shows *marginal* probabilities of individual (relation, governor, child) edges under this same model. These denote our estimated probabilities, across all possible parse structures, that a pair of words are connected with a particular relation (§2.4). For example, the two different PP attachment readings both exist within this parse forest with marginal probabilities

$$p(\ \text{nmod}(\text{saw}_2, \text{telescope}_7) \mid x) = 0.72 \quad (1)$$
$$p(\ \text{nmod}(\text{man}_4, \text{telescope}_7) \mid x) = 0.28, \quad (2)$$

where (1) implies she used a telescope to see the man, and (2) implies she saw a man who had a telescope.

These types of irreducible syntactic ambiguities exist and should be taken into consideration when analyzing syntactic information; for instance, one could transmit multiple samples (Finkel et al., 2006) or confidence scores (Bunescu, 2008) over

[1] Supporting code available at https://github.com/slanglab/transition_sampler

Proceedings of NAACL-HLT 2018, pages 917–928
New Orleans, Louisiana, June 1 - 6, 2018.

ambiguous readings to downstream analysis components.

In this work, we introduce a simple *transition sampling* algorithm for transition-based dependency parsing (§2.2), which, by yielding exact samples from the full joint distribution over trees, makes it possible to infer probabilities of long-distance or other arbitrary structures over the parse distribution (§2.4). We implement transition sampling—a very simple change to pre-existing parsing software—and use it to demonstrate several applications of probabilistic dependency analysis:

- Motivated by how dependency parses are typically used in feature-based machine learning, we introduce a new parsing-related task—*dependency path prediction*. This task involves inference over variable length *dependency paths*, syntactic substructures over only parts of a sentence.
- To accomplish this task, we define a *Monte Carlo syntax marginal* inference method which exploits information across samples of the entire parse forest. It achieves higher accuracy predictions than a traditional greedy parsing algorithm, and allows tradeoffs between precision and recall (§4).
- We provide a quantitative measure of the model's inherent uncertainty in the parse, *whole-tree entropy*, and show how it can be used for error analysis (§3).
- We demonstrate the method's (surprisingly) reasonable calibration (§5).
- Finally, we demonstrate the utility of our method to propagate uncertainty to downstream applications. Our method improves performance for giving probabilistic semantics to a rule-based event extractor to identify civilians killed by police (§6), as well as semantic role assignment (§7).

2 Monte Carlo dependency analysis

2.1 Overview of transition-based dependency parsing

We examine the *basic* form of the Universal Dependencies formalism (Nivre et al., 2016), where, for a sentence x of length N, a possible dependency parse y is a set of (relation, governorToken, childToken) edges, with a tree constraint that every token in the parse has exactly one governor—that is, for every token $w \in \{1..N\}$, there is exactly one triple $(r, g, w) \in y$ where it participates as a child. The governor is either one of the observed tokens, or a special ROOT vertex.

There exist a wide variety of approaches to machine learned, discriminative dependency parsing, which often define a probability distribution $p(y \mid x)$ over a domain of formally legal dependency parse trees $y \in \mathcal{Y}(x)$. We focus on *transition-based* dependency parsers (Nivre, 2003; Kübler et al., 2009), which (typically) use a stack-based automaton to process a sentence, incrementally building a set of edges. Transition-based parsers are very fast, have runtimes linear in sentence length, feature high performance (either state-of-the-art, or nearly so), and are easier to implement than other modeling paradigms (§2.5).

A probabilistic transition-based parser assumes the following stochastic process to generate a parse tree:

- Initialize state S_0
- For $n = 1, 2, \ldots$:
  - (A) $a_n \sim p(a_n \mid S_{n-1})$
  - (B) $S_n := \text{Update}(S_{n-1}, a_n)$
  - (C) Break if $\text{InEndState}(S_n)$

Most state transition systems (Bohnet et al., 2016) use shift and reduce actions to sweep through tokens from left to right, pushing and popping them from a stack to create the edges that populate a new parse tree y. The action decision probability, $p(a_{next} \mid S_{current})$, is a softmax distribution over possible next actions. It can be parameterized by any probabilistic model, such as log-linear features of the sentence and current state (Zhang and Nivre, 2011), multilayer perceptrons (Chen and Manning, 2014), or recurrent neural networks (Dyer et al., 2015; Kiperwasser and Goldberg, 2016).

To predict a single parse tree on new data, a common inference method is *greedy decoding*, which runs a close variant of the above transition model as a deterministic automaton, replacing stochastic step (A) with a best-action decision, $a_n := \arg\max_{a_n} p(a_n \mid S_{n-1})$.[2] An inferred ac-

[2]Since greedy parsing does not require probabilistic semantics for the action model—the softmax normalizer does not need to be evaluated—non-probabilistic training, such as with hinge loss (SVMs), is a common alternative, including in some of the cited work.

tion sequence $a_{1:n}$ determines the resulting parse tree (edge set) y; the relationship can be denoted as $y(a_{1:n})$.

2.2 Transition sampling

In this work, we propose to analyze the full joint posterior $p(y \mid x)$, and use *transition sampling*, a very simple forward/ancestral sampling algorithm,[3] to draw parse tree samples from that distribution. To parse a sentence, we run the automaton stochastically, sampling the action probability in step (A). This yields one action sequence $a_{1:n}$ from the full joint distribution of action sequences, and therefore a parse $y(a_{1:n})$ from the distribution of parses. We can obtain as many parse samples as desired by running the transition sampler S times, yielding a collection (multiset) of parse structures $\{y^{(s)} \mid s \in \{1..S\}\}$, where each $y^{(s)} \sim p(y \mid x)$ is a full dependency parse tree.[4] Runtime to draw one parse sample is very similar to the greedy algorithm's runtime. We denote the set of unique parses in the sample $\tilde{\mathcal{Y}}(x)$.

We implement a transition sampler by modifying an implementation of Chen and Manning's multilayer perceptron transition-based parser[5] and use it for all subsequent experiments.

2.3 MC-MAP single parse prediction

One minor use of transition sampling is a method for predicting a single parse, by selecting the most probable (common) parse tree in the sample,

$$\hat{y}^{\text{MC-MAP}} = \arg\max_{y \in \tilde{\mathcal{Y}}} \tilde{p}(y \mid x) \quad (3)$$

$$= \arg\max_{y \in \tilde{\mathcal{Y}}} \frac{c(y)}{S} \quad (4)$$

where $\tilde{p}(y \mid x)$ denotes the Monte Carlo estimate of a parse's probability, which is proportional to how many times it appears in the sample: $c(y) \equiv \sum_s^S 1\{y = y^{(s)}\}$. Note that $\tilde{p}(y \mid x)$ correctly accounts for the case of an ambiguous transition system where multiple different action sequences can yield the same tree—i.e., $y(a_{1:n})$ is not one-to-one—since the transition sampler can sample the multiple different paths.

This "MC-MAP" method is asymptotically guaranteed to find the model's most probable parse ($\arg\max_y p(y \mid x)$) given enough samples.[6] By contrast, greedy decoding and beam search have no theoretical guarantees. MC-MAP's disadvantage is that it may require a large number of samples, depending on the difference between the top parse's probability compared to other parses in the domain.

2.4 Monte Carlo Syntax Marginal (MCSM) inference for structure queries

Beyond entire tree structures, parse posteriors also define marginal probabilities of particular events in them. Let $f(y) \to \{0, 1\}$ be a boolean-valued *structure query* function of a parse tree—for example, whether the tree contains a particular edge:

$$f(y) = 1\{\text{dobj(kill,Smith)} \in y\}$$

or more complicated structures, such as a length-2 dependency path:

$$f(y) = 1\{\text{nsubj(kill, cop)} \wedge \text{dobj(kill, Smith)} \in y\}.$$

More precisely, these queries are typically formulated to check for edges between specific tokens, and may check tokens' string forms.

Although $f(y)$ is a deterministic function, since the parsing model is uncertain of the correct parse, we find the *marginal probability*, or expectation, of a structure query by integrating out the posterior parse distribution—that is, the predicted probability that the parse has the property in question:

$$p(f(y) \mid x) = \sum_{y \in \mathcal{Y}(x)} f(y)\, p(y \mid x) \quad (5)$$

$$\approx \tilde{p}(f(y) \mid x) = \sum_{y \in \tilde{\mathcal{Y}}(x)} f(y) \frac{c(y)}{S}. \quad (6)$$

Eq. 5 is the expectation with regard to the model's true probability distribution (p) over parses from the domain of all possible parse trees $\mathcal{Y}(x)$ for a sentence, while Eq. 6 is a Monte Carlo estimate of the query's marginal probability—the fraction of parse tree samples where the structure query is true. We use this simple method for all inference

[3] "Ancestral" refers to a directed Bayes net (e.g. Barber (2012)) of action decisions, each conditioned on the full history of previous actions—not ancestors in a parse tree.

[4] Dyer et al. (2016) use the same algorithm to draw samples from a transition-based constituency parsing model, as an importance sampling proposal to support parameter learning and single-tree inference.

[5] CoreNLP 3.8.0 with its 'english_UD' pretrained model.

[6] This holds since the Monte Carlo estimated probability of any tree converges to its true probability, according to, e.g., Hoeffding's inequality or the central limit theorem. Thus, with enough samples, the tree with the highest true probability will have estimated probability higher than any other tree's.

in this work, though importance sampling (Dyer et al., 2016), particle filters (Buys and Blunsom, 2015), or diverse k-best lists (Zhang and McDonald, 2014) could support more efficient inference in future work.

2.5 Probabilistic inference for dependencies: related work

Our transition sampling method aims to be an easy-to-implement algorithm for a highly performant class of dependency models, that conducts exact probabilistic inference for arbitrary structure queries in a reasonable amount of time. A wide range of alternative methods have been proposed for dependency inference that cover some, but perhaps not all, of these goals.

For transition-based parsing, beam search is a commonly used inference method that tries to look beyond a single structure. Beam search can be used to yield an approximate K-best list by taking resulting structures on the beam, though there are no theoretical guarantees about the result, and runtime is no better than the transition sampler.[7] Finkel et al. (2006) further discuss tradeoffs between beam search and sampling, and find they give similar performance when propagating named entity recognition and PCFG parse information to downstream tasks.

Graph-based parsers are the major alternative modeling paradigm for dependency parsing; instead of a sequence of locally normalized decisions, they directly parameterize an entire tree's globally normalized probability. Parse samples could be drawn from a graph-based model via Markov chain Monte Carlo (Zhang et al., 2014), which is asymptotically correct, but may require a large amount of time to obtain non-autocorrelated parses. A range of methods address inference for specific queries in graph-based models—for example, edge marginals for edge-factored models via the matrix-tree theorem (Koo et al., 2007), or approximate marginals with loopy belief propagation (Smith and Eisner, 2008).[8] By contrast, our method is guaranteed to give correct marginal inferences for arbitrary, potentially long-distance, queries.

Given the strong performance of graph-based parsers in the single-structure prediction setting (e.g. Zeman et al. (2017); Dozat et al. (2017)), it may be worthwhile to further explore probabilistic inference for these models. For example, Niculae et al. (2018) present an inference algorithm for a graph-based parsing model that infers a weighted, sparse set of highly-probable parse trees, and they illustrate that it can infer syntactic ambiguities similar to Figure 1.

Dynamic programming for dependency parsing, as far as we are aware, has only been pursued for single-structure prediction (e.g. Huang and Sagae (2010)), but in principle could be generalized to calculate local structure query marginals via an inside-outside algorithm, or to sample entire structures through an inside-outside sampler (Eisner, 2016), which Finkel et al. (2006) use to propagate parse uncertainty for downstream analysis.

3 Exploratory error analysis via whole-tree entropy calculations

In this section we directly explore the model's intrinsic uncertainty, while §5 conducts a quantitative analysis of model uncertainty compared to gold standard structures. Parse samples are able to both pass on parse uncertainty and yield useful insights that typical error analysis approaches cannot. For a sentence x, we can calculate the *whole-tree entropy*, the model's uncertainty of whole-tree parse frequencies in the samples:

$$\begin{aligned} H(p) &= -\sum_{y \in \mathcal{Y}(x)} p(y \mid x) \log p(y \mid x) \\ &\approx H(\tilde{p}) = -\sum_{y \in \tilde{\mathcal{Y}}(x)} \frac{c(y)}{S} \log \frac{c(y)}{S}. \end{aligned} \tag{7}$$

Since this entropy estimate is only based on an S-sample approximation of p, it is upper bounded at $\log(S)$ in the case of a uniform MC distribution. Another intuitive measure of uncertainty is simply the number of unique parses, that is, the cardinality of the MC distribution's domain ($|\tilde{\mathcal{Y}}|$); this quantity is not informative for the true distribution p, but in the MC distribution it is intuitively upper bounded by S.[9]

[7] Loosely, if it takes N transitions to complete a parse, and B possible actions at each transition must be evaluated, our method evaluates KNB actions to obtain K trees. Beam search evaluates a similar number of actions when using a K-sized beam, but also requires non-parallelizable management of the beam's priority queue.

[8] These papers infer marginals to support parameter learning, but we are not aware of previous work that directly analyzes or uses dependency parse marginals.

[9] Shannon entropy, domain support cardinality, and top probability ($\max_{y \in \tilde{Y}} \tilde{p}(y)$), which we show in Table 1, are all instances of the more general Renyi entropy (Smith and Eisner, 2007).

Sentence	Domain Size	Top 3 Freq.	Entropy
In Ramadi , there was a big demonstration .	3	[98, 1, 1]	0.112
US troops there clashed with guerrillas in a fight that left one Iraqi dead .	40	[33, 11, 6]	2.865
The sheikh in wheel - chair has been attacked with a F - 16 - launched bomb .	98	[2, 2, 1]	4.577

Table 1: Example sentences from the UD development set and summaries of their Monte Carlo parse distributions. *Domain Size* gives $|\tilde{\mathcal{Y}}_{100}|$, the number of unique parse structures in 100 samples. *Top 3 Freq.* gives the frequencies of the 3 most probable structures in $\tilde{\mathcal{Y}}_{100}$. *Entropy* is calculated according to Eq. 7; its upper bound is $\log(100) = 4.605$.

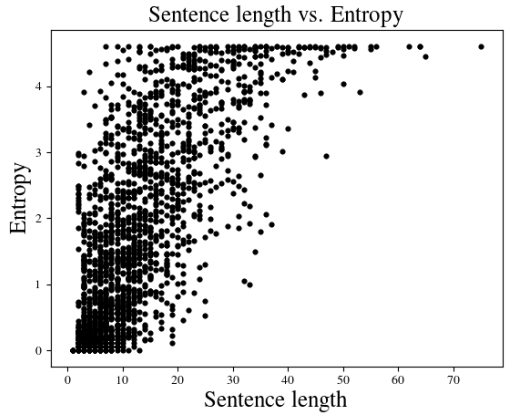


Figure 2: Length of sentences (in number of tokens in UD development set) against entropy (100 samples per sentence).

We run our dependency sampler on the 2002 sentences in the Universal Dependencies 1.3 English Treebank development set, generating 100 samples per sentence; Table 1 shows example sentences along with $|\tilde{\mathcal{Y}}|$ and entropy statistics for each sentence. We find that in general, as sentence length increases, so does the entropy of the parse distribution (Fig. 2). Moreover, we find that entropy is a useful diagnostic tool. For example, 7% of sentences in the UD development corpus with fewer than 15 tokens and $H(\tilde{p}) \geq 2$ exhibit uncertainty around the role of '-' (compare *Sciences - principally biology* and *thought-provoking*), and another 7% of such sentences exhibit uncertainty around 's' (potentially representing a plural or a possessive).

4 Monte Carlo Syntax Marginals for partial dependency parsing

Here we examine the utility of marginal inference for predicting parts of dependency parses, using the UD 1.3 Treebank's English development set to evaluate.[10]

4.1 Greedy decoding

Using its off-the-shelf pretrained model with greedy decoding, the CoreNLP parser achieves 80.8% labeled attachment score (LAS). LAS is equivalent to both the precision and recall of predicting (rel,gov,child) triples in the parse tree.[11]

4.2 Minimum Bayes risk (MBR) decoding

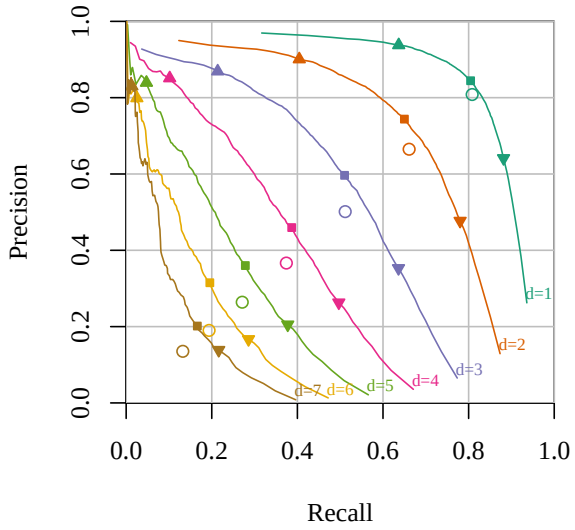
A simple way to use marginal probabilities for parse prediction is to select, for each token, the governor and relation that has the highest marginal probability. This method gives a minimum Bayes risk (MBR) prediction of the parse, minimizing the model's expected LAS with regards to local uncertainty; similar MBR methods have been shown to improve accuracy in tagging and constituent parsing (e.g. Goodman (1996); Petrov and Klein (2007)). This method yields 81.4% LAS, outperforming greedy parsing, though it may yield a graph that is not a tree.

4.3 Syntax marginal inference for dependency paths

An alternative view on dependency parsing is to consider what structures are needed for downstream applications. One commonly used parse substructure is the *dependency path* between two words, which is widely used in unsupervised lexical semantics (Lin and Pantel, 2001), distantly supervised lexical semantics (Snow et al., 2005), relation learning (Riedel et al., 2013), and supervised semantic role labeling (Hacioglu, 2004; Das et al., 2014), as well as applications in economics (Ghose et al., 2007), political science (O'Connor et al., 2013), biology (Fundel et al., 2006), and the humanities (Bamman et al., 2013, 2014).

[10]UD 1.3 is the UD version that this parsing model is most similar to: https://mailman.stanford.edu/pipermail/parser-user/2017-November/003460.html

[11]LAS is typically defined as proportion of tokens whose governor (and relation on that governor-child edge) are correctly predicted; this is equivalent to precision and recall of edges if all observed tokens are evaluated. If, say, punctuation is excluded from evaluation, this equivalence does not hold; in this work we always use all tokens for simplicity.



Path Len.	Greedy F1	Marginal Max F1	Marginal Thresh.	MC-MAP F1
1	◯ 0.808	■ 0.824	0.45	0.807
2	◯ 0.663	■ 0.694	0.44	0.660
3	◯ 0.506	■ 0.550	0.34	0.501
4	◯ 0.370	■ 0.420	0.28	0.363
5	◯ 0.268	■ 0.314	0.25	0.262
6	◯ 0.192	■ 0.241	0.25	0.188
7	◯ 0.134	■ 0.182	0.17	0.131

▲ Conf≥0.9 ▼ Conf≥0.1

Figure 3: Precision-recall analysis against gold UD 1.3 English dev (§4.1) set for all dependency paths of lengths 1 to 7. **Left:** Each path length is a different color (1 in top-right, 7 in bottom-left), with greedy performance (◯) as well as the marginal path predictions' PR curve, with points at confidence thresholds 0.9 (▲), 0.1 (▼), and where F1 is highest (■). **Right:** F1 for each method, as well as the confidence threshold for the marginal PR curve's max-F1 point. For path length 1, Greedy and MC-MAP F1 are the same as labeled attachment score (LAS).

In this work, we consider a dependency path to be a set of edges from the dependency parse; for example, a length-2 path $p = \{\text{nsubj}(3,1), \text{dobj}(3,4)\}$ connects tokens 1 and 4. Let $\mathcal{P}_d(y)$ be the set of all length-d paths from a parse tree y.[12] Figure 3's "Greedy" table column displays the F-scores for the precision and recall of retrieving $\mathcal{P}_d(y^{(gold)})$ from the prediction $\mathcal{P}_d(y^{(greedy)})$ for a series of different path lengths. $\mathcal{P}_1$ gives individual edges, and thus is the same as LAS (80.8%). Longer length paths see a rapid decrease in performance; even length-2 paths are retrieved with only $\approx 66\%$ precision and recall.[13] We are not aware of prior work that evaluates dependency parsing beyond single edge or whole sentence accuracy.

We define *dependency path prediction* as the task of predicting a set of dependency paths for a sentence; the paths do not necessarily have to come from the same tree, nor even be consistent with a single syntactic analysis. We approach this task with our Monte Carlo syntax marginal method, by predicting paths from the transition sampling parser. Here we treat each possible path as a structure query (§2.4) and return all paths whose marginal probabilities are at least threshold t. Varying t trades off precision and recall.

We apply this method to 100 samples per sentence in the UD treebank. When we take all length-1 paths that appear in every single sample (i.e., estimated marginal probability 1.0), precision greatly increases to 0.969, while recall drops to 0.317 (the top-left point on Figure 3's **teal** length-1 curve.) We can also accommodate applications which may prefer to have a higher recall: predicting all paths with at least 0.01 probability results in 0.936 recall (the bottom-right point on the curve in Figure 3).[14]

This marginal path prediction method dominates the greedy parser: for length-1 paths, there are points on the marginal decoder's PR curve that achieve both higher precision and recall than the greedy decoder, giving F1 of 82.4% when accepting all edges with marginal probability at least 0.45. Furthermore, these advantages are more prominent for longer dependency paths. For example, for length-3 paths, the greedy parser only achieves 50.6% F1, while the marginal parser im-

[12] Path construction may traverse both up and down directed edges; we represent a path as an edge set to evaluate its existence in a parse. A path may not include the same vertex twice. The set of all paths for a parse includes all paths from all pairs of vertexes (observed tokens and ROOT).

[13] For length 1 paths, precision and recall are identical; this does not hold for longer paths, though precision and recall from a single parse prediction are similar.

[14] The 6.4% of gold-standard edges with predicted 0 probability often correspond to inconsistencies in the formalism standards between the model and UD; for example, 0.7% of the gold edges are 'name' relations among words in a name, which the model instead analyzes as 'compound'. Inspecting gold edges' marginal probabilities helps error analysis, since when one views a single predicted parse, it is not always clear whether observed errors are systematic, or a fluke for that one instance.

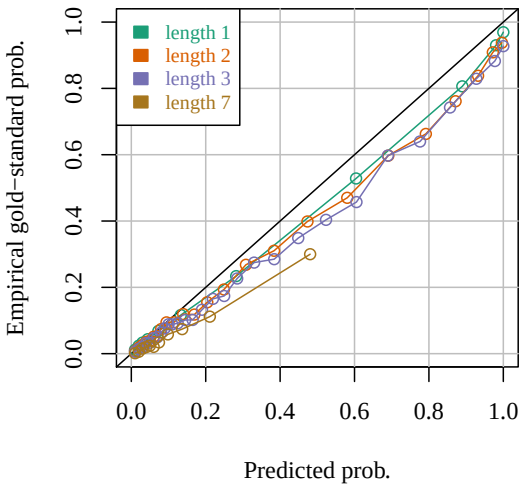


Figure 4: Calibration curves for marginal predictions for several path lengths. Predictions below the $y = x$ line indicate overconfidence. Points denote the average predicted probability versus empirical (gold) probability among predicted paths within each dynamically allocated bin.

proves a bit to 55.0% F1; strikingly, it is possible to select high-confidence paths to get much higher 90.1% precision (at recall 11.6%, with confidence threshold $t = 0.95$). Figure 3 also shows the precision/recall points on each curve for thresholds $t = 0.9$ and $t = 0.1$.

We also evaluated the MC-MAP single-parse prediction method (§2.3), which slightly, but consistently, underperforms the greedy decoder at all dependency lengths. More work is required to understand whether this is is an inference or modeling problem: for example, we may not have enough samples to reliably predict a high-probability parse; or, as some previous work finds in the context of beam search, the label bias phenomenon in this type of locally-normalized transition-based parser may cause it to assign higher probability to non-greedy analyses that in fact have lower linguistic quality (Zhang and Nivre, 2012; Andor et al., 2016).

5 Calibration

The precision-recall analysis shows that the predicted marginal probabilities are meaningful in a ranking sense, but we can also ask whether they are meaningful in a sense of *calibration*: predictions are calibrated if, among all structures with predicted probability $q \pm \epsilon$, they exist in the gold parses with probability q. That is, predictions with confidence q have precision q.[15] If probabilities are calibrated, that implies expectations with regard to their distribution are unbiased, and may also justify intuitive interpretations of probabilities in exploratory analysis (§3). Calibration may also have implications for joint inference, EM, and active learning methods that use confidence scores and confidence-based expectations.

We apply Nguyen and O'Connor (2015)'s adaptive binning method to analyze the calibration of structure queries from an NLP system, by taking the domain of all seen length-d paths from the 100 samples' parse distribution for the treebank, grouping by ranges of predicted probabilities to have at least 5000 paths per bin, to ensure stability of the local precision estimate.[16]

We find that probabilities are reasonably well calibrated, if slightly overconfident—Figure 4 shows the average predicted probability per bin, compared to how often these paths appear in the gold standard (local precision). For example, for edges (length-1 paths), predictions near 60% confidence (the average among predictions in range $[0.42, 0.78]$) correspond to edges that are actually in the gold standard tree only 52.8% of the time. The middle confidence range has worse calibration error, and longer paths perform worse. Still, this level of calibration seems remarkably good, considering there was no attempt to re-calibrate predictions (Kuleshov and Liang, 2015) or to use a model that specifically parameterizes the energy of dependency paths (Smith and Eisner, 2008; Martins et al., 2010)—these predictions are simply a side effect of the overall joint model for incremental dependency parsing.

6 Probabilistic rule-based IE: classifying police fatalities

Supervised learning typically gives the most accurate information extraction or semantic parsing systems, but for many applications where train-

[15] This is a *local* precision, as opposed to the more usual tail probability of measuring precision of all predictions *higher* than some t—the integral of local precision. For example, Figure 3's length-1 $t = 0.9$ precision of 0.942 (△) is the average y value of several rightmost bins in Figure 4. This contrast corresponds to Efron (2010)'s dichotomy of local versus global false discovery rates.

[16] This does not include gold-standard paths with zero predicted probability. As Nguyen and O'Connor found for sequence tagging and coreference, we find the prediction distribution is heavily skewed to near 0 and 1, necessitating adaptive bins, instead of fixed-width bins, for calibration analysis (Niculescu-Mizil and Caruana, 2005; Bennett, 2000).

ing data is scarce, Chiticariu et al. (2013) argue that rule-based systems are useful and widespread in practice, despite their neglect in contemporary NLP research. Syntactic dependencies are a useful abstraction with which to write rule-based extractors, but they can be brittle due to errors in the parser. We propose to integrate over parse samples to infer a *marginal* probability of a rule match, increasing robustness and allowing for precision-recall tradeoffs.

6.1 Police killings victim extraction

We examine the task of extracting the list of names of persons killed by police from a test set of web news articles in Sept–Dec 2016. We use the dataset released by Keith et al. (2017), consisting of 24,550 named entities $e \in \mathcal{E}$ and sentences from noisy web news text extractions (that can be difficult to parse), each of which contains at least one e (on average, 2.8 sentences/name) as well as keywords for both police and killing/shooting. The task is to classify whether a given name is a person who was killed by police, given 258 gold-standard names that have been verified by journalists.

6.2 Dependency rule exractor

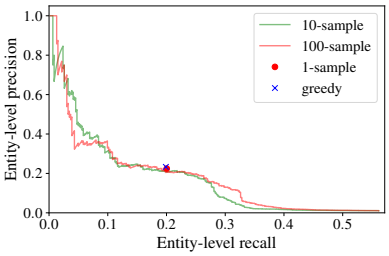
Keith et al. present a baseline rule-based method that uses Li and Ji (2014)'s off-the-shelf RPI-JIE ACE event parser to extract (event type, agent, patient) tuples from sentences, and assigns $f_{\text{JIE}}(x_i, e) = 1$ iff the event type was a killing, the agent's span included a police keyword, and the patient was the candidate entity e. An entity is classified as a victim if at least one sentence is classified as true, resulting in a 0.17 F1 score (as reported in previous work).[17]

We define a similar syntactic dependency rule system using a dependency parse as input: our extractor $f(x, e, y)$ returns 1 iff the sentence has a killing keyword k,[18] which both

1. has an agent token a (defined as, governed by *nsubj* or *nmod*) which is a police keyword, or a has a (*amod* or *compound*) modifier that is a police keyword; and,
2. has a patient token p (defined as, governed by *nsubjpass* or *dobj*) contained in the candidate name e's span.

[17]This measures recall of the entire gold-standard victim database, though the corpus only includes 57% of the victims.

[18]Police and killing/shooting keywords are from Keith et al.'s publicly released software.



Method	F1
RPI-JIE	0.170
Greedy	0.215
1 samp.	0.212
10 samp.	0.219
100 samp.	**0.222**

Figure 5: **Left:** Rule-based entity precision and recall for police fatality victims, with greedy parsing and Monte Carlo inference. **Right:** F1 scores for RPI-JIE, Greedy, and 1-sample methods, and maximum F1 on PR curve for probabilistic (multiple sample) inference.

Applying this $f(x, e, y)$ classifier to greedy parser output, it performs better than the RPI-JIE-based rules (Figure 5, right), perhaps because it is better customized for the particular task.

Treating f as a structure query, we then use our Monte Carlo marginal inference (§2) method to calculate the probability of a rule match for each sentence—that is, the fraction of parse samples where $f(x, e, y^{(s)})$ is true—and infer the entity's probability with the *noisy-or* formula (Craven and Kumlien, 1999; Keith et al., 2017). This gives soft classifications for entities.

6.3 Results

The Monte Carlo method achieves slightly higher F1 scores once there are at least 10 samples (Fig. 5, right). More interestingly, the soft entity-level classifications also allow for precision-recall tradeoffs (Fig. 5, left), which could be used to prioritize the time of human reviewers updating the victim database (filter to higher precision), or help ensure victims are not missed (with higher recall). We found the sampling method retrieved several true-positive entities where only a single sentence had a non-zero rule prediction at probability 0.01—that is, the rule was only matched in one of 100 sampled parses. Since current practitioners are already manually reviewing millions of news articles to create police fatality victim databases, the ability to filter to high recall—even with low precision—may be useful to help ensure victims are not missed.

6.4 Supervised learning

Sampling also slightly improves supervised learning for this problem. We modify Keith et al.'s logistic regression model based on a dependency

path feature vector $f(x_i, y)$, instead creating feature vectors that average over multiple parse samples ($E_{\tilde{p}(y)}[f(x_i, y)]$) at both train and test time. With the greedy parser, the model results in 0.229 F1; using 100 samples slightly improves performance to 0.234 F1.

7 Semantic role assignment

Semantic role labeling (SRL), the task to predict argument structures (Gildea and Jurafsky, 2002), is tightly tied to syntax, and previous work has found it beneficial to conduct it with joint inference with constituency parsing, such as with top-k parse trees (Toutanova et al., 2008) or parse tree samples (Finkel et al., 2006). Since §4 shows that Monte Carlo marginalization improves dependency edge prediction, we hypothesize dependency sampling could improve SRL as well.

SRL includes both identifying argument spans, and assigning spans to specific semantic role labels (argument types). We focus on just the second task of semantic role assignment: assuming argument spans are given, to predict the labels. We experiment with English OntoNotes v5.0 annotations (Weischedel et al., 2013) according to the CoNLL 2012 test split (Pradhan et al., 2013). We focus only on predicting among the five core arguments (Arg0 through Arg4) and ignore spans with gold-standard adjunct or reference labels. We fit a separate model for each predicate[19] among the 2,160 predicates that occur at least once in both the training and test sets (115,811 and 12,216 sentences respectively).

Our semantic model of label $z_t \in \{A0..A4\}$ for argument head token t and predicate token p, $p_{\text{sem}}(z_t \mid p, y)$, is simply the conditional probability of the label, conditioned on y's edge between t and p if one exists.[20] (If they are not directly connected, the model instead conditions on a 'no edge' feature.) Probabilities are maximum likelihood estimates from the training data's (predicate, argument label, path) counts, from either greedy parses, or averaged among parse samples. To predict at test time, the greedy parsing model simply uses $p(z_t \mid p, y^{(greedy)})$. The Monte Carlo model, by contrast, treats it as a directed joint model and marginalizes over syntactic analyses:

$$p_{MC}(z_t \mid p, x) = \sum_{y \in \tilde{\mathcal{Y}}(x)} p_{\text{sem}}(z_t \mid p, y)\, \tilde{p}_{\text{syn}}(y \mid x).$$

Method	Accuracy
Baseline (most common)	0.393
Greedy	0.496
MCSM method, 100-samples	**0.529**

Table 2: Semantic role assignment accuracy on English OntoNotes v5.0. The baseline is for each unique predicate, predict the argument that was seen the most at training time.

The baseline accuracy of predicting the predicate's most common training-time argument label yields 0.393 accuracy, and the greedy parser performs at 0.496. The Monte Carlo method (with 100 samples) improves accuracy to 0.529 (Table 2). Dependency samples' usefulness in this limited case suggests they may help systems that use dependency parses more broadly for SRL (Hacioglu, 2004; Das et al., 2014).

8 Conclusion

In this work, we introduce a straightforward algorithm for sampling from the full joint distribution of a transition-based dependency parser. We explore using these parse samples to discover both parsing error and structural ambiguities. Moreover, we find that our Monte Carlo syntax marginal method not only dominates the greedy method for dependency path prediction (especially for longer paths), but also allows for control of precision-recall tradeoffs. Propagating dependency uncertainty can potentially help a wide variety of semantic analysis and information extraction tasks.

Acknowledgments

The authors would like to thank Rajarshi Das, Daniel Cohen, Abe Handler, Graham Neubig, Emma Strubell, and the anonymous reviewers for their helpful comments.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav

[19] That is, for each unique (lemma, framesetID) pair, such as (view, view-02).

[20] The dataset's argument spans must be reconciled with predicted parse structures to define the argument head t; 90% of spans are consistent with the greedy parser in that all the span's tokens have the same highest ancestor contained with the span, which we define as the argument head. For inconsistent cases, we select the largest subtree (that is, highest within-span ancestor common to the largest number of the span's tokens). It would be interesting to modify the sampler to restrict to parses that are consistent with the span, as a form of rejection sampling.

Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of ACL*.

David Bamman, Brendan O'Connor, and Noah A. Smith. 2013. Learning latent personas of film characters. In *Proceedings of ACL*.

David Bamman, Ted Underwood, and Noah A. Smith. 2014. A Bayesian mixed effects model of literary character. In *Proceedings of ACL*.

David Barber. 2012. *Bayesian Reasoning and Machine Learning*. Cambridge University Press.

Paul N. Bennett. 2000. Assessing the calibration of naive Bayes' posterior estimates. Technical report, Carnegie Mellon University.

Bernd Bohnet, Ryan McDonald, Emily Pitler, and Ji Ma. 2016. Generalized transition-based dependency parsing via control parameters. In *Proceedings of ACL*. http://www.aclweb.org/anthology/P16-1015.

Razvan Bunescu. 2008. Learning with probabilistic features for improved pipeline models. In *Proceedings of EMNLP*. http://www.aclweb.org/anthology/D08-1070.

Jan Buys and Phil Blunsom. 2015. A bayesian model for generative transition-based dependency parsing. In *Proceedings of the Third International Conference on Dependency Linguistics (Depling 2015)*. Uppsala, Sweden. http://www.aclweb.org/anthology/W15-2108.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of EMNLP*. http://www.aclweb.org/anthology/D14-1082.

Laura Chiticariu, Yunyao Li, and Frederick R. Reiss. 2013. Rule-based information extraction is dead! long live rule-based information extraction systems! In *Proceedings of EMNLP*. http://www.aclweb.org/anthology/D13-1079.

Mark Craven and Johan Kumlien. 1999. Constructing biological knowledge bases by extracting information from text sources. In *ISMB*. pages 77–86.

Dipanjan Das, Desai Chen, Andre F. T. Martins, Nathan Schneider, and Noah A. Smith. 2014. Frame-semantic parsing. *Computational Linguistics* .

Timothy Dozat, Peng Qi, and Christopher D Manning. 2017. Stanford's graph-based neural dependency parser at the conll 2017 shared task. *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies* pages 20–30.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of ACL*. http://www.aclweb.org/anthology/P15-1033.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of NAACL*. http://www.aclweb.org/anthology/N16-1024.

Bradley Efron. 2010. *Large-Scale Inference: Empirical Bayes Methods for Estimation, Testing, and Prediction*. Cambridge University Press, 1st edition.

Jason Eisner. 2016. Inside-outside and forward-backward algorithms are just backprop (tutorial paper). In *Proceedings of the Workshop on Structured Prediction for NLP*. Association for Computational Linguistics, Austin, TX, pages 1–17. http://aclweb.org/anthology/W16-5901.

Jenny Rose Finkel, Christopher D. Manning, and Andrew Y. Ng. 2006. Solving the problem of cascading errors: Approximate Bayesian inference for linguistic annotation pipelines. In *Proceedings of EMNLP*.

Katrin Fundel, Robert Küffner, and Ralf Zimmer. 2006. Relex—relation extraction using dependency parse trees. *Bioinformatics* 23(3):365–371.

Anindya Ghose, Panagiotis Ipeirotis, and Arun Sundararajan. 2007. Opinion mining using econometrics: A case study on reputation systems. In *Proceedings of ACL*. Prague, Czech Republic. http://www.aclweb.org/anthology/P07-1053.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational linguistics* 28(3):245–288.

Joshua Goodman. 1996. Parsing algorithms and metrics. In *Proceedings of ACL*. https://doi.org/10.3115/981863.981887.

Kadri Hacioglu. 2004. Semantic role labeling using dependency trees. In *Proceedings of the 20th international conference on Computational Linguistics*. Association for Computational Linguistics.

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Uppsala, Sweden, pages 1077–1086. http://www.aclweb.org/anthology/P10-1110.

Katherine Keith, Abram Handler, Michael Pinkham, Cara Magliozzi, Joshua McDuffie, and Brendan O'Connor. 2017. Identifying civilians killed by police with distantly supervised entity-event extraction. In *Proceedings of EMNLP*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of ACL* .

Terry Koo, Amir Globerson, Xavier Carreras, and Michael Collins. 2007. Structured prediction models via the matrix-tree theorem. In *Proceedings of EMNLP*. http://www.aclweb.org/anthology/D/D07/D07-1015.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. Dependency parsing. *Synthesis Lectures on Human Language Technologies* .

Volodymyr Kuleshov and Percy S. Liang. 2015. Calibrated structured prediction. In *Advances in Neural Information Processing Systems*.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of ACL*. pages 402–412.

Dekang Lin and Patrick Pantel. 2001. DIRT – discovery of inference rules from text. In *Proceedings of KDD*.

Andre Martins, Noah Smith, Eric Xing, Pedro Aguiar, and Mario Figueiredo. 2010. Turbo parsers: Dependency parsing by approximate variational inference. In *Proceedings of EMNLP*. http://www.aclweb.org/anthology/D10-1004.

Khanh Nguyen and Brendan O'Connor. 2015. Posterior calibration and exploratory analysis for natural language processing models. In *Proceedings of EMNLP*. http://aclweb.org/anthology/D15-1182.

Vlad Niculae, André FT Martins, Mathieu Blondel, and Claire Cardie. 2018. SparseMAP: Differentiable sparse structured inference. *arXiv preprint arXiv:1802.04223* .

Alexandru Niculescu-Mizil and Rich Caruana. 2005. Predicting good probabilities with supervised learning. In *Proceedings of ICML*.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of LREC*.

Brendan O'Connor, Brandon Stewart, and Noah A. Smith. 2013. Learning to extract international relations from political context. In *Proceedings of ACL*.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of NAACL HLT 2007*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using ontonotes. In *CoNLL*. pages 143–152.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of NAACL*. Atlanta, Georgia. http://www.aclweb.org/anthology/N13-1008.

David Smith and Jason Eisner. 2008. Dependency parsing by belief propagation. In *Proceedings of EMNLP*. http://www.aclweb.org/anthology/D08-1016.

David A. Smith and Jason Eisner. 2007. Bootstrapping feature-rich dependency parsers with entropic priors. In *Proceedings of EMNLP*. http://www.aclweb.org/anthology/D/D07/D07-1070.

Rion Snow, Dan. Jurafsky, and Andrew Y. Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In *Advances in Neural Information Processing Systems*.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2008. A global joint model for semantic role labeling. *Computational Linguistics* 34(2):161–191.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. Ontonotes release 5.0 LDC2013T19. *Linguistic Data Consortium, Philadelphia, PA* .

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gokirmak, Anna Nedoluzhko, Silvie Cinkova, Jan Hajic jr., Jaroslava Hlavacova, Václava Kettnerová, Zdenka Uresova, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria dePaiva, Kira Droganova, Héctor Martínez Alonso, Çağr Çöltekin, Umut Sulubacak, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadová, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonca, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 2017. Conll 2017 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational

Linguistics, Vancouver, Canada, pages 1–19. http://www.aclweb.org/anthology/K17-3001.

Hao Zhang and Ryan McDonald. 2014. Enforcing structural diversity in cube-pruned dependency parsing. In *Proceedings of ACL*. http://www.aclweb.org/anthology/P14-2107.

Yuan Zhang, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2014. Greed is good if randomized: New inference for dependency parsing. In *Proceedings of EMNLP*. Doha, Qatar. http://www.aclweb.org/anthology/D14-1109.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of ACL*. http://www.aclweb.org/anthology/P11-2033.

Yue Zhang and Joakim Nivre. 2012. Analyzing the effect of global learning and beam-search on transition-based dependency parsing. In *Proceedings of COLING*. http://www.aclweb.org/anthology/C12-2136.

Neural Particle Smoothing for Sampling from Conditional Sequence Models

Chu-Cheng Lin and **Jason Eisner**
Center for Language and Speech Processing
Johns Hopkins University, Baltimore MD, 21218
{kitsing,jason}@cs.jhu.edu

Abstract

We introduce *neural particle smoothing*, a sequential Monte Carlo method for sampling annotations of an input string from a given probability model. In contrast to conventional particle filtering algorithms, we train a proposal distribution that *looks ahead* to the end of the input string by means of a right-to-left LSTM. We demonstrate that this innovation can improve the quality of the sample. To motivate our formal choices, we explain how our neural model and neural sampler can be viewed as low-dimensional but nonlinear approximations to working with HMMs over very large state spaces.

1 Introduction

Many structured prediction problems in NLP can be reduced to labeling a length-T input string $\mathbf{x}$ with a length-T sequence $\mathbf{y}$ of tags. In some cases, these tags are annotations such as syntactic parts of speech. In other cases, they represent actions that incrementally build an output structure: IOB tags build a chunking of the input (Ramshaw and Marcus, 1999), shift-reduce actions build a tree (Yamada and Matsumoto, 2003), and finite-state transducer arcs build an output string (Pereira and Riley, 1997).

One may wish to score the possible taggings using a recurrent neural network, which can learn to be sensitive to complex patterns in the training data. A globally normalized conditional probability model is particularly valuable because it quantifies uncertainty and does not suffer from label bias (Lafferty et al., 2001); also, such models often arise as the predictive conditional distribution $p(\mathbf{y} \mid \mathbf{x})$ corresponding to some well-designed generative model $p(\mathbf{x}, \mathbf{y})$ for the domain. In the neural case, however, inference in such models becomes intractable. It is hard to know what the model actually predicts and hard to compute gradients to improve its predictions.

In such intractable settings, one generally falls back on approximate inference or sampling. In the NLP community, beam search and importance sampling are common. Unfortunately, beam search considers only the approximate-top-k taggings from an exponential set (Wiseman and Rush, 2016), and importance sampling requires the construction of a good proposal distribution (Dyer et al., 2016).

In this paper we exploit the sequential structure of the tagging problem to do *sequential* importance sampling, which resembles beam search in that it constructs its proposed samples incrementally—one tag at a time, taking the actual model into account at every step. This method is known as particle filtering (Doucet and Johansen, 2009). We extend it here to take advantage of the fact that the sampler has access to the entire input string as it constructs its tagging, which allows it to look ahead or—as we will show—to use a neural network to approximate the effect of lookahead. Our resulting method is called *neural particle smoothing*.

1.1 What this paper provides

For $\mathbf{x} = x_1 \cdots x_T$, let $\mathbf{x}_{:t}$ and $\mathbf{x}_{t:}$ respectively denote the prefix $x_1 \cdots x_t$ and the suffix $x_{t+1} \cdots x_T$.

We develop *neural particle smoothing*—a sequential importance sampling method which, given a string $\mathbf{x}$, draws a sample of taggings $\mathbf{y}$ from $p_\theta(\mathbf{y} \mid \mathbf{x})$. Our method works for any conditional probability model of the quite general form[1]

$$p_\theta(\mathbf{y} \mid \mathbf{x}) \stackrel{\text{def}}{\propto} \exp G_T \tag{1}$$

where G is an *incremental stateful global scoring model* that recursively defines scores G_t of prefixes of $(\mathbf{x}, \mathbf{y})$ at all times $0 \le t \le T$:

$$G_t \stackrel{\text{def}}{=} G_{t-1} + g_\theta(\mathbf{s}_{t-1}, x_t, y_t) \quad \text{(with } G_0 \stackrel{\text{def}}{=} 0\text{)} \tag{2}$$

$$\mathbf{s}_t \stackrel{\text{def}}{=} f_\theta(\mathbf{s}_{t-1}, x_t, y_t) \quad \text{(with } \mathbf{s}_0 \text{ given)} \tag{3}$$

These quantities implicitly depend on $\mathbf{x}, \mathbf{y}$ and θ. Here $\mathbf{s}_t$ is the model's *state* after observing the pair of length-t prefixes $(\mathbf{x}_{:t}, \mathbf{y}_{:t})$. G_t is the *score-so-far*

[1] A model may require for convenience that each input end with a special end-of-sequence symbol: that is, $x_T = \text{EOS}$.

Proceedings of NAACL-HLT 2018, pages 929–941
New Orleans, Louisiana, June 1 - 6, 2018.

of this prefix pair, while $G_T - G_t$ is the *score-to-go*. The state $\mathbf{s}_t$ summarizes the prefix pair in the sense that the score-to-go depends only on $\mathbf{s}_t$ and the length-$(T-t)$ suffixes $(\mathbf{x}_{t:}, \mathbf{y}_{t:})$. The *local scoring function* g_θ and *state update function* f_θ may be any functions parameterized by θ—perhaps neural networks. We assume θ is fixed and given.

This model family is expressive enough to capture any desired $p(\mathbf{y} \mid \mathbf{x})$. Why? Take any distribution $p(\mathbf{x}, \mathbf{y})$ with this desired conditionalization (e.g., the true joint distribution) and factor it as

$$\begin{aligned}\log p(\mathbf{x}, \mathbf{y}) &= \textstyle\sum_{t=1}^{T} \log p(x_t, y_t \mid \mathbf{x}_{:t-1}, \mathbf{y}_{:t-1}) \\ &= \textstyle\sum_{t=1}^{T} \underbrace{\log p(x_t, y_t \mid \mathbf{s}_{t-1})}_{\text{use as } g_\theta(\mathbf{s}_{t-1}, x_t, y_t)} = G_T \quad (4)\end{aligned}$$

by making $\mathbf{s}_t$ include as much information about $(\mathbf{x}_{:t}, \mathbf{y}_{:t})$ as needed for (4) to hold (possibly $\mathbf{s}_t = (\mathbf{x}_{:t}, \mathbf{y}_{:t})$).[2] Then by defining g_θ as shown in (4), we get $p(\mathbf{x}, \mathbf{y}) = \exp G_T$ and thus (1) holds for each $\mathbf{x}$.

1.2 Relationship to particle filtering

Our method is spelled out in §4 (one may look now). It is a variant of the popular *particle filtering* method that tracks the state of a physical system in discrete time (Ristic et al., 2004). Our particular *proposal distribution* for y_t can be found in equations (5), (6), (25) and (26). It considers not only past observations $\mathbf{x}_{:t}$ as reflected in $\mathbf{s}_{t-1}$, but also future observations $\mathbf{x}_{t:}$, as summarized by the state $\bar{\mathbf{s}}_t$ of a right-to-left recurrent neural network $\bar{f}$ that we will train:

$$\hat{H}_t \stackrel{\text{def}}{=} h_\phi(\bar{\mathbf{s}}_{t+1}, x_{t+1}) + \hat{H}_{t+1} \quad (5)$$

$$\bar{\mathbf{s}}_t \stackrel{\text{def}}{=} \bar{f}_\phi(\bar{\mathbf{s}}_{t+1}, x_{t+1}) \quad \text{(with } \mathbf{s}_T \text{ given)} \quad (6)$$

Conditioning the distribution of y_t on future observations $\mathbf{x}_{t:}$ means that we are doing "smoothing" rather than "filtering" (in signal processing terminology). Doing so can reduce the bias and variance of our sampler. It is possible so long as $\mathbf{x}$ is provided in its entirety before the sampler runs—which is often the case in NLP.

1.3 Applications

Why sample from p_θ at all? Many NLP systems instead simply search for the *Viterbi sequence* $\mathbf{y}$ that maximizes G_T and thus maximizes $p_\theta(\mathbf{y} \mid \mathbf{x})$. If the space of states $\mathbf{s}$ is small, this can be done efficiently by dynamic programming (Viterbi, 1967); if not, then A^* may be an option (see §2). More common is to use an approximate method: beam search, or perhaps a sequential prediction policy trained with reinforcement learning. Past work has already shown how to improve these approximate search algorithms by conditioning on the future (Bahdanau et al., 2017; Wiseman and Rush, 2016).

Sampling is essentially a generalization of maximization: sampling from $\exp \frac{G_T}{\text{temperature}}$ approaches maximization as $\text{temperature} \to 0$. It is a fundamental building block for other algorithms, as it can be used to take expectations over the whole space of possible $\mathbf{y}$ values. For unfamiliar readers, Appendix E reviews how sampling is crucially used in minimum-risk decoding, supervised training, unsupervised training, imputation of missing data, pipeline decoding, and inference in graphical models.

2 Exact Sequential Sampling

To develop our method, it is useful to first consider exact samplers. Exact sampling is tractable for only some of the models allowed by §1.1. However, the form and notation of the exact algorithms in §2 will guide our development of approximations in §3.

An *exact sequential sampler* draws y_t from $p_\theta(y_t \mid \mathbf{x}, \mathbf{y}_{:t-1})$ for each $t = 1, \ldots, T$ in sequence. Then $\mathbf{y}$ is exactly distributed as $p_\theta(\mathbf{y} \mid \mathbf{x})$.

For each given $\mathbf{x}, \mathbf{y}_{:t-1}$, observe that

$$\begin{aligned}&p_\theta(y_t \mid \mathbf{x}, \mathbf{y}_{:t-1}) && (7)\\ &\propto p_\theta(\mathbf{y}_{:t} \mid \mathbf{x}) = \textstyle\sum_{\mathbf{y}_{t:}} p_\theta(\mathbf{y} \mid \mathbf{x}) && (8)\\ &\propto \textstyle\sum_{\mathbf{y}_{t:}} \exp G_T && (9)\\ &= \exp\,(G_t + \underbrace{\log \textstyle\sum_{\mathbf{y}_{t:}} \exp\,(G_T - G_t)}_{\text{call this } H_t}) && (10)\\ &= \exp\,(G_{t-1} + g_\theta(\mathbf{s}_{t-1}, x_t, y_t) + H_t) && (11)\\ &\propto \exp\,(g_\theta(\mathbf{s}_{t-1}, x_t, y_t) + H_t) && (12)\end{aligned}$$

Thus, we can easily construct the needed distribution (7) by normalizing (12) over all possible values of y_t. The challenging part of (12) is to compute H_t: as defined in (10), H_t involves a sum over exponentially many futures $\mathbf{y}_{t:}$. (See Figure 1.)

We chose the symbols G and H in homage to the A^* search algorithm (Hart et al., 1968). In that algorithm (which could be used to find the Viterbi sequence), g denotes the score-so-far of a partial solution $\mathbf{y}_{:t}$, and h denotes the optimal score-to-go. Thus, $g + h$ would be the score of the *best* sequence with prefix $\mathbf{y}_{:t}$. Analogously, our $G_t +$

[2] Furthermore, $\mathbf{s}_t$ could even depend on all of $\mathbf{x}$ (if $\mathbf{s}_0$ does), allowing direct expression of models such as stacked BiRNNs.

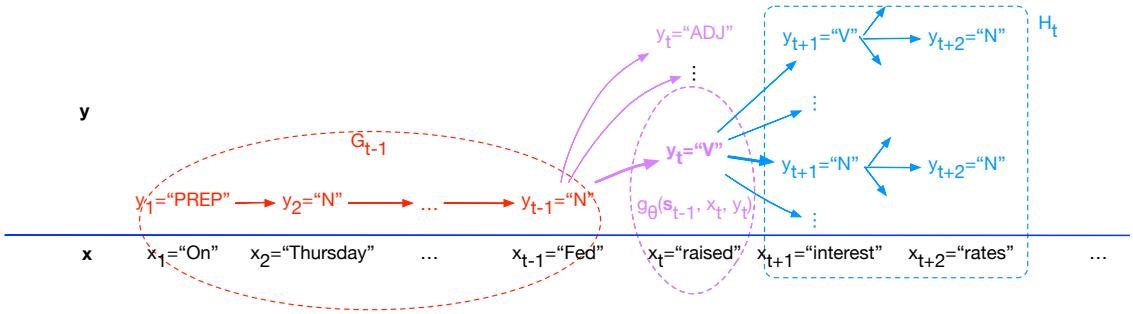


Figure 1: Sampling a single particle from a tagging model. $y_1, \ldots, y_{t-1}$ (orange) have already been chosen, with a total model score of G_{t-1}, and now the sampler is constructing a proposal distribution q (purple) from which the next tag y_t will be sampled. Each y_t is evaluated according to its contribution to G_t (namely g_θ) and its future score H_t (blue). The figure illustrates quantities used throughout the paper, beginning with exact sampling in equations (7)–(12). Our main idea (§3) is to *approximate* the H_t computation (a log-sum-exp over exponentially many sequences) when exact computation by dynamic programming is not an option. The form of our approximation uses a right-to-left recurrent neural network but is *inspired* by the exact dynamic programming algorithm.

H_t is the log of the total exponentiated scores of *all* sequences with prefix $\mathbf{y}_{:t}$. G_t and H_t might be called the *logprob-so-far* and *logprob-to-go* of $\mathbf{y}_{:t}$.

Just as A^* approximates h with a "heuristic" $\hat{h}$, the next section will approximate H_t using a neural estimate $\hat{H}_t$ (equations (5)–(6)). However, the specific form of our approximation is inspired by cases where H_t can be computed exactly. We consider those in the remainder of this section.

2.1 Exact sampling from HMMs

A *hidden Markov model* (HMM) specifies a normalized *joint* distribution $p_\theta(\mathbf{x}, \mathbf{y}) = \exp G_T$ over state sequence $\mathbf{y}$ and observation sequence $\mathbf{x}$,[3] Thus the posterior $p_\theta(\mathbf{y} \mid \mathbf{x})$ is proportional to $\exp G_T$, as required by equation (1).

The HMM specifically defines G_T by equations (2)–(3) with $\mathbf{s}_t = y_t$ and $g_\theta(\mathbf{s}_{t-1}, x_t, y_t) = \log p_\theta(y_t \mid y_{t-1}) + \log p_\theta(x_t \mid y_t)$.[4]

In this setting, H_t can be computed exactly by the *backward algorithm* (Rabiner, 1989). (Details are given in Appendix A for completeness.)

2.2 Exact sampling from OOHMMs

For sequence tagging, a weakness of (first-order) HMMs is that the model state $\mathbf{s}_t = y_t$ may contain little information: only the most recent tag y_t is remembered, so the number of possible model states $\mathbf{s}_t$ is limited by the vocabulary of output tags.

We may generalize so that the data generating process is in a latent state $u_t \in \{1, \ldots, k\}$ at each time t, and the observed y_t—along with x_t—is generated from u_t. Now k may be arbitrarily large. The model has the form

$$p_\theta(\mathbf{x}, \mathbf{y}) = \exp G_T \tag{13}$$
$$= \sum_{\mathbf{u}} \prod_{t=1}^{T} p_\theta(u_t \mid u_{t-1}) \cdot p_\theta(x_t, y_t \mid u_t)$$

This is essentially a pair HMM (Knudsen and Miyamoto, 2003) without insertions or deletions, also known as an "ϵ-free" or "same-length" probabilistic finite-state transducer. We refer to it here as an *output-output HMM* (OOHMM).[5]

Is this still an example of the general model architecture from §1.1? Yes. Since u_t is latent and evolves stochastically, it cannot be used as the state $\mathbf{s}_t$ in equations (2)–(3) or (4). However, we *can* define $\mathbf{s}_t$ to be the model's *belief state* after observing $(\mathbf{x}_{:t}, \mathbf{y}_{:t})$. The belief state is the posterior probability distribution over the underlying state u_t of the system. That is, $\mathbf{s}_t$ deterministically keeps track of all possible states that the OOHMM might be in—just as the state of a determinized FSA keeps track of all possible states that the original nondeterministic FSA might be in.

We may compute the belief state in terms of a vector of *forward probabilities* that starts at $\boldsymbol{\alpha}_0$,

$$(\boldsymbol{\alpha}_0)_u \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } u = \text{BOS (see footnote 4)} \\ 0 & \text{if } u = \text{any other state} \end{cases} \tag{14}$$

and is updated deterministically for each $0 < t \leq T$ by the *forward algorithm* (Rabiner, 1989):

$$(\boldsymbol{\alpha}_t)_u \stackrel{\text{def}}{=} \sum_{u'=1}^{k} (\boldsymbol{\alpha}_{t-1})_{u'} \cdot p_\theta(u \mid u') \cdot p_\theta(x_t, y_t \mid u) \tag{15}$$

[3] The HMM actually specifies a distribution over a pair of infinite sequences, but here we consider the marginal distribution over just the length-T prefixes.

[4] It takes $\mathbf{s}_0 = \text{BOS}$, a beginning-of-sequence symbol, so $p_\theta(y_1 \mid \text{BOS})$ specifies the initial state distribution.

[5] This is by analogy with the *input-output HMM* (IOHMM) of Bengio and Frasconi (1996), which defines $p(\mathbf{y} \mid \mathbf{x})$ directly and conditions the transition to u_t on x_t. The OOHMM instead defines $p(\mathbf{y} \mid \mathbf{x})$ by conditionalizing (13)—which avoids the *label bias* problem (Lafferty et al., 2001) that in the IOHMM, y_t is independent of future input $\mathbf{x}_{t:}$ (given the past input $\mathbf{x}_{:t}$).

$(\boldsymbol{\alpha}_t)_u$ can be interpreted as the logprob-so-far *if* the system is in state u after observing $(\mathbf{x}_{:t}, \mathbf{y}_{:t})$. We may express the update rule (15) by $\boldsymbol{\alpha}_t^\top = \boldsymbol{\alpha}_{t-1}^\top P$ where the matrix P depends on (x_t, y_t), namely $P_{u'u} \stackrel{\text{def}}{=} p_\theta(u \mid u') \cdot p_\theta(x_t, y_t \mid u)$.

The belief state $\mathbf{s}_t \stackrel{\text{def}}{=} [\![\boldsymbol{\alpha}_t]\!] \in \mathbb{R}^k$ simply normalizes $\boldsymbol{\alpha}_t$ into a probability vector, where $[\![\mathbf{u}]\!] \stackrel{\text{def}}{=} \mathbf{u}/(\mathbf{u}^\top \mathbf{1})$ denotes the *normalization operator*. The state update (15) now takes the form (3) as desired, with f_θ a normalized vector-matrix product:

$$\mathbf{s}_t^\top = f_\theta(\mathbf{s}_{t-1}, x_t, y_t) \stackrel{\text{def}}{=} [\![\mathbf{s}_{t-1}^\top P]\!] \tag{16}$$

As in the HMM case, we define G_t as the log of the generative prefix probability,

$$G_t \stackrel{\text{def}}{=} \log p_\theta(\mathbf{x}_{:t}, \mathbf{y}_{:t}) = \log \textstyle\sum_u (\boldsymbol{\alpha}_t)_u \tag{17}$$

which has the form (2) as desired if we put

$$\begin{aligned} g_\theta(\mathbf{s}_{t-1}, x_t, y_t) &\stackrel{\text{def}}{=} G_t - G_{t-1} \qquad (18)\\ &= \log \frac{\boldsymbol{\alpha}_{t-1}^\top P \mathbf{1}}{\boldsymbol{\alpha}_{t-1}^\top \mathbf{1}} = \log (\mathbf{s}_{t-1}^\top P \mathbf{1}) \end{aligned}$$

Again, exact sampling is possible. It suffices to compute (9). For the OOHMM, this is given by

$$\textstyle\sum_{\mathbf{y}_{t:}} \exp G_T = \boldsymbol{\alpha}_t^\top \boldsymbol{\beta}_t \tag{19}$$

where $\boldsymbol{\beta}_T \stackrel{\text{def}}{=} \mathbf{1}$ and the *backward algorithm*

$$\begin{aligned} (\boldsymbol{\beta}_t)_v &\stackrel{\text{def}}{=} p_\theta(\mathbf{x}_{t:} \mid u_t = u) \qquad (20)\\ &= \sum_{\mathbf{u}_{t:}, \mathbf{y}_{t:}} p_\theta(\mathbf{u}_{t:}, \mathbf{x}_{t:}, \mathbf{y}_{t:} \mid u_t = u)\\ &= \sum_{u'} \underbrace{p_\theta(u' \mid u) \cdot p(x_{t+1} \mid u')}_{\text{call this } \overline{P}_{uu'}} \cdot (\boldsymbol{\beta}_{t+1})_{u'} \end{aligned}$$

for $0 \le t < T$ uses dynamic programming to find the total probability of all ways to generate the future observations $\mathbf{x}_{t:}$. Note that $\boldsymbol{\alpha}_t$ is defined for a *specific prefix* $\mathbf{y}_{:t}$ (though it sums over all $\mathbf{u}_{:t}$), whereas $\boldsymbol{\beta}_t$ sums over *all suffixes* $\mathbf{y}_{t:}$ (and over all $\mathbf{u}_{t:}$), to achieve the asymmetric summation in (19).

Define $\bar{\mathbf{s}}_t \stackrel{\text{def}}{=} [\![\boldsymbol{\beta}_t]\!] \in \mathbb{R}^k$ to be a normalized version of $\boldsymbol{\beta}_t$. The $\boldsymbol{\beta}_t$ recurrence (20) can clearly be expressed in the form $\bar{\mathbf{s}}_t = [\![\overline{P} \bar{\mathbf{s}}_{t+1}]\!]$, much like (16).

2.3 The logprob-to-go for OOHMMs

Let us now work out the definition of H_t for OOHMMs (cf. equation (35) in Appendix A for HMMs). We will write it in terms of $\hat{H}_t$ from §1.2. Let us define $\hat{H}_t$ symmetrically to G_t (see (17)):

$$\hat{H}_t \stackrel{\text{def}}{=} \log \sum_u (\boldsymbol{\beta}_t)_u \;\; (= \log \mathbf{1}^\top \boldsymbol{\beta}_t) \tag{21}$$

which has the form (5) as desired if we put

$$h_\phi(\bar{\mathbf{s}}_{t+1}, x_{t+1}) \stackrel{\text{def}}{=} \hat{H}_t - \hat{H}_{t+1} = \log (\mathbf{1}^\top \overline{P} \bar{\mathbf{s}}_{t+1}) \tag{22}$$

From equations (10), (17), (19) and (21), we see

$$\begin{aligned} H_t &= \log \big(\sum_{\mathbf{y}_{t:}} \exp G_T \big) - G_t\\ &= \log \frac{\boldsymbol{\alpha}_t^\top \boldsymbol{\beta}_t}{(\boldsymbol{\alpha}_t^\top \mathbf{1})(\mathbf{1}^\top \boldsymbol{\beta}_t)} + \log (\mathbf{1}^\top \boldsymbol{\beta}_t)\\ &= \underbrace{\log \mathbf{s}_t^\top \bar{\mathbf{s}}_t}_{\text{call this } C_t} + \hat{H}_t \qquad (23) \end{aligned}$$

where $C_t \in \mathbb{R}$ can be regarded as evaluating the *compatibility* of the state distributions $\mathbf{s}_t$ and $\bar{\mathbf{s}}_t$.

In short, the generic strategy (12) for exact sampling says that for an OOHMM, y_t is distributed as

$$\begin{aligned} p_\theta(y_t \mid \mathbf{x}, \mathbf{y}_{:t-1}) &\propto \exp\, (g_\theta(\mathbf{s}_{t-1}, x_t, y_t) + H_t)\\ &\propto \exp\, \big(\underbrace{g_\theta(\mathbf{s}_{t-1}, x_t, y_t)}_{\text{depends on } \mathbf{x}_{:t}, \mathbf{y}_{:t}} + \underbrace{C_t}_{\text{on } \mathbf{x}, \mathbf{y}_{:t}} + \underbrace{\hat{H}_t}_{\text{on } \mathbf{x}_{t:}} \big)\\ &\propto \exp\, (g_\theta(\mathbf{s}_{t-1}, x_t, y_t) + C_t) \qquad (24) \end{aligned}$$

This is equivalent to choosing y_t in proportion to (19)—but we now turn to settings where it is infeasible to compute (19) exactly. There we will use the formulation (24) but approximate C_t. For completeness, we will also consider how to approximate $\hat{H}_t$, which dropped out of the above distribution (because it was the same for all choices of y_t) but may be useful for other algorithms (see §4).

3 Neural Modeling as Approximation

3.1 Models with large state spaces

The expressivity of an OOHMM is limited by the number of states k. The state $u_t \in \{1, \ldots, k\}$ is a bottleneck between the past $(\mathbf{x}_{:t}, \mathbf{y}_{:t})$ and the future $(\mathbf{x}_{t:}, \mathbf{y}_{t:})$, in that past and future are *conditionally independent* given u_t. Thus, the mutual information between past and future is at most $\log_2 k$ bits.

In many NLP domains, however, the past seems to carry substantial information about the future. The first half of a sentence greatly reduces the uncertainly about the second half, by providing information about topics, referents, syntax, semantics, and discourse. This suggests that an accurate HMM language model $p(\mathbf{x})$ would require *very large* k—as would a generative OOHMM model $p(\mathbf{x}, \mathbf{y})$ of *annotated* language. The situation is perhaps better for discriminative models $p(\mathbf{y} \mid \mathbf{x})$, since much of

the information for predicting $\mathbf{y}_{t:}$ might be available in $\mathbf{x}_{t:}$. Still, it is important to let $(\mathbf{x}_{:t}, \mathbf{y}_{:t})$ contribute enough additional information about $\mathbf{y}_{t:}$: even for short strings, making k too small (giving $\leq \log_2 k$ bits) may harm prediction (Dreyer et al., 2008).

Of course, (4) says that an OOHMM can express any joint distribution for which the mutual information is finite,[6] by taking k large enough for v_{t-1} to capture the relevant info from $(\mathbf{x}_{:t-1}, \mathbf{y}_{:t-1})$.

So why not just take k to be large—say, $k = 2^{30}$ to allow 30 bits of information? Unfortunately, evaluating G_T then becomes very expensive—both computationally and statistically. As we have seen, if we define $\mathbf{s}_t$ to be the belief state $[\![\boldsymbol{\alpha}_t]\!] \in \mathbb{R}^k$, updating it at each observation (x_t, y_t) (equation (3)) requires multiplication by a $k \times k$ matrix P. This takes time $O(k^2)$, and requires enough data to learn $O(k^2)$ transition probabilities.

3.2 Neural approximation of the model

As a solution, we might hope that for the inputs $\mathbf{x}$ observed in practice, the very high-dimensional belief states $[\![\boldsymbol{\alpha}_t]\!] \in \mathbb{R}^k$ might tend to lie near a d-dimensional manifold where $d \ll k$. Then we could take $\mathbf{s}_t$ to be a vector in $\mathbb{R}^d$ that compactly encodes the approximate coordinates of $[\![\boldsymbol{\alpha}_t]\!]$ relative to the manifold: $\mathbf{s}_t = \nu([\![\boldsymbol{\alpha}_t]\!])$, where ν is the encoder.

In this new, nonlinearly warped coordinate system, the functions of $\mathbf{s}_{t-1}$ in (2)–(3) are no longer the simple, essentially linear functions given by (16) and (18). They become nonlinear functions operating on the manifold coordinates. (f_θ in (16) should now ensure that $\mathbf{s}_t^\top \approx \nu([\![(\nu^{-1}(\mathbf{s}_{t-1}))^\top P]\!])$, and g_θ in (18) should now estimate $\log\, (\nu^{-1}(\mathbf{s}_{t-1}))^\top P\mathbf{1}$.) In a sense, this is the reverse of the "kernel trick" (Boser et al., 1992) that converts a low-dimensional nonlinear function to a high-dimensional linear one.

Our hope is that $\mathbf{s}_t$ has enough dimensions $d \ll k$ to capture the useful information from the true $[\![\boldsymbol{\alpha}_t]\!]$, **and** that θ has enough dimensions $\ll k^2$ to capture most of the dynamics of equations (16) and (18). We thus proceed to fit the neural networks f_θ, g_θ directly to the data, *without ever knowing* the true k or the structure of the original operators $P \in \mathbb{R}^{k \times k}$.

We regard this as the implicit justification for various published probabilistic sequence models $p_\theta(\mathbf{y} \mid \mathbf{x})$ that incorporate neural networks. These models usually have the form of §1.1. Most simply, (f_θ, g_θ) can be instantiated as one time step in an RNN (Aharoni and Goldberg, 2017), but it is common to use enriched versions such as deep LSTMs. It is also common to have the state $\mathbf{s}_t$ contain not only a vector of manifold coordinates in $\mathbb{R}^d$ but also some unboundedly large representation of $(\mathbf{x}, \mathbf{y}_{:t})$ (cf. equation (4)), so the f_θ neural network can refer to this material with an attentional (Bahdanau et al., 2015) or stack mechanism (Dyer et al., 2015).

A few such papers have used *globally* normalized conditional models that can be viewed as approximating some OOHMM, e.g., the parsers of Dyer et al. (2016) and Andor et al. (2016). That is the case (§1.1) that particle smoothing aims to support. Most papers are *locally* normalized conditional models (e.g., Kann and Schütze, 2016; Aharoni and Goldberg, 2017); these simplify supervised training and can be viewed as approximating IOHMMs (footnote 5). For locally normalized models, $H_t = 0$ by construction, in which case particle filtering (which estimates $H_t = 0$) is just as good as particle smoothing. Particle filtering is still useful for these models, but lookahead's inability to help them is an expressive limitation (known as *label bias*) of locally normalized models. We hope the existence of particle smoothing (which learns an estimate H_t) will make it easier to adopt, train, and decode globally normalized models, as discussed in §1.3.

3.3 Neural approximation of logprob-to-go

We can adopt the same neuralization trick to approximate the OOHMM's logprob-to-go $H_t = C_t + \hat{H}_t$. We take $\bar{\mathbf{s}}_t \in \mathbb{R}^d$ on the same theory that it is a low-dimensional reparameterization of $[\![\beta_t]\!]$, and define $(\bar{f}_\phi, h_\phi)$ in equations (5)–(6) to be neural networks. Finally, we must replace the definition of C_t in (23) with another neural network c_ϕ that works on the low-dimensional approximations:[7]

$$C_t \stackrel{\text{def}}{=} c_\phi(\mathbf{s}_t, \bar{\mathbf{s}}_t) \quad \text{(except that } C_T \stackrel{\text{def}}{=} 0) \tag{25}$$

The resulting approximation to (24) (which does not actually require h_ϕ) will be denoted $q_{\theta,\phi}$:

$$q_{\theta,\phi}(y_t \mid \mathbf{x}, \mathbf{y}_{:t-1}) \stackrel{\text{def}}{\propto} \exp\, (g_\theta(\mathbf{s}_{t-1}, x_t, y_t) + C_t) \tag{26}$$

The neural networks in the present section are all parameterized by ϕ, and are intended to produce an estimate of the logprob-*to-go* H_t—a function of $\mathbf{x}_{t:}$, which sums over all possible $\mathbf{y}_{t:}$.

By contrast, the OOHMM-inspired neural networks suggested in §3.2 were used to specify an

[6] This is not true for the language of balanced parentheses.

[7] $C_T = 0$ is correct according to (23). Forcing this ensures $H_T = 0$, so our approximation becomes exact as of $t = T$.

actual model of the logprob-*so-far* G_t—a function of $\mathbf{x}_{:t}$ and $\mathbf{y}_{:t}$—using separate parameters θ.

Arguably ϕ has a harder modeling job than θ because it must implicitly sum over possible futures $\mathbf{y}_{t:}$. We now consider how to get corrected samples from $q_{\theta,\phi}$ even if ϕ gives poor estimates of H_t, and then how to train ϕ to improve those estimates.

4 Particle smoothing

In this paper, we assume nothing about the given model G_T except that it is given in the form of equations (1)–(3) (including the parameter vector θ).

Suppose we run the exact sampling strategy but approximate p_θ in (7) with a *proposal distribution* $q_{\theta,\phi}$ of the form in (25)–(26). Suppressing the subscripts on p and q for brevity, this means we are effectively drawing $\mathbf{y}$ not from $p(\mathbf{y} \mid \mathbf{x})$ but from

$$q(\mathbf{y} \mid \mathbf{x}) = \prod_{t=1}^{T} q(y_t \mid \mathbf{x}, \mathbf{y}_{:t-1}) \tag{27}$$

If $C_t \approx H_t + \text{const}$ within each y_t draw, then $q \approx p$.

Normalized importance sampling corrects (mostly) for the approximation by drawing *many* sequences $\mathbf{y}^{(1)}, \ldots \mathbf{y}^{(M)}$ IID from (27) and assigning $\mathbf{y}^{(m)}$ a relative *weight* of $w^{(m)} \stackrel{\text{def}}{=} \frac{p(\mathbf{y}^{(m)} \mid \mathbf{x})}{q(\mathbf{y}^{(m)} \mid \mathbf{x})}$. This *ensemble of weighted particles* yields a distribution

$$\hat{p}(\mathbf{y}) \stackrel{\text{def}}{=} \frac{\sum_{m=1}^{M} w^{(m)} \mathbb{I}(\mathbf{y}=\mathbf{y}^{(m)})}{\sum_{m=1}^{M} w^{(m)}} \approx p(\mathbf{y} \mid \mathbf{x}) \tag{28}$$

that can be used as discussed in §1.3. To compute $w^{(m)}$ in practice, we replace the numerator $p(\mathbf{y}^{(m)} \mid \mathbf{x})$ by the unnormalized version $\exp G_T$, which gives the same $\hat{p}$. Recall that each G_T is a sum $\sum_{t=1}^{T} g_\theta(\cdots)$.

Sequential importance sampling is an equivalent implementation that makes t the *outer* loop and m the *inner* loop. It computes a *prefix ensemble*

$$Y_t \stackrel{\text{def}}{=} \{(\mathbf{y}_{:t}^{(1)}, w_t^{(1)}), \ldots, (\mathbf{y}_{:t}^{(M)}, w_t^{(M)})\} \tag{29}$$

for each $0 \leq t \leq T$ in sequence. Initially, $(\mathbf{y}_{:0}^{(m)}, w_0^{(m)}) = (\epsilon, \exp C_0)$ for all m. Then for $0 < t \leq T$, we extend these particles in parallel:

$$\mathbf{y}_{:t}^{(m)} = \mathbf{y}_{:t-1}^{(m)} y_t^{(m)} \quad \text{(concatenation)} \tag{30}$$

$$w_t^{(m)} = w_{t-1}^{(m)} \frac{\exp\left(g_\theta(\mathbf{s}_{t-1}, x_t, y_t) + C_t - C_{t-1}\right)}{q(y_t \mid \mathbf{x}, \mathbf{y}_{:t-1})} \tag{31}$$

where each $y_t^{(m)}$ is drawn from (26). Each Y_t yields a distribution $\hat{p}_t$ over prefixes $\mathbf{y}_{:t}$, which estimates the distribution $p_t(\mathbf{y}_{:t}) \stackrel{\text{def}}{\propto} \exp\left(G_t + C_t\right)$. We return $\hat{p} \stackrel{\text{def}}{=} \hat{p}_T \approx p_T = p$. This gives the same $\hat{p}$ as in (28): the final $\mathbf{y}_T^{(m)}$ are the same, with the same final weights $w_T^{(m)} = \frac{\exp G_T}{q(\mathbf{y}^{(m)} \mid \mathbf{x})}$, where G_T was now summed up as $C_0 + \sum_{t=1}^{T} g_\theta(\cdots) + C_t - C_{t-1}$.

That is our basic *particle smoothing* strategy. If we use the naive approximation $C_t = 0$ everywhere, it reduces to *particle filtering*. In either case, various well-studied improvements become available, such as various resampling schemes (Douc and Cappé, 2005) and the particle cascade (Paige et al., 2014).[8]

An easy improvement is *multinomial resampling*. After computing each $\hat{p}_t$, this replaces Y_t with a set of M new draws from $\hat{p}_t$ ($\approx p_t$), each of weight 1—which tends to drop low-weight particles and duplicate high-weight ones.[9] For this to usefully focus the ensemble on good prefixes $\mathbf{y}_{:t}$, p_t should be a good approximation to the true marginal $p(\mathbf{y}_{:t} \mid \mathbf{x}) \propto \exp\left(G_t + H_t\right)$ from (10). That is why we arranged for $p_t(\mathbf{y}_{:t}) \propto \exp\left(G_t + C_t\right)$. Without C_t, we would have only $p_t(\mathbf{y}_{:t}) \propto \exp G_t$—which is fine for the traditional particle filtering setting, but in our setting it ignores future information in $\mathbf{x}_{t:}$ (which we have assumed is available) and also favors sequences $\mathbf{y}$ that happen to accumulate most of their global score G_T early rather than late (which is possible when the globally normalized model (1)–(2) is *not* factored in the generative form (4)).

5 Training the Sampler Heuristic

We now consider training the parameters ϕ of our sampler. These parameters determine the updates $\bar{f}_\phi$ in (6) and the compatibility function c_ϕ in (25). As a result, they determine the proposal distribution q used in equations (27) and (31), and thus determine the stochastic choice of $\hat{p}$ that is returned by the sampler on a given input $\mathbf{x}$.

In this paper, we simply try to tune ϕ to yield good proposals. Specifically, we try to ensure that $q_\phi(\mathbf{y} \mid \mathbf{x})$ in equation (27) is close to $p(\mathbf{y} \mid \mathbf{x})$ from equation (1). While this may not be necessary for the sampler to perform well downstream,[10] it does

[8] The particle cascade would benefit from an estimate of $\hat{H}_t$, as it (like A* search) compares particles of different lengths.

[9] While resampling mitigates the degeneracy problem, it could also reduce the diversity of particles. In our experiments in this paper, we only do multinomial resampling when the effective sample size of $\hat{p}_t$ is lower than $\frac{M}{2}$. Doucet and Johansen (2009) give a more thorough discussion on when to resample.

[10] In principle, one could attempt to train ϕ "end-to-end" on some downstream objective by using reinforcement learning or the Gumbel-softmax trick (Jang et al., 2017; Maddison et al., 2017). For example, we might try to ensure that $\hat{p}$ closely matches the model's distribution p (equation (28))—the "na-

guarantee it (assuming that the model p is correct). Specifically, we seek to minimize

$$(1-\lambda)\mathrm{KL}(p||q_\phi) + \lambda\mathrm{KL}(q_\phi||p) \quad (\text{with } \lambda \in [0,1]) \tag{32}$$

averaged over examples $\mathbf{x}$ drawn from a training set.[11] (The training set need not provide true $\mathbf{y}$'s.)

The *inclusive KL divergence* $\mathrm{KL}(p||q_\phi)$ is an expectation under p. We estimate it by replacing p with a sample $\hat{p}$, which in practice we can obtain with our sampler under the current ϕ. (The danger, then, is that $\hat{p}$ will be biased when ϕ is not yet well-trained; this can be mitigated by increasing the sample size M when drawing $\hat{p}$ for training purposes.)

Intuitively, this term tries to encourage q_ϕ in future to re-propose those $\mathbf{y}$ values that turned out to be "good" and survived into $\hat{p}$ with high weights.

The *exclusive KL divergence* $\mathrm{KL}(q_\phi||p)$ is an expectation under q_ϕ. Since we can sample from q_ϕ exactly, we can get an unbiased estimate of $\nabla_\phi \mathrm{KL}(q_\phi||p)$ with the likelihood ratio trick (Glynn, 1990).[12] (The danger is that such "REINFORCE" methods tend to suffer from very high variance.)

This term is a popular objective for variational approximation. Here, it tries to discourage q_ϕ from re-proposing "bad" $\mathbf{y}$ values that turned out to have low $\exp G_T$ relative to their proposal probability.

Our experiments balance "recall" (inclusive) and "precision" (exclusive) by taking $\lambda = \frac{1}{2}$ (which Appendix F compares to $\lambda \in \{0, 1\}$). Alas, because of our approximation to the inclusive term, neither term's gradient will "find" and directly encourage good $\mathbf{y}$ values that have never been proposed. Appendix B gives further discussion and formulas.

6 Models for the Experiments

To evaluate our methods, we needed pre-trained models p_θ. We experimented on several models. In each case, we trained a *generative* model $p_\theta(\mathbf{x}, \mathbf{y})$, so that we could try sampling from its posterior distribution $p_\theta(\mathbf{y} \mid \mathbf{x})$. This is a very common setting where particle smoothing should be able to help. Details for replication are given in Appendix C.

tural" goal of sampling. This objective can tolerate inaccurate local proposal distributions in cases where the algorithm could recover from them through resampling. Looking even farther downstream, we might merely want $\hat{p}$—which is typically used to compute expectations—to provide accurate guidance to some decision or training process (see Appendix E). This might not require fully matching the model, and might even make it desirable to deviate from an inaccurate model.

[11] Training a single approximation q_ϕ for all $\mathbf{x}$ is known as *amortized inference*.

[12] The normalizing constant of p from (1) can be ignored because the gradient of a constant is 0.

6.1 Tagging models

We can regard a tagged sentence $(\mathbf{x}, \mathbf{y})$ as a string over the "pair alphabet" $\mathcal{X} \times \mathcal{Y}$. We train an RNN language model over this "pair alphabet"—this is a neuralized OOHMM as suggested in §3.2:

$$\log p_\theta(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{T} \log p_\theta(x_t, y_t \mid \mathbf{s}_{t-1}) \tag{33}$$

This model is locally normalized, so that $\log p_\theta(\mathbf{x}, \mathbf{y})$ (as well as its gradient) is straightforward to compute for a given training pair $(\mathbf{x}, \mathbf{y})$. Joint sampling from it would also be easy (§3.2).

However, $p(\mathbf{y} \mid \mathbf{x})$ is globally renormalized (by an unknown partition function that depends on $\mathbf{x}$, namely $\exp H_0$). Conditional sampling of $\mathbf{y}$ is therefore potentially hard. Choosing y_t optimally requires knowledge of H_t, which depends on the future $\mathbf{x}_{t:}$.

As we noted in §1, many NLP tasks can be seen as tagging problems. In this paper we experiment with two such tasks: **English stressed syllable tagging**, where the stress of a syllable often depends on the number of remaining syllables,[13] providing good reason to use the *lookahead* provided by particle smoothing; and **Chinese NER**, which is a familiar textbook application and reminds the reader that our formal setup (tagging) provides enough machinery to treat other tasks (chunking).

English stressed syllable tagging This task tags a sequence of phonemes $\mathbf{x}$, which form a word, with their stress markings $\mathbf{y}$. Our training examples are the stressed words in the CMU pronunciation dictionary (Weide, 1998). We test the sampler on held-out unstressed words.

Chinese social media NER This task does named entity recognition in Chinese, by tagging the characters of a Chinese sentence in a way that marks the named entities. We use the dataset from Peng and Dredze (2015), whose tagging scheme is a variant of the BIO scheme mentioned in §1. We test the sampler on held-out sentences.

6.2 String source separation

This is an artificial task that provides a discrete analogue of speech source separation (Zibulevsky and Pearlmutter, 2001). The generative model is that J strings (possibly of different lengths) are generated

[13] English, like many other languages, assigns stress from right to left (Hayes, 1995).

IID from an RNN language model, and are then combined into a single string $\mathbf{x}$ according to a random *interleaving* string $\mathbf{y}$.[14] The posterior $p(\mathbf{y} \mid \mathbf{x})$ predicts the interleaving string, which suffices to reconstruct the original strings. The interleaving string is selected from the uniform distribution over all possible interleavings (given the J strings' lengths). For example, with $J = 2$, a possible generative story is that we first sample two strings Foo and Bar from an RNN language model. We then draw an interleaving string 112122 from the aforementioned uniform distribution, and interleave the J strings deterministically to get FoBoar.

$p(\mathbf{x}, \mathbf{y})$ is proportional to the product of the probabilities of the J strings. The only parameters of p_θ, then, are the parameters of the RNN language model, which we train on clean (non-interleaved) samples from a corpus. We test the sampler on random interleavings of held-out samples.

The state $\mathbf{s}$ (which is provided as an input to c_θ in (25)) is the concatenation of the J states of the language model as it independently generates the J strings, and $g_\theta(\mathbf{s}_{t-1}, x_t, y_t)$ is the log-probability of generating x_t as the next character of the $y_t{}^{\text{th}}$ string, given that string's language model state within $\mathbf{s}_{t-1}$. As a special case, $\mathbf{x}_T = \text{EOS}$ (see footnote 1), and $g_\theta(\mathbf{s}_{T-1}, \text{EOS}, \text{EOS})$ is the total log-probability of termination in all J language model states.

String source separation has good reason for lookahead: appending character "o" to a reconstructed string "␣gh" is only advisable if "s" and "t" are coming up soon to make "ghost." It also illustrates a powerful application setting—posterior inference under a generative model. This task conveniently allowed us to construct the generative model from a pre-trained language model. Our constructed generative model illustrates that the state $\mathbf{s}$ and transition function f can reflect interesting problem-specific structure.

CMU Pronunciation dictionary The CMU pronunciation dictionary (already used above) provides sequences of phonemes. Here we use words no longer than 5 phonemes. We interleave the (unstressed) phonemes of $J = 5$ words.

Penn Treebank The PTB corpus (Marcus et al., 1993) provides English sentences, from which we use only the sentences of length ≤ 8. We interleave the words of $J = 2$ sentences.

[14] We formally describe the generative process in Appendix G.

7 Experiments

In our experiments, we are given a pre-trained scoring model p_θ, and we train the parameters ϕ of a particle smoothing algorithm.[15]

We now show that our proposed neural particle smoothing sampler does better than the particle filtering sampler. To define "better," we evaluate samplers on the *offset KL divergence* from the true posterior.

7.1 Evaluation metrics

Given $\mathbf{x}$, the "natural" goal of conditional sampling is for the sample distribution $\hat{p}(\mathbf{y})$ to approximate the true distribution $p_\theta(\mathbf{y} \mid \mathbf{x}) = \exp G_T / \exp H_0$ from (1). We will therefore report—averaged over all held-out test examples $\mathbf{x}$—the KL divergence

$$\begin{aligned} \mathrm{KL}(\hat{p}||p) = \mathbb{E}_{\mathbf{y}\sim\hat{p}}\left[\log \hat{p}(\mathbf{y})\right] \qquad (34) \\ - \left(\mathbb{E}_{\mathbf{y}\sim\hat{p}}\left[\log \tilde{p}(\mathbf{y} \mid \mathbf{x})\right] - \log Z(\mathbf{x})\right), \end{aligned}$$

where $\tilde{p}(\mathbf{y} \mid \mathbf{x})$ denotes the *unnormalized* distribution given by $\exp G_T$ in (2), and $Z(\mathbf{x})$ denotes its normalizing constant, $\exp H_0 = \sum_{\mathbf{y}} \tilde{p}(\mathbf{y} \mid \mathbf{x})$.

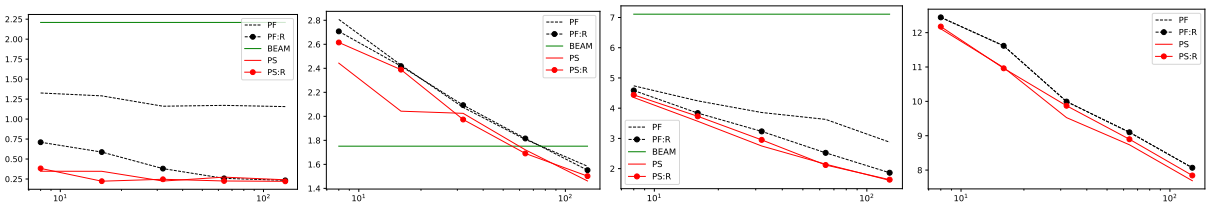
As we are unable to compute $\log Z(\mathbf{x})$ in practice, we replace it with an estimate $z(\mathbf{x})$ to obtain an *offset KL divergence*. This change of constant does not change the measured difference between two samplers, $\mathrm{KL}(\hat{p}_1||p) - \mathrm{KL}(\hat{p}_2||p)$. Nonetheless, we try to use a reasonable estimate so that the reported KL divergence is interpretable in an absolute sense. Specifically, we take $z(\mathbf{x}) = \log \sum_{\mathbf{y}\in\mathcal{Y}} \tilde{p}(\mathbf{y} \mid \mathbf{x}) \leq \log Z$, where $\mathcal{Y}$ is the full set of distinct particles $\mathbf{y}$ that we ever drew for input $\mathbf{x}$, including samples from the beam search models, while constructing the experimental results graph.[16] Thus, the offset KL divergence is a "best effort" lower bound on the true exclusive KL divergence $\mathrm{KL}(\hat{p}||p)$.

7.2 Results

In all experiments we compute the offset KL divergence for both the particle filtering samplers and the particle smoothing samplers, for varying ensemble sizes M. We also compare against a beam search baseline that keeps the highest-scoring M particles at each step (scored by $\exp G_t$ with no lookahead). The results are in Figures 2a–2d.

[15] For the details of the training procedures and the specific neural architectures in our models, see Appendices C and D.

[16] Thus, $\mathcal{Y}$ was collected across all samplings, iterations,and ensemble sizes M, in an attempt to make the summation over $\mathcal{Y}$ as complete as possible. For good measure, we added some extra particles: whenever we drew M particles via particle smoothing, we drew an additional $2M$ particles by particle filtering and added them to $\mathcal{Y}$.



(a) tagging: stressed syllables (b) tagging: Chinese NER (c) source separation: PTB (d) source separation: CMUdict

Figure 2: Offset KL divergences for the tasks in §§ 6.1 and 6.2. The logarithmic x-axis is the size of particles M ($8 \leq M \leq 128$). The y-axis is the offset KL divergence described in §7.1 (in bits per sequence). The smoothing samplers offer considerable speedup: for example, in Figure 2a, the non-resampled smoothing sampler achieves comparable offset KL divergences with only $1/4$ as many particles as its filtering counterparts. Abbreviations in the legend: PF=particle filtering. PS=particle smoothing. BEAM=beam search. ':R' suffixes indicate resampled variants. For readability, beam search results are omitted from Figure 2d, but appear in Figure 3 of the appendices.

Given a fixed ensemble size, we see the smoothing sampler consistently performs better than the filtering counterpart. It often achieves comparable performance at a fraction of the ensemble size.

Beam search on the other hand falls behind on three tasks: stress prediction and the two source separation tasks. It does perform better than the stochastic methods on the Chinese NER task, but only at small beam sizes. Varying the beam size barely affects performance at all, across all tasks. This suggests that beam search is unable to explore the hypothesis space well.

We experiment with resampling for both the particle filtering sampler and our smoothing sampler. In source separation and stressed syllable prediction, where the right context contains critical information about how viable a particle is, resampling helps particle filtering *almost* catch up to particle smoothing. Particle smoothing itself is not further improved by resampling, presumably because its effective sample size is high. The goal of resampling is to kill off low-weight particles (which were overproposed) and reallocate their resources to higher-weight ones. But with particle smoothing, there are fewer low-weight particles, so the benefit of resampling may be outweighted by its cost (namely, increased variance).

8 Related Work

Much previous work has employed sequential importance sampling for approximate inference of intractable distributions (e.g., Thrun, 2000; Andrews et al., 2017). Some of this work learns adaptive proposal distributions in this setting (e.g. Gu et al., 2015; Paige and Wood, 2016). The key difference in our work is that we consider future inputs, which is impossible in online decision settings such as robotics. Klaas et al. (2006) did do particle smoothing, like us, but they did not learn adaptive proposal distributions.

Just as we use a right-to-left RNN to guide *posterior sampling* of a left-to-right generative model, Krishnan et al. (2017) employed a right-to-left RNN to guide *posterior marginal inference* in the same sort of model. Serdyuk et al. (2018) used a right-to-left RNN to regularize training of such a model.

9 Conclusion

We have described neural particle smoothing, a sequential Monte Carlo method for approximate sampling from the posterior of incremental neural scoring models. Sequential importance sampling has arguably been underused in the natural language processing community. It is quite a plausible strategy for dealing with rich, globally normalized probability models such as neural models—particularly if a good sequential proposal distribution can be found. Our contribution is a neural proposal distribution, which goes beyond particle filtering in that it uses a right-to-left recurrent neural network to "look ahead" to future symbols of $\mathbf{x}$ when proposing each symbol y_t. The form of our distribution is well-motivated.

There are many possible extensions to the work in this paper. For example, we can learn the generative model and proposal distribution jointly; we can also infuse them with hand-crafted structure, or use more deeply stacked architectures; and we can try training the proposal distribution end-to-end (footnote 10). Another possible extension would be to allow each step of q to propose a *sequence* of actions, effectively making the tagset size ∞. This extension relaxes our $|\mathbf{y}| = |\mathbf{x}|$ restriction from §1 and would allow us to do general sequence-to-sequence transduction.

Acknowledgements

This work has been generously supported by a Google Faculty Research Award and by Grant No. 1718846 from the National Science Foundation.

References

Roee Aharoni and Yoav Goldberg. 2017. Morphological inflection generation with hard monotonic attention. In *ACL*.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *ACL*.

Nicholas Andrews, Mark Dredze, Benjamin Van Durme, and Jason Eisner. 2017. Bayesian modeling of lexical resources for low-resource settings. In *ACL*.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. In *ICLR*.

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

Yoshua Bengio and Paolo Frasconi. 1996. Input-output HMMs for sequence processing. *IEEE Transactions on Neural Networks*, 7(5):1231–1249.

Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. 1992. A training algorithm for optimal margin classifiers. In *COLT*.

Alexandre Bouchard-Côté, Percy Liang, Thomas Griffiths, and Dan Klein. 2007. A probabilistic approach to diachronic phonology. In *EMNLP-CoNLL*, pages 887–896.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*.

Ryan Cotterell, John Sylak-Glassman, and Christo Kirov. 2017. Neural graphical models over strings for principal parts morphological paradigm completion. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 759–765.

Randal Douc and Olivier Cappé. 2005. Comparison of resampling schemes for particle filtering. In *Image and Signal Processing and Analysis, 2005. ISPA 2005. Proceedings of the 4th International Symposium on*, pages 64–69. IEEE.

Arnaud Doucet and Adam M. Johansen. 2009. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, 12(656-704):3.

Markus Dreyer and Jason Eisner. 2009. Graphical models over multiple strings. In *EMNLP*.

Markus Dreyer, Jason R. Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *EMNLP*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *ACL*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *HLT-NAACL*.

Jenny Rose Finkel, Christopher D. Manning, and Andrew Y. Ng. 2006. Solving the problem of cascading errors: Approximate Bayesian inference for linguistic annotation pipelines. In *EMNLP*.

Peter W. Glynn. 1990. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33(10):75–84.

Shixiang Gu, Zoubin Ghahramani, and Richard E. Turner. 2015. Neural adaptive sequential Monte Carlo. In *NIPS*.

Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. 1968. A formal basis for the heuristic determination of minimal cost paths. 4(2):100–107.

Bruce Hayes. 1995. *Metrical Stress Theory: Principles and Case Studies*. University of Chicago Press.

Alexander T. Ihler and David A. McAllester. 2009. Particle belief propagation. In *AISTATS*.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with Gumbel-softmax. In *ICLR*.

Katharina Kann and Hinrich Schütze. 2016. Single-model encoder-decoder with explicit morphological representation for reinflection. In *ACL*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Mike Klaas, Mark Briers, Nando de Freitas, Arnaud Doucet, Simon Maskell, and Dustin Lang. 2006. Fast particle smoothing: If I had a million particles. In *ICML*.

Bjarne Knudsen and Michael M. Miyamoto. 2003. Sequence alignments and pair hidden Markov models using evolutionary history. *Journal of Molecular Biology*, 333(2):453 – 460.

Rahul G. Krishnan, Uri Shalit, and David Sontag. 2017. Structured inference networks for nonlinear state space models. In *AAAI*.

John Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Thibaut Lienart, Yee Whye Teh, and Arnaud Doucet. 2015. Expectation particle belief propagation. In *NIPS*.

Roderick J. A. Little and Donald B. Rubin. 1987. *Statistical Analysis with Missing Data*. J. Wiley & Sons, New York.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *ICLR*.

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

Andrew McCallum, Dayne Freitag, and Fernando Pereira. 2000. Maximum entropy Markov models for information extraction and segmentation. In *Machine Learning: Proceedings of the 17th International Conference (ICML 2000)*, pages 591–598, Stanford, CA.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3.

Brooks Paige and Frank D. Wood. 2016. Inference networks for sequential Monte Carlo in graphical models. In *ICML*.

Brooks Paige, Frank D. Wood, Arnaud Doucet, and Yee Whye Teh. 2014. Asynchronous anytime sequential Monte Carlo. In *NIPS*.

Nanyun Peng and Mark Dredze. 2015. Named entity recognition for Chinese social media with jointly trained embeddings. In *EMNLP*.

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

Fernando C. N. Pereira and Michael D. Riley. 1997. Speech recognition by composition of weighted finite automata. *Finite-State Language Processing*, page 431.

Lawrence R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of IEEE*, 77(2):257–285.

Lance A. Ramshaw and Mitchell P. Marcus. 1999. Text chunking using transformation-based learning. In *Natural Language Processing Using Very Large Corpora*, pages 157–176. Springer.

Branko Ristic, Sanjeev Arulampalam, and Neil James Gordon. 2004. *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Artech House.

Herbert Robbins and Sutton Monro. 1951. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407.

Dmitriy Serdyuk, Nan Rosemary Ke, Alessandro Sordoni, Adam Trischler, Chris Pal, and Yoshua Bengio. 2018. Twin networks: Matching the future for sequence generation. In *ICLR*.

Andreas Stuhlmüller, Jacob Taylor, and Noah Goodman. 2013. Learning stochastic inverses. In *NIPS*.

Sebastian Thrun. 2000. Monte Carlo POMDPs. In *NIPS*.

Andrew J. Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, IT-13(2):260–269.

Greg C. G. Wei and Martin A. Tanner. 1990. A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms. *Journal of the American Statistical Association*, 85(411):699–704.

Robert L. Weide. 1998. The CMU pronunciation dictionary, release 0.6.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(23).

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *EMNLP*.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*, volume 3, pages 195–206.

Michael Zibulevsky and Barak A. Pearlmutter. 2001. Blind source separation by sparse decomposition in a signal dictionary. *Neural Computation*, 13(4):863–882.

A The logprob-to-go for HMMs

As noted in §2.1, the logprob-to-go H_t can be computed by the backward algorithm. By the definition of H_t in equation (10),

$$\exp H_t = \sum_{\mathbf{y}_{t:}} \exp\left(G_T - G_t\right) \tag{35}$$

$$= \sum_{\mathbf{y}_{t:}} \exp \sum_{j=t+1}^{T} g_\theta(\mathbf{s}_{j-1}, x_j, y_j) \tag{36}$$

$$= \sum_{\mathbf{y}_{t:}} \prod_{j=t+1}^{T} p_\theta(y_j \mid y_{j-1}) \cdot p_\theta(x_j \mid y_j)$$

$$= (\boldsymbol{\beta}_t)_{y_t} \text{ (backward prob of } y_t \text{ at time } t)$$

where the vector $\boldsymbol{\beta}_t$ is defined by base case $(\boldsymbol{\beta}_T)_y = 1$ and for $0 \leq t < T$ by the recurrence

$$(\beta_t)_y \stackrel{\text{def}}{=} \sum_{\mathbf{y}_{t:}} p_\theta(\mathbf{x}_{t:}, \mathbf{y}_{t:} \mid y_t = y) \tag{37}$$

$$= \sum_{y'} p_\theta(y' \mid y) \cdot p_\theta(x_{t+1} \mid y') \cdot (\boldsymbol{\beta}_{t+1})_{y'}$$

The backward algorithm (20) for OOHMMs in §2.2 is a variant of this.

B Gradients for Training the Proposal Distribution

For a given $\mathbf{x}$, both forms of KL divergence achieve their minimum of 0 when $(\forall \mathbf{y})\ q_\phi(\mathbf{y} \mid \mathbf{x}) = p(\mathbf{y} \mid \mathbf{x})$. However, we are unlikely to be able to find such a ϕ; the two metrics penalize q_ϕ differently for mismatches. We simplify the notation below by writing $q_\phi(\mathbf{y})$ and $p(\mathbf{y})$, suppressing the conditioning on $\mathbf{x}$.

Inclusive KL Divergence The inclusive KL divergence has that name because it is finite only when $\text{support}(q_\phi) \supseteq \text{support}(p)$, i.e., when q_ϕ is capable of proposing any string $\mathbf{y}$ that has positive probability under p. This is required for q_ϕ to be a valid proposal distribution for importance sampling.

$$\begin{aligned} &\text{KL}\left(p||q_\phi\right) \\ &\quad = \mathbb{E}_{\mathbf{y}\sim p}\left[\log p\left(\mathbf{y}\right) - \log q_\phi(\mathbf{y})\right] \\ &\quad = \mathbb{E}_{\mathbf{y}\sim p}\left[\log p\left(\mathbf{y}\right)\right] \\ &\qquad - \mathbb{E}_{\mathbf{y}\sim p}\left[\log q_\phi\left(\mathbf{y}\right)\right] \end{aligned} \tag{38}$$

The first term $\mathbb{E}_{\mathbf{y}\sim p}\left[\log p\left(\mathbf{y}\right)\right]$ is a constant with regard to ϕ. As a result, the gradient of the above is just the gradient of the second term:

$$\nabla_\phi \text{KL}(p||q_\phi) = \nabla_\phi \underbrace{\mathbb{E}_{\mathbf{y}\sim p}\left[-\log q_\phi\left(\mathbf{y}\right)\right]}_{\text{the cross-entropy } H(p,q_\phi)}$$

We cannot directly sample from p. However, our weighted mixture $\hat{p}$ from equation (28) (obtained by sequential importance sampling) could be a good approximation:

$$\begin{aligned} \nabla_\phi \text{KL}(p||q_\phi) &\approx \nabla_\phi \mathbb{E}_{\mathbf{y}\sim\hat{p}}\left[-\log q_\phi\left(\mathbf{y}\right)\right] \\ &= \sum_{t=1}^{T} \mathbb{E}_{\hat{p}}\left[-\nabla_\phi \log q_\phi(y_t \mid y_{:t-1}, \mathbf{x})\right] \end{aligned} \tag{39}$$

Following this approximate gradient downhill has an intuitive interpretation: if a particular y_t value ends up with high relative weight in the final ensemble $\hat{p}$, then we will try to adjust q_ϕ so that it would have had a high probability of proposing that y_t value at step t in the first place.

Exclusive KL Divergence The exclusive divergence has that name because it is finite only when $\text{support}(q_\phi) \subseteq \text{support}(p)$. It is defined by

$$\begin{aligned} \text{KL}(q_\phi||p) &= \mathbb{E}_{\mathbf{y}\sim q_\phi}\left[\log q_\phi(\mathbf{y}) - \log p(\mathbf{y})\right] \\ &= \mathbb{E}_{\mathbf{y}\sim q_\phi}\left[\log q_\phi(\mathbf{y}) - \log \tilde{p}(\mathbf{y})\right] + \log Z \\ &= \sum_{\mathbf{y}} q_\phi(\mathbf{y}) \underbrace{\left[\log q_\phi(\mathbf{y}) - \log \tilde{p}(\mathbf{y})\right]}_{\text{call this } d_\phi(\mathbf{y})} + \log Z \end{aligned} \tag{40}$$

where $p(\mathbf{y}) = \frac{1}{Z}\tilde{p}(\mathbf{y})$ for $\tilde{p}(\mathbf{y}) = \exp G_T$ and $Z = \sum_{\mathbf{y}} \tilde{p}(\mathbf{y})$. With some rearrangement, we can write its gradient as an expectation that can be estimated by sampling from q_ϕ.[17] Observing that Z is constant with respect to ϕ, first write

$$\nabla_\phi \text{KL}(q_\phi||p) \tag{41}$$

$$= \sum_{\mathbf{y}} \nabla_\phi \left(q_\phi(\mathbf{y})\, d_\phi(\mathbf{y})\right) \tag{42}$$

$$\begin{aligned} &= \sum_{\mathbf{y}} \left(\nabla_\phi q_\phi(\mathbf{y})\right)\, d_\phi(\mathbf{y}) \\ &\qquad + \sum_{\mathbf{y}} \underbrace{q_\phi(\mathbf{y}) \nabla_\phi \log q_\phi(\mathbf{y})}_{=\nabla_\phi q_\phi(\mathbf{y})} \\ &= \sum_{\mathbf{y}} \left(\nabla_\phi q_\phi(\mathbf{y})\right)\, d_\phi(\mathbf{y}) \end{aligned}$$

where the last step uses the fact that $\sum_{\mathbf{y}} \nabla_\phi q_\phi(\mathbf{y}) = \nabla_\phi \sum_{\mathbf{y}} q_\phi(\mathbf{y}) = \nabla_\phi 1 = 0$. We can turn this into an expectation with a second use of Glynn (1990)'s observation that

[17] This is an extension of the REINFORCE trick (Williams, 1992), which estimates the gradient of $\mathbb{E}_{\mathbf{y}\sim q_\phi}\left[\text{reward}(\mathbf{y})\right]$ when the reward is independent of ϕ. In our case, the expectation is over a quantity that does depend on ϕ.

$\nabla_\phi q_\phi(\mathbf{y}) = q_\phi(\mathbf{y}) \nabla_\phi \log q_\phi(\mathbf{y})$ (the "likelihood ratio trick"):

$$\begin{aligned}
&\nabla_\phi \mathrm{KL}(q_\phi || p) \\
&\quad = \sum_{\mathbf{y}} q_\phi(\mathbf{y}) d_\phi(\mathbf{y}) \nabla_\phi \log q_\phi(\mathbf{y}) \\
&\quad = \mathbb{E}_{\mathbf{y} \sim q_\phi} \left[d_\phi(\mathbf{y}) \nabla_\phi \log q_\phi(\mathbf{y}) \right] \qquad (43)
\end{aligned}$$

which can, if desired, be further rewritten as

$$\begin{aligned}
&= \mathbb{E}_{\mathbf{y} \sim q_\phi} \left[d_\phi(\mathbf{y}) \nabla_\phi d_\phi(\mathbf{y}) \right] \\
&= \mathbb{E}_{\mathbf{y} \sim q_\phi} \left[\nabla_\phi \left(\tfrac{1}{2} d_\phi(\mathbf{y})^2 \right) \right] \qquad (44)
\end{aligned}$$

If we regard $d_\phi(\mathbf{y})$ as a signed error (in the log domain) in trying to fit q_ϕ to $\tilde{p}$, then the above gradient of KL can be interpreted as the gradient of the mean squared error (divided by 2).[18]

We would get the same gradient for any rescaled version of the unnormalized distribution $\tilde{p}$, but the formula for obtaining that gradient would be different. In particular, if we rewrite the above derivation but add a constant b to both $\log \tilde{p}(\mathbf{y})$ and $\log Z$ throughout (equivalent to adding b to G_T), we will get the slightly generalized expectation formulas

$$\mathbb{E}_{\mathbf{y} \sim q_\phi} \left[(d_\phi(\mathbf{y}) - b) \nabla_\phi \log q_\phi(\mathbf{y}) \right] \qquad (45)$$

$$\mathbb{E}_{\mathbf{y} \sim q_\phi} \left[\nabla_\phi \left(\tfrac{1}{2} \left(d_\phi(\mathbf{y}) - b \right)^2 \right) \right] \qquad (46)$$

in place of equations (43) and (44). By choosing an appropriate "baseline" b, we can reduce the variance of the sampling-based estimate of these expectations. This is similar to the use of a baseline in the REINFORCE algorithm (Williams, 1992). In this work we choose b using an exponential moving average of past $\mathbb{E}\left[d_\phi(\mathbf{y})\right]$ values: at the end of each training minibatch, we update $b \leftarrow 0.1 \cdot b + 0.9 \cdot \bar{d}$, where $\bar{d}$ is the mean of the estimated $\mathbb{E}_{\mathbf{y} \sim q_\phi(\cdot|\mathbf{x})}\left[d_\phi(\mathbf{y})\right]$ values for all examples $\mathbf{x}$ in the minibatch.

C Implementation Details

We implement all RNNs in this paper as GRU networks (Cho et al., 2014) with $d = 32$ hidden units (state space $\mathbb{R}^{32}$). Each of our models (§6) always specifies the logprob-so-far in equations (2) and (3) using a 1-layer left-to-right GRU,[19] while the corresponding proposal distribution (§3.3) always specifies the state $\bar{\mathbf{s}}_t$ in (6) using a 2-layer right-to-left GRU, and specifies the compatibility function C_t in (23) using a 4-layer feedforward ReLU network.[20] For the Chinese social media NER task (§6.1), we use the Chinese character embeddings provided by Peng and Dredze (2015), while for the source separation tasks (§6.2), we use the 50-dimensional GloVe word embeddings (Pennington et al., 2014). In other cases, we train embeddings along with the rest of the network. We optimize with the Adam optimizer using the default parameters (Kingma and Ba, 2015) and L_2 regularization coefficient of 10^{-5}.

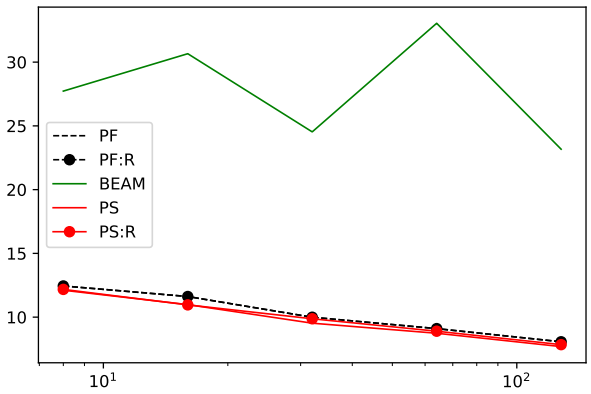


Figure 3: Offset KL divergence for the source separation task on phoneme sequences.

D Training Procedures

In all our experiments, we train the incremental scoring models (the tagging and source separation models described in §6.1 and §6.2, respectively) on the training dataset T. We do early stopping, using perplexity on a held-out development set D_1 to choose the number of epochs to train (maximum of 3).

Having obtained these model parameters θ, we train our proposal distributions $q_{\theta,\phi}$ on T, keeping θ fixed and only tuning ϕ. Again we use early stopping, using the KL divergence from §7.1 on a separate development set D_2 to choose the number of epochs to train (maximum of 20 for the two tagging tasks and source separation on the PTB dataset, and maximum of 50 for source separation on the phoneme sequence dataset). We then evaluate q_{θ^*,ϕ^*} on the test dataset E.

[Appendices E–G appear in the supplementary material file.]

[18] We thank Hongyuan Mei, Tim Vieira, and Sanjeev Khudanpur for insightful discussions on this derivation.

[19] For the tagging task described in §6.1, $g_\theta(\mathbf{s}_{t-1}, x_t, y_t) \stackrel{\text{def}}{=} \log p_\theta(x_t, y_t \mid \mathbf{s}_{t-1})$, where the GRU state $\mathbf{s}_{t-1}$ is used to define a softmax distribution over possible (x_t, y_t) pairs in the same manner as an RNN language model (Mikolov et al., 2010). Likewise, for the source separation task (§6.2), the source language models described in Appendix G are GRU-based RNN language models.

[20] As input to C_t, we actually provide not only $\mathbf{s}_t, \bar{\mathbf{s}}_t$ but also the states $f_\theta(\mathbf{s}_{t-1}, x_t, y)$ (including $\mathbf{s}_t$) that could have been reached for *each* possible value y of y_t. We have to compute these anyway while constructing the proposal distribution, and we find that it helps performance to include them.

Neural Syntactic Generative Models with Exact Marginalization

Jan Buys[1,2] **and Phil Blunsom**[1,3]
[1]Department of Computer Science, University of Oxford
[2]Paul G. Allen School of Computer Science & Engineering, University of Washington
[3]DeepMind
jbuys@cs.washington.edu, phil.blunsom@cs.ox.ac.uk

Abstract

We present neural syntactic generative models with exact marginalization that support both dependency parsing and language modeling. Exact marginalization is made tractable through dynamic programming over shift-reduce parsing and minimal RNN-based feature sets. Our algorithms complement previous approaches by supporting batched training and enabling online computation of next word probabilities. For supervised dependency parsing, our model achieves a state-of-the-art result among generative approaches. We also report empirical results on unsupervised syntactic models and their role in language modeling. We find that our model formulation of latent dependencies with exact marginalization do not lead to better intrinsic language modeling performance than vanilla RNNs, and that parsing accuracy is not correlated with language modeling perplexity in stack-based models.

1 Introduction

We investigate the feasibility of neural syntactic generative models with structured latent variables in which exact inference is tractable. Recent models have added structure to recurrent neural networks at the cost of giving up exact inference, or through using soft structure instead of latent variables (Dyer et al., 2016; Yogatama et al., 2016; Grefenstette et al., 2015). We propose generative models in which syntactic structure is modelled with a discrete stack which can be marginalized as a latent variable through dynamic programming. This enables us to investigate the trade-off between model expressivity and exact marginalization in probabilistic models based on recurrent neural networks (RNNs).

While Long Short-term Memory (Hochreiter and Schmidhuber, 1997) (LSTM) RNNs have driven strong improvements in intrinsic language modelling performance, they fail at capturing certain long-distance dependencies, such as those required for modelling subject-verb agreement (Linzen et al., 2016) or performing synthetic transduction tasks based on context-free grammars (Grefenstette et al., 2015). We propose generative models, based on transition-based dependency parsing (Nivre, 2008), a widely used framework for incremental syntactic parsing, that are able to capture desirable dependencies.

Our generative approach to dependency parsing encodes sentences with an RNN and estimate transition and next word probability distributions by conditioning on a small number of features represented by RNN encoder vectors. In contrast to previous syntactic language models such as RNNG (Dyer et al., 2016), marginal word probabilities can be computed both online and exactly. A GPU implementation which exploits parallelization enables unsupervised learning and fast training and decoding. The price of exact inference is that our models are less expressive than RNNG, as the recurrence is not syntax-dependent.

Our generative models are based on the arc-eager and arc-hybrid transition systems, with $O(n^3)$ dynamic programs based on Kuhlmann et al. (2011). Previous work on dynamic programming for transition-based parsing either required approximate inference due to a too high polynomial order run-time complexity (Huang and Sagae, 2010), or had too restrictive feature spaces to be used as accurate models (Kuhlmann et al., 2011; Cohen et al., 2011). Recent work showed that bidirectional RNNs enable accurate graph-based and transition-based dependency parsing using minimal feature spaces (Kiperwasser and Goldberg, 2016; Cross and Huang, 2016; Dozat and Manning, 2017). Shi et al. (2017) further showed that under this approach exact decoding

Proceedings of NAACL-HLT 2018, pages 942–952
New Orleans, Louisiana, June 1 - 6, 2018.

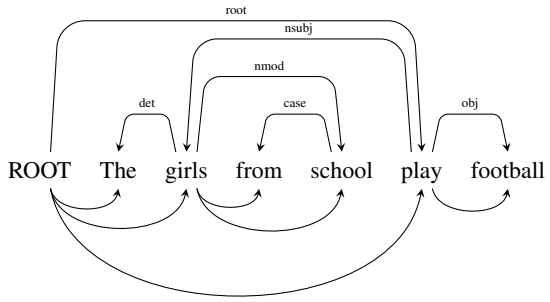


Figure 1: A dependency tree (arcs above words) together with dependencies captured by the generative model for word prediction (arcs below words).

Stack σ	Index $\beta - 1$	Prediction
ROOT	ROOT	sh(The)
ROOT, The	The	la(det)
ROOT	The	sh(girls)
ROOT, girls	girls	sh(from)
ROOT, girls, from	from	la(case)
ROOT, girls	from	sh(school)
ROOT, girls, school	school	ra(nmod)
ROOT, girls	school	la(nsubj)
ROOT	school	sh(play)
ROOT, play	play	sh(football)
ROOT, play, football	football	ra(obj)
ROOT, play	football	ra(root)
ROOT	football	re

Table 1: Arc-hybrid transition system derivation for the sentence "The girls from school play football." The transitions are shift (sh), reduce (re), left-arc (la) and right-arc (ra).

and globally-normalized discriminative training is tractable with dynamic programming.

While discriminative neural network-based models obtain state-of-the-art parsing accuracies (Dozat and Manning, 2017), generative models for structured prediction have a number of advantages: They do not suffer from label bias or explaining away effects (Yu et al., 2017), have lower sample complexity (Yogatama et al., 2017), are amenable to unsupervised learning and can model uncertainty and incorporate prior knowledge through latent variables.

As a supervised parser our model obtains state of the art performance in transition-based generative dependency parsing. While its intrinsic language modelling performance is worse than that of a well-tuned vanilla RNN, we see that the formulation of the generative model has a large impact on both the informedness of the syntactic structure and the parsing accuracy of the model. Furthermore, there is a discrepancy between the model structure most suitable for parsing and for language modeling. Our analysis shows that there exist informative syntactically-motivated dependencies which LSTMs are not capturing, even though our syntactic models are not able to predict them accurately enough during online processing to improve language modelling performance. Our implementation is available at `https://github.com/janmbuys/ndp-parser`.

2 Generative shift-reduce parsing

We start by defining a shift-reduce transition system which does not predict dependency arcs, but simply processes the words in a sentence left to right through shifting words onto a stack and reducing (popping) them from the stack. We define a generative model for this transition system and a dynamic program to perform inference over all possible shift-reduce transitions to process a given sentence. An example dependency tree is given in Figure 1, along with the dependencies our generative model is capturing when making word predictions. The arc-hybrid transition sequence for the example is given in Table 1.

Let sentence $\mathbf{w}_{0:n}$ be a sequence of words, where w_0 is always the designated root symbol `ROOT` and w_n the end-of-sentence symbol `EOS`. The state variables of the transition system are the stack σ, consisting of word indexes, and a current word index β, also referred to as the buffer. The first and second elements on the stack are referred to as σ_0 and σ_1, respectively. We use the notation $\sigma|j$ to indicate that j is on top of the stack. The initial state (σ, β) is $([0], 1)$ and the final state is $([], n)$. There are two transition actions, *shift* and *reduce*. Shift updates the transition state from (σ, j) to $(\sigma|j, j+1)$. Reduce changes the state from $(\sigma|i, j)$ to (σ, j).

2.1 Generative model

The generative model for this transition system is defined by a probability distribution over $\mathbf{w}$,

$$p(\mathbf{w}) = \sum_{\mathbf{t}} p(\mathbf{w}_{0:n}, \mathbf{t}_{0:2n}), \qquad (1)$$

where $\mathbf{t}_{0:2n}$ is a transition sequence that processes the sentence. Shift actions predict (assign probability to) the next word in the sentence. The end-

of-sentence symbol is generated implicitly when ROOT is reduced from the stack.

The sentence is encoded left-to-right by an LSTM RNN taking the word embedding of the last predicted word as input at each time step, independent of $\mathbf{t}$. The RNN hidden states $\boldsymbol{h}_{0:n}$ represent each sentence position in its linear context. The probability of a shift action and the word that it predicts is

$$p_{\text{tr}}(\text{sh}|h_{\sigma_0}, h_{\beta-1})p_{\text{gen}}(w|h_{\sigma_0}, h_{\beta-1}).$$

Reduce is predicted with probability $p_{\text{tr}}(\text{re}|h_{\sigma_0}, h_{\beta-1}) = 1 - p_{\text{tr}}(\text{sh}|h_{\sigma_0}, h_{\beta-1})$.

The transition and word probability distributions are estimated by non-linear output layers that take the context-depending RNN representations of positions in the transition system as input,

$$p_{\text{tr}} = \text{sigmoid}(r^T \text{relu}(W_{ts}h_{\sigma_0} + W_{tb}h_{\beta-1})) \quad (2)$$

$$p_{\text{gen}} = \text{softmax}(R^T \tanh(W_{gs}h_{\sigma_0} + W_{tb}h_{\beta-1})), \quad (3)$$

where R and the W's are neural network parameter matrices and r is a parameter vector.

The model has two ways of representing context: The RNN encoding, which has a recency bias, and the stack, which can represent long range dependencies and has a syntactic distance bias. The choice of RNN states (corresponding to stack elements) to condition on is restricted by our goal of making the dynamic programming tractable.

We propose two formulations of the generative model: In the first, referred to as *stack-next*, shift generates the word pushed on the stack, which is currently at position β, as in the equations above. In the second formulation, referred to as *buffer-next*, shift generates the word at position $\beta+1$, i.e., the next word on the buffer. The first formulation has a more intuitive generative story as the generation of a word is conditioned on the top of the stack when it is generated (see Table 1), but the second formulation has the advantage that transition predictions are conditioned on the current word at position β, which is more informative for parsing predictions. Models are defined using *stack-next* unless stated otherwise.

2.2 Dynamic program

We now define a dynamic program for this model, based on the algorithms proposed by Kuhlmann et al. (2011) and their application to generative dependency parsing (Cohen et al., 2011).

The key to the dynamic program is the decomposition of the transition sequence into *push computations*. Each push computation is a sequence of transitions which results in a single node having been pushed to the stack. The simplest push computation is a single shift operation. Push computations can be composed recursively: combining two consecutive push computations followed by a reduce transition yields a new push operation. Therefore the derivation of a sentence under the transition system can be seen as a composition of push computations.

Items in the deduction system (Shieber et al., 1995) of the dynamic program have the form $[i, j]$, which has the interpretation that there exists a push computation between actions a_k and a_l such that $\beta = i$ at time step k and $\sigma_0 = i$ and $\beta = j$ at time step l. In the deduction system $[0, 1]$ is an axiom, $[0, n]$ is the goal and the deduction rules corresponding to the transitions are

$$[i, j-1] \rightarrow [j-1, j] \qquad \text{(shift)}$$
$$[i, k][k, j] \rightarrow [i, j] \qquad \text{(reduce)}.$$

The marginal probability distribution is computed by defining the inside score $I(i, j) = p(\mathbf{w}_{i:j-1})$ for every deduction system item. Computing the sentence probability corresponds to computing the inside score of the goal, $I(0, n) = p(\mathbf{w}_{0:n-1})$, followed by computing the final reduce probability.

Reduce probabilities are computed conditioned on positions k and j, which are accessible through the dynamic program deduction rule. However the shift probabilities cannot be computed at the *shift* rule for deducing $[j-1, j]$, as it does not have access there to the top of the stack. One solution is to extend the deduction system to a three-tuple that can track the value of an additional position, leading to a $O(n^4)$ dynamic program. Instead Shi et al. (2017) showed that the computation can be performed in the $O(n^3)$ algorithm by computing the shift probability of word k during the *reduce* deduction, as it was generated when i was on top of the stack. The inside algorithm is given in Algorithm 1.

To train the model without supervised transition sequences, we can optimize the negative log likelihood of $p(w_{0:n})$ directly with gradient-based optimization using automatic differentiation, which

Algorithm 1 Inside algorithm for the shift-reduce transition-based generative model.

1: $I(0,1) \leftarrow 1$
2: **for** $j = 2, \ldots, n$ **do**
3: $\quad I(j-1,j) \leftarrow 1$
4: $\quad$ **for** $i = j-2, \ldots, 0$ **do**
5: $\quad\quad$ **for** $k = i+1, \ldots, j-1$ **do**
6: $\quad\quad\quad T(k) \leftarrow p_{\text{tr}}(\text{sh}|h_i, h_{k-1}) p_{\text{gen}}(w_k|h_i, h_{k-1})$
7: $\quad\quad$ **end for**
8: $\quad\quad I(i,j) \leftarrow \sum_{k=i+1}^{j-1} I(i,k) I(k,j) p_{tr}(\text{re}|h_k, h_{j-1}) T(k)$
9: $\quad$ **end for**
10: **end for**
11: **return** $I(0,n) + p_{tr}(\text{re}|h_0, h_{n-1})$

is equivalent to computing the gradients with the outside algorithm (Eisner, 2016). For decoding we perform Viterbi search over the dynamic program by maximizing rather than summing over different split positions (values of k when reducing).

The *buffer-next* generative formulation, where shift generates the next word β, can also be computed with the dynamic program. Here w_1 is predicted at the initial state in $I(0,1)$, while the end-of-sentence token is generated explicitly when a shift action results in buffer being set to position n, regardless of the state of the stack.

3 Transition-based dependency parsing

The arc-eager (Nivre, 2008) and arc-hybrid (Kuhlmann et al., 2011) transition systems for projective dependency parsing use the same shift-reduce operations but predict left- and right-arcs at different time steps. We propose generative models for these transition systems based on the dynamic program for shift-reduce parsing proposed above, again following Kuhlmann et al. (2011). For supervised training we optimize the joint probability distribution $p(\mathbf{w}, \mathbf{t})$, where an oracle is used to derive transition sequence $\mathbf{t}$ from the training examples. In cases of spurious ambiguity arcs are added as soon as possible.

3.1 Arc-hybrid parser

The arc-hybrid transition system has three actions: Shift, left-arc and right-arc (see Table 2 for definitions). *Left-arc* and *right-arc* are both reduce actions, but they add arcs between different word pairs. Arc label predictions are conditioned on the same context as transition predictions. Right-arc adds a dependency of which σ_1 is the head, but the dynamic program does not allow conditioning on it when making transition decisions. However, we found that this does not actually decrease performance.

The dynamic program for the arc-hybrid parser has the same structure as the shift-reduce model. The marginal probability is independent of arc directionality, as it does not influence future decisions. Consequently unsupervised training based on this model cannot learn to predict arc directions. Exact decoding is performed with the Viterbi algorithm: At every item $[i,j]$ the highest scoring arc direction is recorded. After the most likely transition sequence is extracted, arc labels are predicted greedily.

3.2 Arc-eager parser

The arc-eager parser has four transitions, as defined in Table 2. *Shift* and *right-arc* are shift actions, while *left-arc* and *reduce* are reduce actions. However the two reduce actions, reduce and left-arc, are always mutually exclusive; the former is only valid if the stack top has already been assigned a head (through a previous right-arc) and the latter only if the stack top is not headed. To keep track of which actions are valid, the state configuration and the dynamic program are augmented to record whether elements on the stack are headed. As with arc-hybrid, we decompose the transition probability into deciding between shifting and reducing, and then predicting directionality. In this case, the shift decision decomposes into *shift* and *right-arc* transitions, where *shift* is implicitly deciding that the shifted word will be reduced through a left-arc. Consequently the only real difference between the arc-hybrid and arc-eager transition systems under dynamic programming is the information conditioned on when arc directionality is predicted.

A different deduction system is defined for arc-eager, although it follows the same structure as the shift-reduce one. Items have the form $[i^c, j]$, where c is a binary variable indicating whether node i is headed. The axiom and goal are $[0^0, n]$ and $[0^0, 1]$, respectively. The deduction rules are

$$[i^c, j] \rightarrow [j^0, j+1] \quad \text{(shift)}$$
$$[i^c, j] \rightarrow [j^1, j+1] \quad \text{(right-arc)}$$
$$[i^c, k][k^0, j] \rightarrow [i^c, j] \quad \text{(left-arc)}$$
$$[i^c, k][k^1, j] \rightarrow [i^c, j] \quad \text{(reduce)}$$

The inside algorithm for arc-eager parsing is given in Algorithm 2. The algorithm is structured such that the inner loop computations (lines $8-22$) can be vectorized, which is crucial for efficient

Action	State before	State after	Arc added	Probability
Shift	$(\sigma\|i,j)$	$(\sigma\|i\|j,j+1)$	-	$p_{tr}(\text{sh}\|h_i,h_{j-1})p_{gen}(w_j\|h_i,h_{j-1})$
Left-arc	$(\sigma\|i,j)$	(σ,j)	$j \to i$	$p_{tr}(\text{re}\|h_i,h_{j-1})p_{dir}(\text{la}\|h_i,h_{j-1})$
Right-arc	$(\sigma\|l\|i,j)$	$(\sigma\|l,j)$	$l \to i$	$p_{tr}(\text{re}\|h_i,h_{j-1})p_{dir}(\text{ra}\|h_i,h_{j-1})$
Shift	$(\sigma\|i^b,j)$	$(\sigma\|i^b\|j^0,j+1)$	-	$p_{tr}(\text{sh}\|h_i,h_{j-1})p_{dir}(\text{la}\|h_i,h_{j-1})p_{gen}(w_j\|h_i,h_{j-1})$
Right-arc	$(\sigma\|i^b,j)$	$(\sigma\|i^b\|j^1,j+1)$	$i \to j$	$p_{tr}(\text{sh}\|h_i,h_{j-1})p_{dir}(\text{ra}\|h_i,h_{j-1})p_{gen}(w_j\|h_i,h_{j-1})$
Left-arc	$(\sigma\|i^0,j)$	(σ,j)	$j \to i$	$p_{tr}(\text{re}\|h_i,h_{j-1})$
Reduce	$(\sigma\|i^1,j)$	(σ,j)	-	$p_{tr}(\text{re}\|h_i,h_{j-1})$

Table 2: The arc-hybrid (above) and arc-eager (below) transition systems. States represent (stack, current index).

Algorithm 2 Inside algorithm for arc-eager parser.

for $j = 0, \ldots, n-1$ **do**
  $I(j^0, j{+}1) \leftarrow 1$
  $I(j^1, j{+}1) \leftarrow 1$
end for
for $gap = 2, \ldots, n$ **do**
  for $i = 0, \ldots, n - gap$ **do**
    $j = i + gap$
    for $c = 0, 1$ **do**
      for $k = i+1, \ldots, j-1$ **do**
        if $j > 0$ **then**
          $W(k) \leftarrow p_{\text{tr}}(\text{sh}|h_i, h_{k-1})$
            $\cdot p_{\text{dir}}(\text{ra}|h_i, h_{k-1}) p_{\text{gen}}(w_k|h_i, h_{k-1})$
          $T(k) \leftarrow I(k^1, j) p_{\text{tr}}(\text{re}|h_k, h_{j-1}) W(k)$
        end if
        if $j < n$ **then**
          $V(k) \leftarrow p_{\text{tr}}(\text{sh}|h_i, h_{k-1})$
            $\cdot p_{\text{dir}}(\text{la}|h_i, h_{k-1}) p_{\text{gen}}(w_k|h_i, h_{k-1})$
          $T(k) \leftarrow T(k) + I(k^0, j) p_{\text{tr}}(\text{re}|h_k, h_{j-1})) V(k)$
        end if
      end for
      $I(i^c, j) \leftarrow \sum_{k=i+1}^{j-1} I(i^c, k) T(k)$
    end for
  end for
end for
return $I(0, n) + p_{tr}(\text{re}|h_0, h_{n-1})$

GPU implementation. At $\beta = n$, the dynamic program is restricted to allow only reduce transitions, requiring the remaining stack elements (apart from ROOT) to be headed. The Viterbi algorithm again follows the same structure as the inside algorithm: For every item $[i^c, j]$ the highest scoring splitting item k^b is recorded, where k is the splitting point and b indicates whether word k is headed or not, which corresponds to whether a reduce or left-arc is performed.

4 Experiments

We follow the standard setup for English dependency parsing, training on sections 2-21 of the Penn Treebank (PTB) Wall Street Journal corpus, using section 22 for development and section 23 for testing. Dependency trees follow the Stanford dependency (SD) representation (version 3.3.0) used in recent parsing research (Chen and Manning, 2014; Dyer et al., 2015). We also report some results using the older representation of Yamada and Matsumoto (2003) (YM). We follow Buys and Blunsom (2015b) and Dyer et al. (2016) in replacing training singletons and unknown words in the test set with unknown word class tokens based to their surface forms, following the rules implemented in the Berkeley parser.[1]

Our models are implemented in PyTorch, which constructs computation graphs dynamically.[2] During training, sentences are shuffled at each epoch, and minibatches are constructed of sentences of the same length. We base the hyperparameters of our models primarily on the language models of Zaremba et al. (2014). Models are based on two-layer LSTMs with embedding and hidden state size 650 with dropout of 0.5 on the RNN inputs and outputs. For all models weights are initialized randomly from the uniform distribution over $[-0.05, 0.05]$. Gradient norms are clipped to 5.0. The supervised parsers are trained with batch size 16, with an initial learning rate 1.0, which is decreased by a factor of 1.7 for every epoch after 6 initial epochs. The sequential LSTM baseline is trained with the same parameters, except that the learning rate decay is 1.4. The unsupervised models are trained with an initial learning rate 0.1, which is decreased by a factor of 2.0 for every epoch, with batch size 8.

We train and execute our models on a GPU, obtaining significant speed improvements over CPUs. For supervised training we also perform batch processing: After the sentences are encoded with an RNN, we extract the inputs to the transition, word and relation prediction models across the batch, and then perform the neural network computations in parallel. The supervised models' training speed is about 3 minutes per epoch.

[1] http://github.com/slavpetrov/berkeleyparser

[2] http://pytorch.org/

Model	Greedy	Exact
Arc-Hybrid uniRNN	84.12/81.54	84.21/81.61
Arc-Eager uniRNN	79.90/77.67	81.37/79.08
Arc-Hybrid biRNN	92.85/90.42	92.89/90.47
Arc-Eager biRNN	92.82/90.63	**92.90/90.68**
Arc-Hybrid Gen stack-next	56.98/52.22	82.77/78.01
Arc-Hybrid Gen buffer-next	85.25/82.83	**91.19/88.66**
Arc-Eager Gen buffer-next	80.79/78.56	87.34/84.84

Table 3: PTB development set parsing results (SD dependencies), reporting unlabelled and labelled attachment scores (UAS/LAS). Discriminative models (above the line) use either unidirectional or bidirectional RNNs.

Model	SD	YM
Buys and Blunsom (2015b)	90.10/87.74	90.16/88.83
Titov and Henderson (2007)	91.43/89.02	90.75/89.29
Arc-eager	88.20/85.91	87.61/86.36
Arc-hybrid	**91.01/88.54**	**90.71/88.68**

Table 4: PTB test set parsing results with supervised generative models, on the Stanford (SD) and Yamada and Matsumoto (2003) (YM) dependencies. The models from Buys and Blunsom (2015b) and Titov and Henderson (2007) were retrained to make results directly comparable.

4.1 Parsing

In order to benchmark parsing performance, we train discriminative baselines using the same feature space as the generative models. Unidirectional or bidirectional RNNs can be used; we see that the bidirectional encoder is crucial for accuracy (Table 3). The performance of our implementation is on par with than that of the arc-hybrid transition-based parser of Kiperwasser and Goldberg (2016), which obtains 93.2/91.2 UAS/LAS on the test set against 93.29/90.83 for our arc-hybrid model. State of the art parsing performance is 95.7%/94.1 UAS/LAS (Dozat and Manning, 2017).

Exact decoding is only marginally more accurate than greedy decoding, giving further evidence of the label bias problem. Andor et al. (2016) similarly showed that a locally normalised model without lookahead features cannot obtain good performance even with beam-search (81.35% UAS), while their globally normalised model can reach close to optimal performance without look-ahead. Shi et al. (2017) showed that globally normalised training improves the accuracy of these discriminative models.

Exact decoding is crucial to the performance of the generative models (Table 3). They are much more accurate than the unidirectional discriminative models, which shows that the word prediction model benefits parsing accuracy. The arc-hybrid model is more accurate than arc-eager, as was the case for the unidirectional discriminative models. This can be explained by arc-eager making attachment decisions earlier in the transition sequence than arc-hybrid, which means that it has access to less context to condition these predictions on.

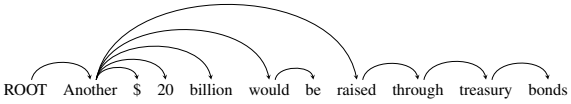


Figure 2: Sentence with dependencies induced by the unsupervised model.

Our best generative model outperforms a previous incremental generative dependency parser based on feed-forward neural networks and approximate inference (Buys and Blunsom, 2015b) (Table 4). It is competitive with a previous RNN-based generative parser with a much more complex architecture than our model, including recurrent connections based on parsing decision (Titov and Henderson, 2007). Our exact decoding algorithm is also actually faster than the beam-search approaches for previous models, as it is implemented on GPU. Our arc-hybrid model parses 7.4 sentences per second, against 4 sentences per second for Buys and Blunsom (2015b) and approximately 1 sentence per second for Titov and Henderson (2007).

We also train the model as an unsupervised parser by directly optimizing the marginal sentence probability. The limitation of our approach is that our models cannot learn arc directionality without supervision, so we interpret shift as adding a (right-arc) dependency between top of the stack and the word being generated. In our experiments the model did not succeed in learning informative, non-trivial tree structures – in most cases it learns to attach words either to the immediate previous word or to the root. However, unsupervised dependency parsers usually require elaborate initialization schemes or biases to produce non-trivial trees (Klein and Manning, 2004; Spitkovsky et al., 2010; Bisk and Hockenmaier, 2015). An example dependency tree predicted by the unsupervised model is given in Figure 2.

Model	Perplexity
Interpolated Kneser-Ney 5-gram	170.09
Sequential LSTM (unbatched)	118.69
Sequential LSTM (batched)	**100.67**
Buys and Blunsom (2015b)	138.62
RNNG (Kuncoro et al., 2017)	**101.2**
Supervised (SD) shift-reduce buffer-next	111.53
Supervised (SD) shift-reduce stack-next	**107.61**
Unsupervised shift-reduce stack-next	125.20

Table 5: Language modelling perplexity results on the PTB parsing test set.

4.2 Language modelling

We apply our model to language modelling with both supervised and unsupervised training. The supervised models are trained as arc-hybrid parsers; the performance of arc-eager is almost identical as arc labels and directionality are not predicted. The unsupervised model is trained with only shift and reduce transitions as latent.

We evaluate language models with a sentence i.i.d. assumption. In contrast, the standard evaluation setup for RNN language models treats the entire corpus as single sequence. To evaluate the consequence of the sentence independence assumption, we trained a model on the most widely used PTB language modelling setup (Chelba and Jelinek, 2000; Mikolov et al., 2011), which uses a different training/testing split and preprocessing which limits the vocabulary to 10k. Our baseline LSTM obtains 92.71 test perplexity on this setup, against 78.4 of Zaremba et al. (2014), which uses the same hyperparameters without a sentence i.i.d. assumption. The syntactic neural language model of Emami and Jelinek (2005) obtained 131.3.

Results are reported in Table 5. Perplexity is obtained by exponentiating the negative log likelihood per token; end of sentence symbols are predicted but excluded from the token counts. As baselines without syntactic structure we use the interpolated Kneser-Ney n-gram model (Kneser and Ney, 1995) and vanilla LSTMs, trained with or without batching.

The LSTM baselines already outperform the syntactic feed-forward neural model of Buys and Blunsom (2015b). We see that there is a significant difference between training with or without mini-batching for the baseline; similarly our model's perplexities also improve when trained with batching. The batched baseline performs slightly better than Recurrent Neural Network Grammars (RNNG) (Dyer et al., 2016; Kuncoro et al., 2017), a constituency syntax-based RNN language model trained without batching.[3]

The results show that our syntactic language models perform slightly worse than the LSTM baseline. We experimented with different dependency representations on the development set, including SD, YM and Universal Dependencies (Nivre et al., 2016). We found little difference in language modelling performance between the dependency representations. Unsupervised training does not lead to better perplexity than the supervised models; however, due to much longer training times we did less hyperparameter tuning for the unsupervised model.

4.3 Analysis

We further analyze the probability distribution that our model is learning by calculating some perplexity-related quantities. We compare the perplexity of the marginal distribution $p(\mathbf{w})$ to the perplexity based only on the most likely transition sequence $\mathbf{a} = \text{argmax}\ p(\mathbf{w}, \mathbf{a})$, based on either the joint distribution $p(\mathbf{w}, \mathbf{a})$ or the conditional distribution $p(\mathbf{w}|\mathbf{a})$. Note that while the former is a bound on the marginal perplexity, the latter is not a true perplexity but simply helps us to quantify the contribution of the syntactic structure to reducing the uncertainty in the prediction.

The results (Table 6) show that the difference between the joint and marginal perplexities are relatively small for the supervised models, indicating that the distribution is very peaked around the most likely parse trees. However the conditional quantity shows that the syntactic structure encoded by the *stack-next* model is much more informative than that of the *buffer-next* model, although the only difference between them is the choice of elements to condition on when predicting the next word. Although the *stack-next* model has a better marginal perplexity, the disadvantage is that it has more uncertainty in the syntactic structure it is predicting (as can be seen by lower parsing accuracy) even though that structure is more informative.

The strength of RNNG over our approach is that it computes a compositional representation of

[3]Our experimental setup is the same as Dyer et al. (2016), except for a minor implementation difference in unknown word clustering; Dyer et al. (2016) reports 169.31 perplexity on the same IKN model.

Model	Marginal ppl	Argmax parse joint ppl	Argmax parse conditional ppl
RNNG (Dyer et al., 2016)	104.10	107.58	41.60
Supervised (SD) shift-reduce buffer-next	111.53	120.09	102.28
Supervised (SD) shift-reduce stack-next	107.61	119.20	71.27
Unsupervised shift-reduce stack-next	125.20	350.01	169.87

Table 6: Language modelling perplexity analysis on the PTB test set.

the stack and the partially constructed parse tree, while our model can only make use of the position on top of the stack and otherwise has to rely on the sequentially computed RNN representations. The disadvantage of RNNG is that inference can only be performed over entire sentences, as the proposal distribution for their importance sampling method is a discriminative parser. Exact inference allows our models to estimate next word probabilities from partially observed sequences.

5 Related work

5.1 Syntactic generative models

Chelba and Jelinek (2000) and Emami and Jelinek (2005) proposed incremental syntactic language models that predict binarized constituency trees with a shift-reduce model, parameterized by interpolated n-gram smoothing and feed-forward neural networks, respectively. Language modelling probabilities were approximated incrementally using beam-search. Rastrow et al. (2012) applied a transition-based dependency n-gram language model to speech recognition. These models obtained perplexity improvements primarily when interpolated with standard n-gram models, and were not employed as parsers.

Henderson (2004) proposed an incremental constituency parser based on recurrent neural networks that have additional connections to previous recurrent states based on the parser configuration at each time step. The generative version of this model was more accurate than the discriminative one. Titov and Henderson (2007) applied a similar approach to dependency parsing. Buys and Blunsom (2015a) and Buys and Blunsom (2015b) proposed generative syntactic models that are applied to both dependency parsing and language modelling, using Bayesian and feed-forward neural networks, respectively. Recurrent Neural Network Grammar (RNNG) (Dyer et al., 2016) is a generative transition-based constituency parser based on stack LSTMs (Dyer et al., 2015), that was also applied as a language model.

Recently, Shen et al. (2017) proposed an RNN-based language model that uses a soft gating mechanism to learn structure that can be interpreted as constituency trees, reporting strong language modelling performance. There has also been work on non-incremental syntactic language modelling: Mirowski and Vlachos (2015) proposed a dependency neural language model where each word is conditioned on its ancestors in the dependency tree, and showed that this model achieves strong performance on a sentence completion task.

5.2 Neural models with latent structure

There have been a number of recent proposals for neural abstract machines that augment RNNs with external memory, including stacks and other data structures that are operated on with differentiable operations to enable end-to-end learning. Neural Turing machines (Graves et al., 2014) have read-write memory that is updated at each timestep. Grefenstette et al. (2015) proposed a neural stack that is operated on with differentiable push and pop computations.

Another strand of recent work which our models are related to has proposed neural models with structured latent variables: Rastogi et al. (2016) incorporated neural context into weighted finite-state transducers with a bidirectional RNN, while Tran et al. (2016) proposed a neural hidden Markov model for Part-of-Speech (POS) induction. Yu et al. (2016) proposed a neural transduction model with polynomial-time inference where the alignment is a latent variable. Kim et al. (2017) proposed structured attention mechanisms that compute features by taking expectations over latent structure. They define a tree-structured model with a latent variable for head selection,

along with projectivity constraints. The soft head selection learned by the model is used as features in an attention-based decoder.

Reinforcement learning has been proposed to learn compositional tree-based representations in the context of an end task (Andreas et al., 2016; Yogatama et al., 2016), but this approach has high variance and provide no guarantees of finding optimal trees.

6 Conclusion

We proposed a new framework for generative models of syntactic structure based on recurrent neural networks. We presented efficient algorithms for training these models with or without supervision, and to apply them to make online predictions for language modelling through exact marginalization. Results show that the model obtains state-of-the-art performance on supervised generative dependency parsing, but does not obtain better intrinsic language modelling performance than a standard RNN.

Acknowledgments

We thank members of the Oxford NLP group for discussions, Yejin Choi for valuable feedback, and the anonymous reviewers for their comments.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of ACL*. Association for Computational Linguistics, Berlin, Germany, pages 2442–2452. http://www.aclweb.org/anthology/P16-1231.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Learning to compose neural networks for question answering. In *Proceedings of NAACL*. pages 1545–1554. http://www.aclweb.org/anthology/N16-1181.

Yonatan Bisk and Julia Hockenmaier. 2015. Probing the linguistic strengths and limitations of unsupervised grammar induction. In *Proceedings of ACL*. Beijing,China, pages 1395–1404.

Jan Buys and Phil Blunsom. 2015a. A Bayesian model for generative transition-based dependency parsing. In *Proceedings of the 3rd International Conference on Dependency Linguistics (Depling)*.

Jan Buys and Phil Blunsom. 2015b. Generative incremental dependency parsing with neural networks. In *Proceedings of ACL-IJCNLP (short papers)*. pages 863–869.

Ciprian Chelba and Frederick Jelinek. 2000. Structured language modeling. *Computer Speech & Language* 14(4):283–332.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *EMNLP*.

Shay B. Cohen, Carlos Gómez-Rodríguez, and Giorgio Satta. 2011. Exact inference for generative probabilistic non-projective dependency parsing. In *Proceedings of EMNLP*. pages 1234–1245.

James Cross and Liang Huang. 2016. Incremental parsing with minimal features using bi-directional LSTM. In *Proceedings of ACL*. page 32.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of ICLR*. http://arxiv.org/abs/1611.01734.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of ACL*. pages 334–343. http://www.aclweb.org/anthology/P15-1033.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of NAACL*.

Jason Eisner. 2016. Inside-outside and forward-backward algorithms are just backprop (tutorial paper). In *Proceedings of the Workshop on Structured Prediction for NLP*. Association for Computational Linguistics, Austin, TX, pages 1–17. http://aclweb.org/anthology/W16-5901.

Ahmad Emami and Frederick Jelinek. 2005. A neural syntactic language model. *Machine Learning* 60(1-3):195–227. https://doi.org/10.1007/s10994-005-0916-y.

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

Edward Grefenstette, Karl Moritz Hermann, Mustafa Suleyman, and Phil Blunsom. 2015. Learning to transduce with unbounded memory. In *Advances in Neural Information Processing Systems*. pages 1828–1836.

James Henderson. 2004. Discriminative training of a neural network statistical parser. In *Proceedings of ACL*. Association for Computational Linguistics, page 95.

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

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of ACL*. pages 1077–1086. http://www.aclweb.org/anthology/P10-1110.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. 2017. Structured attention networks. In *Proceedings of ICLR*.

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

Dan Klein and Christopher D Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *ACL*. pages 478–586.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *ICASSP*. IEEE, volume 1, pages 181–184.

Marco Kuhlmann, Carlos Gómez-Rodríguez, and Giorgio Satta. 2011. Dynamic programming algorithms for transition-based dependency parsers. In *Proceedings of ACL*. pages 673–682.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. 2017. What do recurrent neural network grammars learn about syntax? In *Proceedings of EACL*. Association for Computational Linguistics, Valencia, Spain, pages 1249–1258. http://www.aclweb.org/anthology/E17-1117.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics* 4:521–535. https://www.transacl.org/ojs/index.php/tacl/article/view/972.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *ICASSP*. IEEE, pages 5528–5531.

Piotr Mirowski and Andreas Vlachos. 2015. Dependency recurrent neural language models for sentence completion. *CoRR* abs/1507.01193. http://arxiv.org/abs/1507.01193.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics* 34(4):513–553.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.

Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. 2016. Weighting finite-state transductions with neural context. In *Proceedings of NAACL*.

Ariya Rastrow, Mark Dredze, and Sanjeev Khudanpur. 2012. Efficient structured language modeling for speech recognition. In *INTERSPEECH*. http://interspeech2012.org/accepted-abstract.html?id=1436.

Yikang Shen, Zhouhan Lin, Chin-Wei Huang, and Aaron C. Courville. 2017. Neural language modeling by jointly learning syntax and lexicon. *CoRR* abs/1711.02013. http://arxiv.org/abs/1711.02013.

Tianze Shi, Liang Huang, and Lillian Lee. 2017. Fast(er) exact decoding and global training for transition-based dependency parsing via a minimal feature set. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 12–23. https://www.aclweb.org/anthology/D17-1002.

Stuart M Shieber, Yves Schabes, and Fernando CN Pereira. 1995. Principles and implementation of deductive parsing. *The Journal of logic programming* 24(1):3–36.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2010. From baby steps to leapfrog: how less is more in unsupervised dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. pages 751–759.

Ivan Titov and James Henderson. 2007. A latent variable model for generative dependency parsing. In *Proceedings of the Tenth International Conference on Parsing Technologies*. pages 144–155. http://www.aclweb.org/anthology/W/W07/W07-2218.

Ke M. Tran, Yonatan Bisk, Ashish Vaswani, Daniel Marcu, and Kevin Knight. 2016. Unsupervised neural hidden markov models. In *Proceedings of the Workshop on Structured Prediction for NLP*. Association for Computational Linguistics, Austin, TX, pages 63–71. http://aclweb.org/anthology/W16-5907.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of the International Conference on Parsing Technologies*. volume 3.

D. Yogatama, C. Dyer, W. Ling, and P. Blunsom. 2017. Generative and discriminative text classification with recurrent neural networks. *CoRR* abs/1703.01898. http://arxiv.org/abs/1703.01898.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2016. Learning to compose words into sentences with reinforcement learning. *CoRR* abs/1611.09100.

Lei Yu, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Tomas Kocisky. 2017. The neural noisy channel. *Proceedings of ICLR* .

Lei Yu, Jan Buys, and Phil Blunsom. 2016. Online segment to segment neural transduction. In *Proceedings of EMNLP*. pages 1307–1316.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *CoRR* abs/1409.2329. http://arxiv.org/abs/1409.2329.

Noise-Robust Morphological Disambiguation for Dialectal Arabic

Nasser Zalmout, Alexander Erdmann, Nizar Habash
Computational Approaches to Modeling Language Lab
New York University Abu Dhabi
United Arab Emirates
{nasser.zalmout,ae1541,nizar.habash}@nyu.edu

Abstract

User-generated text tends to be noisy with many lexical and orthographic inconsistencies, making natural language processing (NLP) tasks more challenging. The challenging nature of noisy text processing is exacerbated for dialectal content, where in addition to spelling and lexical differences, dialectal text is characterized with morpho-syntactic and phonetic variations. These issues increase sparsity in NLP models and reduce accuracy. We present a neural morphological tagging and disambiguation model for Egyptian Arabic, with various extensions to handle noisy and inconsistent content. Our models achieve about 5% relative error reduction (1.1% absolute improvement) for full morphological analysis, and around 22% relative error reduction (1.8% absolute improvement) for part-of-speech tagging, over a state-of-the-art baseline.

1 Introduction

There has been a growing interest in noise-robust NLP tools recently, motivated by the sheer magnitude of user-generated content in social media platforms. The noisy nature of user-generated content makes its processing very challenging for NLP tools. Noisy content is non-canonical in nature, with lexical, orthographic, and phonetic variations that increase the perplexity and sparsity of NLP models. Several contributions show considerable drop in performance for a number of tasks, where simply retraining existing models with social media data does not provide substantial improvement (Gimpel et al., 2011; Ritter et al., 2011; Habash et al., 2013a).

Morphological disambiguation for noisy content is further complicated for dialectal content, with additional morpho-syntactic variations. Morphological disambiguation is also more challenging for morphologically rich and ambiguous languages, like Arabic and Dialectal Arabic (DA). Arabic is morphologically rich, having more fully inflected words (types) than morphologically poorer languages. It is also ambiguous, with short vowels (diacritic marks) often dropped and disambiguated in context. These issues result in more morpho-syntactic variations for DA in written text compared to other dialectal content, and increase the number of potential analyses.

We present several morphological disambiguation models for Egyptian Arabic (EGY), based on previous models for EGY and Modern Standard Arabic (MSA). We use a bidirectional long short term memory (Bi-LSTM) architecture and various noise reduction techniques, including character embedding and embedding space mapping. We also experiment with the width of the embedding window in the pre-trained embeddings. Character embeddings allow access to subword units, while the embedding space mapping normalizes non-canonical forms to canonical neighbors. The narrow/wide embedding window in the pre-trained embeddings allows for more of syntactic/semantic modeling, respectively.

The goal of the various models is to achieve noise-robust analysis, rather than explicit noise normalization. We therefore use the normalization techniques on the vector-level only, instead of replacing the raw forms, which allows for less aggressive lexical normalization. The separation of raw forms and vector normalization also allows for independent word and character level normalization, eliminating any propagation of error.

Our system achieves a 5% relative error reduction (1.1% absolute accuracy boost) over a state-of-the-art baseline, using a strict metric. Our noise-robust system also matches the performance of a version of the system trained and tested on a manually orthography-normalized copy of the data. This indicates that the system performs as well as could be expected without orthographic in-

Proceedings of NAACL-HLT 2018, pages 953–964
New Orleans, Louisiana, June 1 - 6, 2018.

consistency. We also present an error analysis of the system and identify areas of improvement.

The rest of the paper is structured as follows. We present common challenges to DA processing in Section 2. This is followed by background and related work in Section 3. We introduce the approach and various models in Section 4, and discuss the experimental setup and results in Section 5. We conclude and provide some directions for future work in Section 6.

2 Linguistic Issues

Dialectal Arabic, including EGY among other dialects, is the primarily spoken language used by native Arabic speakers in daily exchanges. The outbreak of social media platforms expanded the use of DA as a written language. The lack of a standard orthography (Habash et al., 2012a), combined with the fact that user-generated content in social media is prone to noise, increase sparsity and reduce performance.

EGY, similar to MSA, is also a morphologically complex language, having a number of morphological features, e.g., gender, number, person, mood, and attachable clitics. Moreover, the diacritization-optional orthography for Arabic (both DA and MSA) results in orthographic ambiguity, leading to several interpretations of the same surface forms. Richness of form increases model sparsity, and ambiguity makes disambiguation harder. One approach to model complexity, richness, and ambiguity uses *morphological analyzers*, also known as morphological dictionaries. Morphological analyzers are usually used to encode all potential word inflections in the language. A good morphological dictionary should return all the possible analyses of a surface word (ambiguity), and cover all the inflected forms of a word lemma (richness), covering all related features. The best analysis is then chosen through *morphological disambiguation*.

The set of morphological features that we model for EGY morphological disambiguation includes:

- Lexicalized features: lemma, diacritization.
- Non-lexicalized features: aspect, case, gender, person, part-of-speech (POS), number, mood, state, voice.
- Clitics: enclitics, like pronominal enclitics, negative particle enclitics; proclitics, like article proclitic, preposition proclitics, conjunction proclitics, question proclitics.

Despite the similarities, EGY and MSA have many differences that prevent MSA tools from being effectively utilized for EGY text. These include lexical, phonological, and morphological inconsistencies. Lexical differences can be numerous, beyond simple cognates, like the word ازاي *AzAy*[1] 'how' in EGY corresponds to the word كيف *kyf* in MSA. There are also many morphological differences, for example the MSA future proclitic /sa/+ (spelled +سـ s+) appears in EGY as either /ha/+ (+هـ) or /Ha/+ (+حـ). There are also many phonological variations between EGY and MSA that have direct implications on orthography as well. These include the consonant ث /θ/ in MSA, which can be mapped to either ت /t/ or س /s/ in EGY. These variations make the written EGY content more susceptible to noise and inconsistency. Table 1 shows an EGY sentence example, along with the set of potential analyses for a given word.

3 Background and Related Work

Explicit handling of noisy content in NLP has recently gained momentum with the increasing use of social media outlets. Notable contributions for POS tagging include the ARK tagger (Owoputi et al., 2013), which is targeted for online conversational text. ARK tagger uses conditional random fields with word clusters as features, obtained via Brown clustering (Brown et al., 1992), along with various lexical features. Gimpel et al. (2011) also use conditional random fields for POS tagging, trained on annotated Twitter content. Derczynski et al. (2013) use manually curated lists to map low frequency and out of vocabulary terms to more frequent terms. Noisy content has also been addressed for named entity recognition (Liu et al., 2011; Ritter et al., 2011; Aguilar et al., 2017), and syntactic parsing (Foster et al., 2011; Petrov and McDonald, 2012).

Most relevant to our work is the paper by van der Goot et al. (2017), where they use Word2vec (Mikolov et al., 2013) to find potential normalization candidates for non-canonical words on the lexical level, and rank them using a classifier. They experiment with various normalization and embedding settings, and they find that both normalization and pre-trained embeddings are helpful for the task of POS tagging.

[1] Arabic transliterations are in the Habash-Soudi-Buckwalter transliteration scheme (Habash et al., 2007).

AntA fyn? fy AlmHl **brdk** *?* انتا فين؟ في المحل **بردك** ؟

Where are you? Are you at the shop **still**?

diacritization	lemma	gloss	pos	prc3	prc2	prc1	prc0	per	asp	vox	mod	gen	num	stt	cas	enc0
bi+rad∼+ak	rad∼	response	noun	0	0	bi_prep	0	na	na	na	na	m	s	c	u	2ms:poss
barDak	**barDak**	**still**	**adv**	**0**	**0**	**0**	**0**	**na**	**i**	**na**	**na**	**m**	**s**	**i**	**u**	**0**
bi+rad∼+ik	rad∼	response	noun	0	0	bi_prep	0	na	na	na	na	m	s	c	u	2fs:poss
b+Aarud∼+ak	rad∼	return	verb	0	0	0	bi_prog	1	i	a	i	m	s	na	na	2ms:dobj
bard+ak	bard	cold	noun	0	0	0	0	na	na	na	na	m	s	c	u	2ms:poss

Table 1: An example highlighting the effect of non-standard and ambiguous orthography, along with rich morphology, on EGY morphological disambiguation. The word برضك *barDak* 'still' is provided in the example with the non-standard (non-CODA compliant) orthography بردك *bardak*, which can lead to different morphological analyses than the one intended in context.

The issue of noisy text processing is exacerbated for dialectal content. Most contributions focus on spelling/lexical variations, whereas dialectal content is further characterized with morpho-syntactic and phonetic variations that make automatic processing more challenging (Jørgensen et al., 2015). In addition to the issues of morphological complexity, ambiguity, and lack of standard orthography for MSA and DA. There has been several contributions covering various NLP tasks including morphological analysis, disambiguation, POS tagging, tokenization, lemmatization and diacritization, addressing both MSA and DA (Al-Sabbagh and Girju, 2010; Mohamed et al., 2012; Habash et al., 2012b, 2013a; Abdelali et al., 2016; Khalifa et al., 2016b). Notable contributions for both MSA and EGY include MADAMIRA (Pasha et al., 2014), a morphological disambiguation tool that uses morphological analyzers to handle complexity and ambiguity. MADAMIRA can automatically correct common spelling errors as a side effect of disambiguation, but does not include explicit processing steps for noisy content. A neural version of MADAMIRA for MSA is presented by Zalmout and Habash (2017), who use Bi-LSTMs and morphological tag embeddings. Their system shows significant improvement over MADAMIRA, but does not use any explicit character embeddings nor noise reduction techniques.

To address the lack of standardized orthography for DA, Habash et al. (2012a) proposed CODA, a Conventional Orthography for Dialectal Arabic. CODA presents a detailed description of orthographic guidelines, mainly for the purpose of developing DA computational models, applied to EGY, and later extended to several other Arabic dialects (Zribi et al., 2014; Saadane and Habash, 2015; Turki et al., 2016; Khalifa et al., 2016a; Jarrar et al., 2016; Habash et al., 2018). CODA-treated DA content should be less sparse and less noisy. Eskander et al. (2013) presented a tool to normalize raw texts into a CODA compliant version using the K-nearest neighbor algorithm. Scaling this tool to other dialects, however, is challenging due to the lack of training data.

Our morphological tagging architecture is similar to the work of Inoue et al. (2017) and Zalmout and Habash (2017), but we further experiment with CNN-based character embeddings, and pre-train the word embeddings. The architecture is also similar to the work of Heigold et al. (2017) and Plank et al. (2016) in terms of the character embeddings, both LSTM and CNN-based systems. Our architecture, however, uses neural language models for modeling lemmas and diacritized forms, and utilizes the word-level embeddings in various configurations to combat noise, as explained throughout the rest of the paper.

4 Approach

We present a morphological disambiguation model for EGY. We use an LSTM-based architecture for morphological tagging and language modeling for the various morphological features in EGY. We also experiment with several embedding models for words and characters, and present several approaches for noise-robust modeling on the raw form and vector levels.

We present the overall tagging and disambiguation architecture, in addition to the character embedding model, in 4.1. We then present the noise handling approaches in 4.2 and 4.3.

4.1 Morphological Tagging and Disambiguation Architecture

We use a similar disambiguation approach as in previous contributions for MSA and EGY (Habash and Rambow, 2005; Habash et al., 2009, 2013b).

The morphological disambiguation task is intended to choose the correct morphological analysis from the set of potential analyses, obtained from the morphological analyzer. The analyzer provides a set of morphological features for each given word. These features can be grouped into non-lexical features, where a tagger is used to predict the relevant morphological tag, handled through *morphological feature tagging*, and lexical features that need a language model (Roth et al., 2008), handled through *lexicalized feature language models*. The inflectional, clitic, and part-of-speech features are handled with a tagger, while the lexical features are handled with a language model.

4.1.1 Morphological Feature Tagging

Overall Architecture We use Bi-LSTM-based taggers for the morphological feature tagging tasks. Given a sentence of length L words $\{w_1, w_2, ..., w_L\}$, every word w_i is converted into vector v_i:

$$v_i = [v_{w_i}; v_{c_i}; v_{t_i}]$$

composed of the word embedding vector v_{w_i}, word-level characters embedding vector v_{c_i}, and the candidate morphological tag embedding vector v_{t_i}. This separation of word and character embedding vectors enables further noise handling on the word embedding level alone, with the character embeddings learnt from the raw forms without any modification. We pre-train the word embeddings using Word2vec (Mikolov et al., 2013).

We use two LSTM layers to model the relevant context for both directions of the target word, where the input is represented by the v_i vectors mentioned above:

$$\overrightarrow{h}_i = g(v_i, \overrightarrow{h}_{i-1})$$

$$\overleftarrow{h}_i = g(v_i, \overleftarrow{h}_{i+1})$$

where h_i is the context vector from the LSTM for each direction. We join both sides, apply a non-linearity function, and softmax to get a probability distribution. Figure 1 shows the architecture.

Character Embedding We use convolutional neural networks (CNN) and LSTM-based architectures for the character embedding vectors v_{c_i}, both applied to the character sequence within each word separately. LSTM-based architectures have been shown to outperform CNN-based character

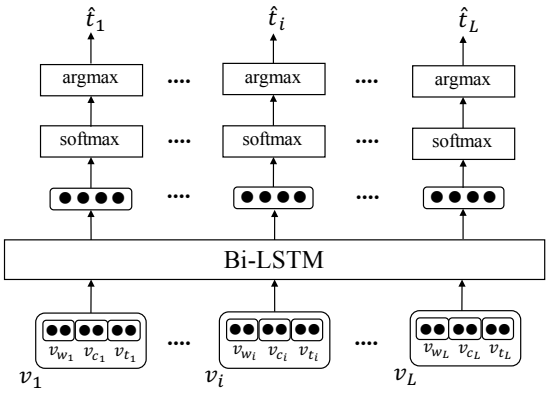


Figure 1: The overall tagging architecture, with the input vector as the concatenation of the word, characters, and candidate tag embeddings.

embedding in POS tagging (Heigold et al., 2017), but we experiment with both architectures to report their performance in noisy EGY content. We use various filter widths and max pooling for the CNN system, with the output fed to a dense connection layer. The resulting vector is used as the character embedding vector for the given word. For the LSTM-based architecture we use the last state vector as the embedding representation of the word's characters. Both architectures are outlined at figure 2.

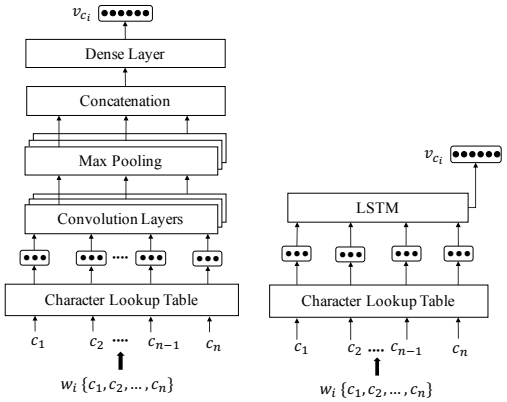


Figure 2: CNN-based (left) and LSTM-based (right) architectures for word-level character embedding. Similar to the architecture by Heigold et al. (2017).

Morphological Tag Embedding The morphological features vector v_{t_i} embeds the candidate tags for each feature. The tags include the collection of morphological features. We use the morphological analyzer to obtain all possible tag values of the word to be analyzed. We use a lookup table to map the tags to their trainable vector representation, then sum all the resulting vectors to

get v_{t_i}, since these tags are alternatives and do not constitute a sequence of any sort. Figure 3 outlines the tag embedding model.

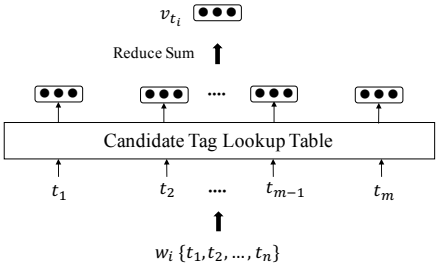


Figure 3: Candidate tag embedding, through summing the vectors of the individual tags.

Embedding the morphological tags using the analyzer does not constitute a hard constraint in the system, and the v_{t_i} vector can be discarded or substituted with less resource-demanding options for other languages or dialects.

4.1.2 Lexicalized Feature Language Models

We use LSTM-based neural language models (Enarvi and Kurimo, 2016) for the lexical features (lemma and diacritization). Lemmas and diacritized forms are lexical and cannot be modeled directly using a classifier (Habash and Rambow, 2007), since the target space is big (around 13K for lemmas, and 33K for the diacritized forms, in Train). We therefore use a language model to choose among the candidate lemmas and diacritized forms obtained from the analyzer. We encode the runtime dataset in the HTK Standard Lattice Format (SLF), with a word mesh representation for the various options of each word.

4.2 Embedding Window Width

Several contributions show that the window size (i.e. amount of context) in word embeddings affects the type of linguistic information that gets modeled. Goldberg (2016) and Trask et al. (2015) explain that larger windows tend to create more semantic and topical embeddings, whereas smaller windows capture syntactic similarities. Tu et al. (2017) also find that a window of one (one word before the target word and one word after) is optimal for syntactic tasks.

We experiment with both wide and narrow window embeddings, and evaluate their effects on tagging accuracy. These experiments show the role of topical or semantic vs syntactic embeddings in the morphological disambiguation model.

We then experiment with embedding vector extension, by combining both wide and narrow embeddings through concatenation. This technique is expected to handle noisy and unstandardized spellings, since spelling variants are not just semantically related, but must share the same syntactic valency.

Figure 4 shows the updated architecture, with the narrow window embedding $v_{w_i}^{narrow}$ concatenated to the v_i vector, along with the existing wide window embedding $v_{w_i}^{wide}$.

4.3 Embedding Space Mapping

The embedding space mapping approach is based on the hypothesis that non-standard words are likely to have similar contexts as their canonical equivalents. We define the canonical equivalent here as the most frequent semantically and syntactically equivalent word to the target word. We use this definition since the operation is unsupervised, and for the lack of a standard canonical forms. Variants of this approach have been used in several spelling error correction tasks (Sridhar, 2015). Dasigi and Diab (2011) also use a similar approach to identify variants in DA. We use the Word2vec framework (Mikolov et al., 2013) in the Gensim implementation (Řehůřek and Sojka, 2010) to generate the embedding spaces. We use these embeddings to learn and score normalization candidates based on their cosine distance as a semantic score, and edit distance as a lexical score. In this scope, we first learn a weighted distance function for the individual insertion, deletion, and substitution operations, then use these weights to score the candidates.

Edit Distance Weights The spelling variants are first identified based on narrow window and wide window embeddings, to capture both semantic and syntactic based relationships. For each word in each embedding space we get the nearest *N* neighbors, and intersect them with the *N* nearest neighbors of the word in the other embedding space. We get these neighbors to obtain the weights first, and then use them again for the actual normalization in the next step. We discard candidates that have an edit distance above two, and obtain the individual edit operation weights through their normalized frequencies in the remaining candidates.

Word Mapping We use the learnt edit distance weights to score the normalization candidates mentioned above from the wide and nar-

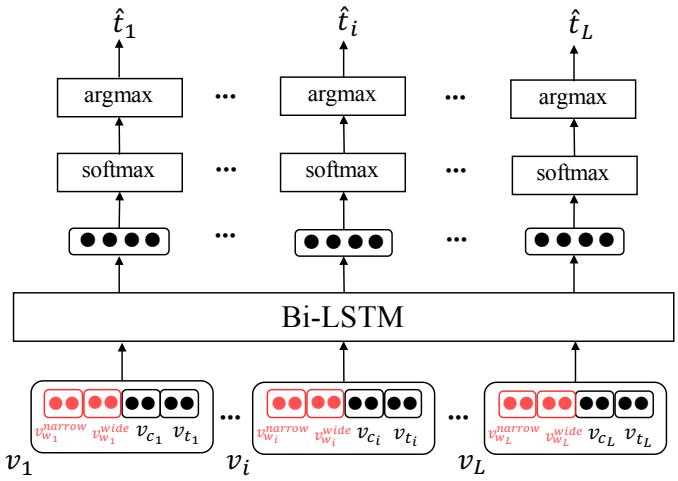


Figure 4: The tagging architecture with the extended (wide and narrow) window embeddings.

row window embedding spaces, and further prune them based on their weighted edit distance. We select the candidate with the highest frequency in the text as the canonical equivalent.

Low Frequency Words Word2vec has a minimum count threshold for the words to be embedded. This value is tunable based on the used corpus. For the words below this threshold Word2vec does not guarantee a good vector representation, and discards them in the embedding model, so we can not use this normalization approach in this case. Instead, we use the weighted edit distances to score and map these words to more frequent cognates, on the character level only.[2]

Normalized Embeddings The pipeline so far results in a more consistent version of the text, which we use to learn the final embeddings upon. These embeddings are used as the pre-trained embeddings in the tagging architecture. This results in normalization at the embedding space level only, where the raw forms are still unmodified. The raw forms can be used for character-level noise reduction later in the tagging pipeline.

5 Experiments

5.1 Data

We use the "ARZ" (Maamouri et al., 2012) manually annotated EGY Arabic corpus, from the Linguistic Data Consortium (LDC), parts 1, 2, 3, 4 and 5. The corpus is based on the POS guidelines used by the LDC for Egyptian Arabic, and consists of about 160K words (excluding numbers and punctuations, 175K overall). The set of analyses for a given raw word includes the correct CODA orthography, in addition to the full morphological and POS annotations.

We use the splits suggested by Diab et al. (2013), comprised of a training set (Train) of about 134K words, a development set (Dev) of 20K words, and a blind testing set (Blind Test) of 21K words. The Dev set is used during the system development to assess design choices. The Blind Test set is used at the end to present the results.

The morphological analyzer we use in this paper is similar to the one used by Habash et al. (2013b). It is based on the SAMA (Graff et al., 2009), CALIMA (Habash et al., 2012b), and ADAM (Salloum and Habash, 2014) databases. EGY content, as in DA in general, contains many MSA cognates. The decision therefore to use all three analyzers was to maximize the recall of the overall analyzer.

We also use an in-house EGY monolingual corpus of about 410 million words, collected from online commentaries of blogs and social media platforms, to pre-train the word embeddings.

To better assess the notions of noise and ambiguity in the EGY dataset, we compare it to the Penn Arabic Treebank (PATB parts 1, 2 and 3) (Maamouri et al., 2004), which is commonly used for morphological disambiguation systems in MSA. MSA is also morphologically rich with high ambiguity levels, so it should provide a suitable reference for EGY. We sample an MSA data of size similar to the EGY dataset size, to be able to

[2] Instead of searching through the entire word space for each word to be normalized, which is computationally expensive, we pruned the search space by only looking at words sharing at least two consonants (in the same order) with it.

draw comparable comparison. Table 2 provides some statistics regarding both datasets. The average number of unique types per lemma (different types mapped to the same lemma encountered in the corpus) is relatively higher for the raw EGY content compared to MSA, at 2.7 vs 2.4. The average for the CODA-based EGY, however, is similar to MSA. This indicates that the normalized version of EGY has a similar sparsity as that for MSA, which is inherently less noisy. The difference in the ratio between raw and CODA EGY is a good indicator of the noise and inconsistency in the EGY dataset.

	EGY Raw	EGY CODA	MSA
Tokens	133,751	133,751	133,763
Inflected types	32,927	30,272	22,022
Lemmas	13,242	13,242	9,522
Avg types/lemma	2.7	2.4	2.4

Table 2: Dataset statistics showing tokens, types, and lemmas count in EGY and an MSA subset. Both from the Train set. The average inflections per lemma is calculated by counting the average unique types that map to the same lemma.

Regarding ambiguity, we calculated the average number of different analyses from the morphological analyzer for a given word in EGY at about 24 analyses per word (about 15 MSA, 6.5 DA, and 2.5 "no-analysis" analyses[3]), whereas for MSA it is around 12. This reflects the severe ambiguity of the EGY dataset compared with MSA in this context. Both noise and ambiguity issues make morphological tagging and disambiguation systems for EGY a very challenging task.

5.2 Experimental Setup

For the Bi-LSTM tagging architecture we use two hidden layers of size 800. Each layer is composed of two LSTM layers for each direction, and a dropout wrapper with keep probability of 0.8, and peephole connections. We use Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.002, and cross-entropy cost function. We use Tensorflow as the development environment.

The LSTM character embedding architecture uses two LSTM layers of size 100, and embedding size 50. The CNN architecture also uses embedding size 50, with filter widths ranging from one to six and max pooling strides of 50.

[3] The morphological analyzer has a backoff mode of "no-analysis" that provides a "proper noun" analysis to all word. The "proper noun" analysis can sometimes be cliticized, so some words might have multiple backoff analyses.

As for the neural language models for lemmatization and diacritization, we use two hidden layers of size 400 for lemmatization, and 600 for diacritization. We also use an input layer of size 300. We use Adam optimizer (Kingma and Ba, 2014) as the optimization algorithm, with learning rate of 0.002. We use TheanoLM (Enarvi and Kurimo, 2016) to develop the models.

The pre-trained word embeddings are of size 250, for both narrow and wide window embeddings. The wide window is set to five, whereas the narrow window is set to two (we experimented with a window of one but it performed slightly lower than a window of two). The number of nearest neighbors in the embedding space mapping experiment is 10 neighbors.

Metrics We use the following evaluation metrics for all systems:

- POS Accuracy (POS): The accuracy over the POS tag set comprised of 36 tags (Habash et al., 2013b).
- Morph Tags Accuracy (Morph Tags): The analysis and disambiguation accuracy over the 14 morphological features we work with, excluding lemmas and diacritized forms.
- Lemmatization Accuracy (Lemma): The accuracy of the lemma form of the words.
- Diacritization Accuracy (Diac): The accuracy of the diacritized form of the words.
- Full Analysis Accuracy (Full): The evaluation accuracy over the entire analysis, including the morphological features, lemma, and diacritized form.

5.3 Results

Table 3 shows the results of all systems for Dev, and Table 4 shows the results for the Blind Test set. We use the MADAMIRA results as the baseline.

Narrow embeddings seem to consistently outperform wide embeddings across all experiments. Regarding character embeddings, using both CNN and LSTM-based character embeddings improve the overall performance for both wide and narrow word embeddings, but LSTMs show consistent improvement over CNNs, which is in line with the conclusions of Heigold et al. (2017).

Embedding extension, through combining the wide and narrow window word embeddings, with

Model	Lemma	Diac	POS	Morph Tags	Full
MADAMIRA EGY (Baseline)	86.4	82.4	91.7	86.7	76.2
Bi-LSTM wide window embeddings	87.3	82.6	92.2	88.0	76.5
+ CNN character embeddings	87.3	82.5	92.6	88.2	76.6
+ LSTM character embeddings	87.4	82.5	92.6	88.3	76.7
+ Embedding space mapping	87.5	82.8	92.6	88.6	76.9
Bi-LSTM narrow window embeddings	87.5	82.9	92.3	88.0	76.7
+ CNN character embeddings	87.5	82.9	92.6	88.6	76.9
+ LSTM character embeddings	87.6	82.9	**92.9**	88.8	77.0
+ Embedding space mapping	87.4	82.8	92.7	88.7	76.9
Bi-LSTM wide+narrow embeddings and LSTM character embeddings	87.6	83.0	92.8	88.8	77.1
+ Embedding space mapping (Best System)	**87.7**	**83.2**	**92.9**	**88.9**	**77.4**
Relative error reduction of best result compared to baseline	9.6%	4.5%	14.5%	16.5 %	5.0%

Table 3: Results of the various systems over the Dev dataset, with MADAMIRA EGY (Pasha et al., 2014) as a state-of-the-art baseline.

Model	Lemma	Diac	POS	Morph Tags	Full
MADAMIRA EGY (Baseline)	87.3	83.3	91.8	86.9	77.3
Bi-LSTM wide window embeddings	87.5	83.1	92.6	87.9	77.4
+ CNN character embeddings	87.7	83.3	92.9	88.1	77.5
+ LSTM character embeddings	87.8	83.3	93.1	88.2	77.6
+ Embedding space mapping	87.8	83.5	93.4	88.9	78.0
Bi-LSTM narrow window embeddings	87.6	83.4	92.7	88.2	77.6
+ CNN character embeddings	87.8	83.6	93.3	88.8	78.0
+ LSTM character embeddings	88.0	83.6	93.5	89.1	78.2
+ Embedding space mapping	87.8	83.6	93.2	88.8	78.1
Bi-LSTM wide+narrow embeddings and LSTM character embeddings	87.9	83.5	93.1	88.6	78.0
+ Embedding space mapping (Best System)	**88.1**	**83.8**	**93.6**	**89.2**	**78.4**
Relative error reduction of best result compared to baseline	6.3%	3.0%	21.9%	17.6 %	4.9%

Table 4: Results of the various systems over the Blind Test dataset.

the LSTM-based character embeddings, significantly enhances the performance beyond the character embeddings alone for the wide embeddings. This is not the case though for narrow window embeddings. This highlights the significance of narrow embeddings for syntactic and morphological modeling, since the extension approach merely adds narrow window embedding capability to the wide window embeddings.

We observe the same pattern for the embedding space mapping approach for noise reduction against the narrow window embeddings. However, combining the extension with the embedding space mapping methods, along with the LSTM-based character embeddings, results in the best performing system. Both approaches seem to complement each other, as the accuracy exceeds any of the methods alone.

The result of the narrow window embeddings is particularly interesting, as it shows that to achieve a relatively good noise-robust morphological disambiguation accuracy, using narrow window embeddings should go a long way. Using more sophisticated, and computationally expensive, noise handling approaches, like embedding extension with embedding space mapping, should achieve even better results.

5.4 System Analysis

Oracle Conventional Orthography Experiment The availability of the manually annotated CODA equivalent of the EGY dataset allows for a deeper analysis of the noise effects on morphological disambiguation. We trained and tested the system using the CODA version of the data, as an oracle experiment of noise-reduced content. CODA-based content is not guaranteed to be noise-free, or be optimal for such syntactic and morphological tasks, but it should provide a good reference in terms of orthography-normalized content.

We train the model on the CODA-EGY training, and test it with the CODA-EGY Dev set. We use the same word pre-training dataset as before. We use LSTM-based character embeddings, and experiment with both wide and narrow embedding window. Table 5 shows the results for the CODA based modeling for Dev. The results are very similar to the best performing model in our earlier ex-

Model	Lemma	Diac	POS	Morph Tags	Full
Bi-LSTM wide window embeddings	87.4	82.5	92.6	88.3	76.7
Bi-LSTM narrow window embeddings	87.6	82.9	**92.9**	88.8	77.0
Bi-LSTM wide+narrow window embeddings+embeddings space mapping	**87.7**	**83.2**	**92.9**	**88.9**	**77.4**
(Oracle Experiment) CODA narrow window embeddings	87.9	83.3	93.0	89.1	77.4
(Oracle Experiment) CODA wide window embeddings	87.7	83.1	92.8	88.8	77.2

Table 5: Results of training and testing the system using the CODA-based Dev data, compared to the results of our system (taken from Table 3). All systems use LSTM-based character embeddings.

periments. These results indicate that our model is very close to the upper performance limit in terms of noise and inconsistency, and achieves noise-robust tagging and disambiguation.

The results for wide and narrow window contexts are also consistent with our earlier experiments, with narrow windowed contexts performing better across all evaluation metrics.

Manual Error Analysis

POS analysis We first analyze the overall error distribution in the POS tagging results. The most common POS error type is mistagging a nominal tag (Noun, Adjective, etc) with a different nominal tag, at 74% of the errors. Nominals include many very frequent tags, such as nouns and adjectives. The next most common error category is mistagging particles with other particles, at around 15%. Mistagging nominals with verbs is at around 4%. Several other low frequency errors cover the remaining 7%. To better understand the nature of the errors we manually checked a sample of 100 POS tagging errors. Almost 48% of them are gold errors, out of which our system gets 74% correct.

Lemma analysis We also manually checked a sample of 100 lemmatization errors. We observe that 30% of them are gold errors, 23% are the result of a wrong POS tag, 15% are acceptable MSA lemmas, 12% are due to minor and normally acceptable spelling issues, mainly the Hamza letter (glottal stops), and 6% are due to inconsistent diacritization. The MSA-related errors are due to the many MSA cognates in DA content. So providing an MSA-based analysis instead of an equivalent DA analysis can be acceptable for the purpose of this analysis. Hamza spelling variations, especially at the beginning of the word, are common in both DA and MSA written content.

Diacritization analysis We checked a sample of 100 diacritization errors. We observed more errors attributed to error propagation, as wrong POS tags and lemmas lead to many diacritization errors. The percentage of gold errors is only 17%, whereas MSA-cognate related errors are about 32%, POS related errors cover 13%, Hamza errors 11%, lemmatization errors include 7%, and the rest are mostly due to wrong case, gender, person tags, and other unidentified issues.

6 Conclusion and Future Work

We presented several neural morphological disambiguation models for EGY, and used several approaches for noise-robust processing. Our system outperforms a state-of-the-art system for EGY. We observed that character embeddings, combined with pre-trained word embeddings, provide a significant performance boost over the baseline. We showed that LSTM-based character embeddings outperform CNN-based models for EGY. We also showed that narrow window embeddings significantly outperform wide window embeddings for tagging. We also experimented with a normalization model on the word-level vectors, mapping non-canonical words to canonical neighbors through embedding space mapping. The results showed an additional improvement over the narrow window embeddings.

Future directions include exploring additional deep learning architectures for morphological modeling and disambiguation, especially joint and multitasking architectures. We also plan to explore knowledge transfer and adaptation models for more dialects with limited resources.

Acknowledgment The first author was supported by the New York University Abu Dhabi Global PhD Student Fellowship program. The support and resources from the High Performance Computing Center at New York University Abu Dhabi are also gratefully acknowledged.

References

Ahmed Abdelali, Kareem Darwish, Nadir Durrani, and Hamdy Mubarak. 2016. Farasa: A fast and furious segmenter for Arabic. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*. San Diego, California, pages 11–16. http://www.aclweb.org/anthology/N16-3003.

Gustavo Aguilar, Suraj Maharjan, Adrian Pastor López Monroy, and Thamar Solorio. 2017. A multi-task approach for named entity recognition in social media data. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*. pages 148–153.

Rania Al-Sabbagh and Roxana Girju. 2010. Mining the Web for the Induction of a Dialectical Arabic Lexicon. In *LREC*. Valetta, Malta.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics* 18(4):467–479.

Pradeep Dasigi and Mona Diab. 2011. Codact: Towards identifying orthographic variants in dialectal arabic. In *Proceedings of the International Joint Conference on Natural Language Processing*. Chiang Mai, Thailand.

Leon Derczynski, Alan Ritter, Sam Clark, and Kalina Bontcheva. 2013. Twitter part-of-speech tagging for all: Overcoming sparse and noisy data. In *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*. INCOMA Ltd. Shoumen, BULGARIA, pages 198–206. http://www.aclweb.org/anthology/R13-1026.

Mona Diab, Nizar Habash, Owen Rambow, and Ryan Roth. 2013. LDC Arabic treebanks and associated corpora: Data divisions manual. *arXiv preprint arXiv:1309.5652* .

Seppo Enarvi and Mikko Kurimo. 2016. Theanolm - an extensible toolkit for neural network language modeling. *CoRR* abs/1605.00942. http://arxiv.org/abs/1605.00942.

Ramy Eskander, Nizar Habash, Owen Rambow, and Nadi Tomeh. 2013. Processing Spontaneous Orthography. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. Atlanta, GA.

Jennifer Foster, Özlem Çetinoğlu, Joachim Wagner, Joseph Le Roux, Stephen Hogan, Joakim Nivre, Deirdre Hogan, and Josef Van Genabith. 2011. #hardtoparse: Pos tagging and parsing the twitterverse. In *Proceedings of the 5th AAAI Conference on Analyzing Microtext*. AAAI Press, AAAIWS'11-05, pages 20–25. http://dl.acm.org/citation.cfm?id=2908630.2908634.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *ACL-HLT '11: Short Papers - Volume 2*. Association for Computational Linguistics, Stroudsburg, PA, USA, HLT '11, pages 42–47. http://dl.acm.org/citation.cfm?id=2002736.2002747.

Yoav Goldberg. 2016. A primer on neural network models for natural language processing. *J. Artif. Intell. Res.(JAIR)* 57:345–420.

David Graff, Mohamed Maamouri, Basma Bouziri, Sondos Krouna, Seth Kulick, and Tim Buckwalter. 2009. Standard Arabic Morphological Analyzer (SAMA) Version 3.1. Linguistic Data Consortium LDC2009E73.

Nizar Habash, Mona Diab, and Owen Rambow. 2012a. Conventional Orthography for Dialectal Arabic: Principles and Guidelines – Egyptian Arabic. Technical Report CCLS-12-02, Columbia University Center for Computational Learning Systems.

Nizar Habash, Fadhl Eryani, Salam Khalifa, Owen Rambow, Dana Abdulrahim, Alexander Erdmann, Reem Faraj, Wajdi Zaghouani, Houda Bouamor, Nasser Zalmout, Sara Hassan, Faisal Al-Shargi, Sakhar Alkhereyf, Basma Abdulkareem, Ramy Eskander, Mohammad Salameh, and Hind Saddiki. 2018. Unified guidelines and resources for arabic dialect orthography. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*.

Nizar Habash, Ramy Eskander, and Abdelati Hawwari. 2012b. A Morphological Analyzer for Egyptian Arabic. In *Proceedings of the Twelfth Meeting of the Special Interest Group on Computational Morphology and Phonology*. Montréal, Canada, pages 1–9.

Nizar Habash and Owen Rambow. 2005. Arabic Tokenization, Part-of-Speech Tagging and Morphological Disambiguation in One Fell Swoop. In *Proceedings of the 43rd Annual Meeting of the ACL*. Ann Arbor, Michigan, pages 573–580.

Nizar Habash and Owen Rambow. 2007. Arabic Diacritization through Full Morphological Tagging. In *Proceedings of the 8th Meeting of the North American Chapter of the Association for Computational Linguistics/Human Language Technologies Conference (HLT-NAACL07)*.

Nizar Habash, Owen Rambow, and Ryan Roth. 2009. MADA+TOKAN: A toolkit for Arabic tokenization, diacritization, morphological disambiguation, POS tagging, stemming and lemmatization. In Khalid Choukri and Bente Maegaard, editors, *Proceedings of the Second International Conference on Arabic Language Resources and Tools*. The MEDAR Consortium.

Nizar Habash, Ryan Roth, Owen Rambow, Ramy Eskander, and Nadi Tomeh. 2013a. Morphological Analysis and Disambiguation for Dialectal Arabic. In *Proceedings of the 2013 Conference of NAACL-HLT*. Atlanta, GA.

Nizar Habash, Ryan Roth, Owen Rambow, Ramy Eskander, and Nadi Tomeh. 2013b. Morphological

Analysis and Disambiguation for Dialectal Arabic. In *Proceedings of NAACL-HLT*. Atlanta, Georgia, pages 426–432. http://www.aclweb.org/anthology/N13-1044.

Nizar Habash, Abdelhadi Soudi, and Tim Buckwalter. 2007. On Arabic Transliteration. In A. van den Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*, Springer.

Georg Heigold, Guenter Neumann, and Josef van Genabith. 2017. An extensive empirical evaluation of character-based morphological tagging for 14 languages. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, pages 505–513. http://www.aclweb.org/anthology/E17-1048.

Go Inoue, Hiroyuki Shindo, and Yuji Matsumoto. 2017. Joint prediction of morphosyntactic categories for fine-grained arabic part-of-speech tagging exploiting tag dictionary information. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. pages 421–431.

Mustafa Jarrar, Nizar Habash, Faeq Alrimawi, Diyam Akra, and Nasser Zalmout. 2016. Curras: an annotated corpus for the Palestinian Arabic dialect. *Language Resources and Evaluation* pages 1–31. https://doi.org/10.1007/s10579-016-9370-7.

Anna Jørgensen, Dirk Hovy, and Anders Søgaard. 2015. Challenges of studying and processing dialects in social media. In *Proceedings of the Workshop on Noisy User-generated Text*. pages 9–18.

Salam Khalifa, Nizar Habash, Dana Abdulrahim, and Sara Hassan. 2016a. A Large Scale Corpus of Gulf Arabic. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*. Portorož, Slovenia.

Salam Khalifa, Nasser Zalmout, and Nizar Habash. 2016b. Yamama: Yet another multi-dialect arabic morphological analyzer. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*. pages 223–227.

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

Xiaohua Liu, Shaodian Zhang, Furu Wei, and Ming Zhou. 2011. Recognizing named entities in tweets. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, HLT '11, pages 359–367. http://dl.acm.org/citation.cfm?id=2002472.2002519.

Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. 2004. The Penn Arabic Treebank: Building a Large-Scale Annotated Arabic Corpus. In *NEMLAR Conference on Arabic Language Resources and Tools*. Cairo, Egypt, pages 102–109.

Mohamed Maamouri, Sondos Krouna, Dalila Tabessi, Nadia Hamrouni, and Nizar Habash. 2012. Egyptian Arabic Morphological Annotation Guidelines.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., pages 3111–3119.

Emad Mohamed, Behrang Mohit, and Kemal Oflazer. 2012. Annotating and Learning Morphological Segmentation of Egyptian Colloquial Arabic. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*. Istanbul.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 380–390. http://www.aclweb.org/anthology/N13-1039.

Arfath Pasha, Mohamed Al-Badrashiny, Ahmed El Kholy, Ramy Eskander, Mona Diab, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. 2014. MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic. In *In Proceedings of LREC*. Reykjavik, Iceland.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. In *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*. volume 59.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 412–418. https://doi.org/10.18653/v1/P16-2067.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, Valletta, Malta, pages 45–50. http://is.muni.cz/publication/884893/en.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: An experimental study. In *Proceedings of the Conference on Empirical Methods in Natural*

Language Processing. Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP '11, pages 1524–1534. http://dl.acm.org/citation.cfm?id=2145432.2145595.

Ryan Roth, Owen Rambow, Nizar Habash, Mona Diab, and Cynthia Rudin. 2008. Arabic morphological tagging, diacritization, and lemmatization using lexeme models and feature ranking. In *ACL 2008: The Conference of the Association for Computational Linguistics; Companion Volume, Short Papers*. Association for Computational Linguistics, Columbus, Ohio.

Houda Saadane and Nizar Habash. 2015. A Conventional Orthography for Algerian Arabic. In *ANLP Workshop 2015*. page 69.

Wael Salloum and Nizar Habash. 2014. ADAM: Analyzer for Dialectal Arabic Morphology. *Journal of King Saud University-Computer and Information Sciences* 26(4):372–378.

Vivek Kumar Rangarajan Sridhar. 2015. Unsupervised text normalization using distributed representations of words and phrases. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*. pages 8–16.

Andrew Trask, David Gilmore, and Matthew Russell. 2015. Modeling order in neural word embeddings at scale. *arXiv preprint arXiv:1506.02338* .

Lifu Tu, Kevin Gimpel, and Karen Livescu. 2017. Learning to embed words in context for syntactic tasks. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*. pages 265–275.

Houcemeddine Turki, Emad Adel, Tariq Daouda, and Nassim Regragui. 2016. A Conventional Orthography for Maghrebi Arabic. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*. Portorož, Slovenia.

Rob van der Goot, Barbara Plank, and Malvina Nissim. 2017. To normalize, or not to normalize: The impact of normalization on part-of-speech tagging. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*. Association for Computational Linguistics, Copenhagen, Denmark, pages 31–39. http://www.aclweb.org/anthology/W17-4404.

Nasser Zalmout and Nizar Habash. 2017. Don't throw those morphological analyzers away just yet: Neural morphological disambiguation for arabic. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 704–713. https://www.aclweb.org/anthology/D17-1073.

I. Zribi, R. Boujelbane, A. Masmoudi, M. Ellouze Khmekhem, L. Hadrich Belguith, and N. Habash. 2014. A Conventional Orthography for Tunisian Arabic. In *In Proceedings of LREC*. Reykjavik, Iceland.

Parsing Tweets into Universal Dependencies

Yijia Liu
Harbin Institute of Technology
yjliu@ir.hit.edu.cn

Yi Zhu
University of Cambridge
yz568@cam.ac.uk

Wanxiang Che **Bing Qin**
Harbin Institute of Technology

Nathan Schneider
Georgetown University

Noah A. Smith
University of Washington

Abstract

We study the problem of analyzing tweets with Universal Dependencies (UD; Nivre et al., 2016). We extend the UD guidelines to cover special constructions in tweets that affect tokenization, part-of-speech tagging, and labeled dependencies. Using the extended guidelines, we create a new tweet treebank for English (TWEEBANK V2) that is four times larger than the (unlabeled) TWEEBANK V1 introduced by Kong et al. (2014). We characterize the disagreements between our annotators and show that it is challenging to deliver consistent annotation due to ambiguity in understanding and explaining tweets. Nonetheless, using the new treebank, we build a pipeline system to parse raw tweets into UD. To overcome annotation noise without sacrificing computational efficiency, we propose a new method to distill an ensemble of 20 transition-based parsers into a single one. Our parser achieves an improvement of 2.2 in LAS over the un-ensembled baseline and outperforms parsers that are state-of-the-art on other treebanks in both accuracy and speed.

1 Introduction

NLP for social media messages is challenging, requiring domain adaptation and annotated datasets (e.g., treebanks) for training and evaluation. Pioneering work by Foster et al. (2011) annotated 7,630 tokens' worth of tweets according to the phrase-structure conventions of the Penn Treebank (PTB; Marcus et al., 1993), enabling conversion to Stanford Dependencies. Kong et al. (2014) further studied the challenges in annotating tweets and presented a tweet treebank (TWEEBANK), consisting of 12,149 tokens and largely following conventions suggested by Schneider et al. (2013), fairly close to Yamada and Matsumoto (2003) dependencies (without labels). Both annotation efforts were highly influenced by the PTB, whose guidelines have good grammatical coverage on newswire. However, when it comes to informal, unedited, user-generated text, the guidelines may leave many annotation decisions unspecified.

Universal Dependencies (Nivre et al., 2016, UD) were introduced to enable consistent annotation across different languages. To allow such consistency, UD was designed to be adaptable to different genres (Wang et al., 2017) and languages (Guo et al., 2015; Ammar et al., 2016). We propose that analyzing the syntax of tweets can benefit from such adaptability. In this paper, we introduce a new English tweet treebank of 55,607 tokens that follows the UD guidelines, but also contends with social media-specific challenges that were not covered by UD guidelines.[1] Our annotation includes tokenization, part-of-speech (POS) tags, and (labeled) Universal Dependencies. We characterize the disagreements among our annotators and find that consistent annotation is still challenging to deliver even with the extended guidelines.

Based on these annotations, we nonetheless designed a pipeline to parse raw tweets into Universal Dependencies. Our pipeline includes: a bidirectional LSTM (bi-LSTM) tokenizer, a word cluster–enhanced POS tagger (following Owoputi et al., 2013), and a stack LSTM parser with character-based word representations (Ballesteros et al., 2015), which we refer to as our "baseline" parser. To overcome the noise in our annotated

[1] We developed our treebank independently of a similar effort for Italian tweets (Sanguinetti et al., 2017). See §2.5 for a comparison.

Proceedings of NAACL-HLT 2018, pages 965–975
New Orleans, Louisiana, June 1 - 6, 2018.

data and achieve better performance without sacrificing computational efficiency, we distill a 20-parser ensemble into a single greedy parser (Hinton et al., 2015). We show further that learning directly from the exploration of the ensemble parser is more beneficial than learning from the gold standard "oracle" transition sequence. Experimental results show that an improvement of more than 2.2 points in LAS over the baseline parser can be achieved with our distillation method. It outperforms other state-of-the-art parsers in both accuracy and speed.

The contributions of this paper include:

- We study the challenges of annotating tweets in UD (§2) and create a new tweet treebank (TWEEBANK V2), which includes tokenization, part-of-speech tagging, and labeled Universal Dependencies. We also characterize the difficulties of creating such annotation.
- We introduce and evaluate a pipeline system to parse the raw tweet text into Universal Dependencies (§3). Experimental results show that it performs better than a pipeline of the state-of-the-art alternatives.
- We propose a new distillation method for training a greedy parser, leading to better performance than existing methods and without efficiency sacrifices.

Our dataset and system are publicly available at `https://github.com/Oneplus/Tweebank` and `https://github.com/Oneplus/twpipe`.

2 Annotation

We first review TWEEBANK V1 of Kong et al. (2014), the previous largest Twitter dependency annotation effort (§2.1). Then we introduce the differences in our tokenization (§2.2) and part-of-speech (§2.3) (re)annotation with O'Connor et al. (2010) and Gimpel et al. (2011), respectively, on which TWEEBANK V1 was built. We describe our effort of adapting the UD conventions to cover tweet-specific constructions (§2.4). Finally, we present our process of creating a new tweet treebank, TWEEBANK V2, and characterize the difficulties in reaching consistent annotations (§2.6).

2.1 Background: TWEEBANK

The annotation effort we describe stands in contrast to the previous work by Kong et al. (2014). Their aim was the rapid development of a dependency parser for tweets, and to that end they contributed a new annotated corpus, TWEEBANK, consisting of 12,149 tokens. Their annotations added unlabeled dependencies to a portion of the data annotated with POS tags by Gimpel et al. (2011) and Owoputi et al. (2013) after rule-based tokenization (O'Connor et al., 2010). Kong et al. also contributed a system for parsing; we defer the discussion of their parser to §3.

Kong et al.'s rapid, small-scale annotation effort was heavily constrained. It was carried out by annotators with only cursory training, no clear annotation guidelines, and no effort to achieve consensus on controversial cases. Annotators were allowed to underspecify their analyses. Most of the work was done in a very short amount of time (a day). Driven both by the style of the text they sought to annotate and by exigency, some of their annotation conventions included:

- Allowing an annotator to exclude tokens from the dependency tree. A clear criterion for exclusion was not given, but many tokens were excluded because they were deemed "non-syntactic."
- Allowing an annotator to merge a multiword expression into a single node in the dependency tree, with no internal structure. Annotators were allowed to take the same step with noun phrases.
- Allowing multiple roots, since a single tweet might contain more than one sentence.

These conventions were justified on the grounds of making the annotation easier for non-experts, but they must be revisited in our effort to apply UD to tweets.

2.2 Tokenization

Our tokenization strategy lies between the strategy of O'Connor et al. (2010) and that of UD. There is a tradeoff between preservation of original tweet content and respecting the UD guidelines.

The regex-based tokenizer of O'Connor et al. (2010)—which was originally designed for an exploratory search interface called TweetMotif, not for NLP—preserves most whitespace-delimited tokens, including hashtags, at-mentions, emoticons, and unicode glyphs. They also treat contractions and acronyms as whole tokens and do not

split them. UD tokenization,[2] in order to better serve dependency annotation, treats each syntactic word as a token. They therefore more aggressively split clitics from contractions (e.g., *gonna* is tokenized as *gon* and *na*; *its* is tokenized as *it* and *s* when *s* is a copula). But acronyms are not touched in the UD tokenization guidelines. Thus, we follow the UD tokenization for contractions and leave acronyms like *idc* ("I don't care") as a single token.

In the different direction of splitting tokens, UD guidelines also suggest to merge *multi-token words* (e.g., *20 000*) into one single token in some special cases. We witnessed a small number of tweets that contain multi-token words (e.g., *Y O*, and *R E T W E E T*) but didn't combine them for simplicity. Such tokens only account for 0.07% and we use the UD *goeswith* relation to resolve these cases in the dependency annotations.

2.3 Part-of-Speech Annotation

Before turning to UD annotations, we (re)annotated the data with POS tags, for consistency with other UD efforts, which adopt the universal POS tagset.[3] In some cases, non-corresponding tag conflicts arose between the UD English Web Treebank treebank conventions (UD_English-EWT; de Marneffe et al., 2014)[4] and the conventions of Gimpel et al. (2011). In these cases, we always conformed to UD, enabling consistency (e.g., when we exploit the existing UD_English-EWT treebank in our parser for tweets, §3). For example, the nominal URL in Figure 2 is tagged as *other* (`X`) and + is tagged as *symbol* (`SYM`) rather than *conjunction* (`CCONJ`).

Tokens that do not have a syntactic function (see Figure 1, discussed at greater length in the next section) were usually annotated as *other* (`X`), except for emoticons, which are tagged as *symbol* (`SYM`), following UD_English-EWT.

Tokens that abbreviate multiple words (such as *idc*) are resolved to the POS of the syntactic head of the expression, following UD conventions (in this example, the head *care* is a verb, so *idc* is tagged as a verb). When the token is not phrasal, we use the POS of the left-most sub-phrase. For example, *mfw* ("my face when") is tagged as a noun (for *face*).

Compared to the effort of Gimpel et al. (2011), our approach simplifies some matters. For example, if a token is not considered syntactic by UD conventions, it gets an *other* (`X`) tag (Gimpel et al. had more extensive conventions). Other phenomena, like abbreviations, are more complicated for us, as discussed above; Gimpel et al. used a single part of speech for such expressions.

Another important difference follows from the difference in tokenization. As discussed in §2.2, UD calls for more aggressive tokenization than that of O'Connor et al. (2010) which opted out of splitting contractions and possessives. As a consequence of adopting O'Connor et al.'s (2010) tokenization, Gimpel et al. introduced new parts of speech for these cases instead.[5] For us, these tokens must be split, but universal parts of speech can be applied.

2.4 Universal Dependencies Applied to Tweets

We adopt UD version 2 guidelines to annotate the syntax of tweets. In applying UD annotation conventions to tweets, the choices of Kong et al. (2014) must be revisited. We consider the key questions that arose in our annotation effort, and how we resolved them.

Acronym abbreviations. We follow Kong et al. (2014) and annotate the syntax of an acronym as a single word without normalization. Their syntactic functions are decided according to their context. Eisenstein (2013) studied the necessity of normalization in social media text and argued that such normalization is problematic. Our solution to the syntax of abbreviations follows the spirit of his argument. Because abbreviations which clearly carry syntactic functions only constitute 0.06% of the tokens in our dataset, we believe that normalization for acronyms is an unnecessarily complicated step.

Non-syntactic tokens. The major characteristic that distinguishes tweets from standard texts is that a large proportion of tokens don't carry any syntactic function. In our annotation, there are five types of non-syntactic tokens commonly seen in tweets: sentiment emoticons, retweet markers and

[2] http://universaldependencies.org/u/overview/tokenization.html

[3] A revised and extended version of Petrov et al. (2012) with 17 tags.

[4] https://github.com/UniversalDependencies/UD_English-EWT

[5] These tags only account for 2.7% of tokens, leading to concerns about data sparseness in tagging and all downstream analyses.

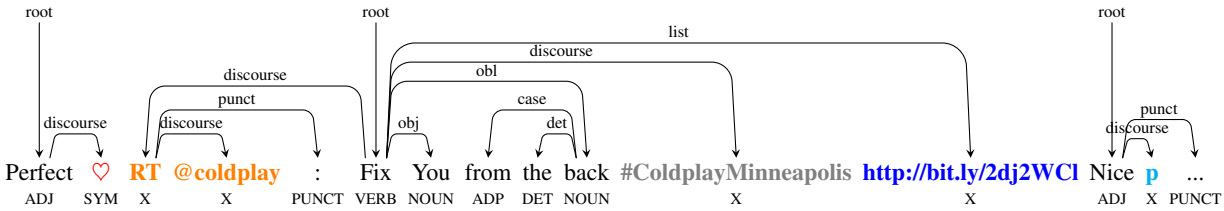


Figure 1: An example to illustrate non-syntactic tokens: sentiment emoticon, retweet marker and its following at-mention, topical hashtag, referential URL, and truncated word. This is a concatenation of three real tweets.

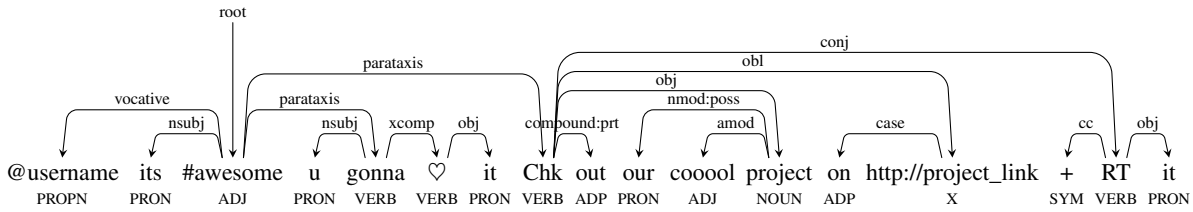


Figure 2: An example to illustrate informal but syntactic tokens. This is a contrived example inspired by several tweets.

	syntactic (%)	non-syntactic (%)
emoticons	0.25	0.95
RT	0.14	2.49
hashtag	1.02	1.24
URL	0.67	2.38
truncated words	0.00	0.49
total	2.08	7.55

Table 1: Proportions of non-syntactic tokens in our annotation. These statistics are obtained on 140 character–limited tweets.

their following at-mentions, topical hashtags, referential URLs, and truncated words.[6] Figure 1 illustrates examples of these non-syntactic tokens. As discussed above, these are generally tagged with the *other* (X) part of speech, except emoticons, which are tagged as *symbol* (SYM). In our annotation, 7.55% of all tokens are belong to one of the five types; detailed statistics can be found in Table 1.

It is important to note that these types may, in some contexts, have syntactic functions. For example, besides being a discourse marker, *RT* can abbreviate the verb *retweet*; emoticons and hashtags may be used as content words within a sentence; and at-mentions can be normal vocative proper nouns: see Figure 2. Therefore, the criteria for annotating a token as non-syntactic must be context-dependent.

Inspired by the way UD deals with *punctuation* (which is canonically non-syntactic), we adopt the following conventions:

- If a non-syntactic token is within a sentence that has a clear predicate, it will be attached to this predicate. The retweet construction is a special case and we will discuss its treatment in the following paragraph.
- If the whole sentence is a sequence of non-syntactic tokens, we attach all these tokens to the first one.
- Non-syntactic tokens are mostly labeled as *discourse*, but URLs are always labeled as *list*, following the UD_English-EWT dataset.

Kong et al. (2014) proposed an additional preprocessing step, *token selection*, in their annotation process. They required the annotators to first select the non-syntactic tokens and exclude them from the final dependency annotation. In order to keep our annotation conventions in line with UD norms and preserve the original tweets as much as possible, we include non-syntactic tokens in our annotation following the conventions above. Compared with Kong et al. (2014), we also gave a clear definition of non-syntactic tokens, which helped us avoid confusion during annotation.

[6]The tweets we analyze have at most 140 characters. Although Twitter has doubled the tweet length limit to 280 characters since our analysis, we believe this type of token will still remain.

Retweet construction. Figure 1 shows an example of the retweet construction (*RT @coldplay :*). This might be treated as a verb phrase, with *RT* as a verb and the at-mention as an argument. This solution would lead to an uninformative root word and, since this expression is idiomatic to Twitter, might create unnecessary confusion for downstream applications aiming to identify the main predicate(s) of a tweet. We therefore treat the whole expression as non-syntactic, including assigning the *other* (X) part of speech to both *RT* and *@coldplay*, attaching the at-mention to *RT* with the *discourse* label and the colon to *RT* with the *punct*(uation) label, and attaching *RT* to the predicate of the following sentence.

Constructions handled by UD. A number of constructions that are especially common in tweets are handled by UD conventions: ellipsis, irregular word orders, and paratactic phrases and sentences not explicitly delineated by punctuation.

Vocative at-mentions. Another idiomatic construction on Twitter is a vocative at-mention (sometimes a signal that a tweet is a reply to a tweet by the mentioned user). We treat these at-mentions as vocative expressions, labeling them with POS tag *proper noun* (PROPN) and attaching them to the main predicate of the sentence it is within with the label *vocative* as in UD guidelines (see Figure 2 for an example).

2.5 Comparison to PoSTWITA-UD

The first Twitter treebank annotated with Universal Dependencies was the *PosTWITA-UD* corpus for Italian (Sanguinetti et al., 2017), which consists of 6,738 tweets (119,726 tokens). In their convention, tokenization tends to preserve the original tweet content but two special cases, *articulated prepositions* (e.g., *nella* as *in la*) and *clitic clusters* (e.g. *guardandosi* as *guardando si*), are tokenized. Their lemmas include spelling normalization, whereas our lemmas only normalize casing and inflectional morphology. The current UD guidelines on lemmas are flexible, so variation between treebanks is expected.[7]

With respect to tweet-specific constructions, Sanguinetti et al.'s (2017) and our interpretations of headedness are the same, but we differ in the relation label. For topical hashtags, we use *discourse* while they used *parataxis*. In referential URLs, we use *list* (following the precedent of UD_English-EWT) while they used *dep*. Our choice of *discourse* for sentiment emoticons is inspired by the observation that emoticons are annotated as *discourse* by UD_English-EWT; Sanguinetti et al. (2017) used the same relation for the emoticons. Retweet constructions and truncated words were not explicitly touched by Sanguinetti et al. (2017). Judging from the released treebank[8], the *RT* marker, at-mention, and colon in the retweet construction are all attached to the predicate of the following sentence with *dep*, *vocative:mention* and *punct*. We expect that the official UD guidelines will eventually adopt standards for these constructions so the treebanks can be harmonized.

2.6 TWEEBANK V2

Following the guidelines presented above, we create a new Twitter dependency treebank, which we call TWEEBANK V2.

2.6.1 Data Collection

TWEEBANK V2 is built on the original data of TWEEBANK V1 (840 unique tweets, 639/201 for training/test), along with an additional 210 tweets sampled from the POS-tagged dataset of Gimpel et al. (2011) and 2,500 tweets sampled from the Twitter stream from February 2016 to July 2016.[9] The latter data source consists of 147.4M English tweets after being filtered by the *lang* attribute in the tweet JSON and *langid.py*.[10] As done by Kong et al. (2014), the annotation unit is always the tweet in its entirety—which may consist of multiple sentences—not the sentence alone. Before annotation, we use a simple regular expression to anonymize usernames and URLs.

2.6.2 Annotation Process

Our annotation process was conducted in two stages. In the first stage, 18 researchers worked on the TWEEBANK V1 portion and the additional 210 tweets and created the initial annotations in one day. Before annotating, they were given a tutorial overview of the general UD annotation conventions and our guidelines specifically for annotating tweets. Both the guidelines and annotations

[7] http://universaldependencies.org/u/overview/morphology.html#lemmas

[8] https://github.com/UniversalDependencies/UD_Italian-PoSTWITA

[9] Data downloaded from https://archive.org/.

[10] https://github.com/saffsd/langid.py

split	TWEEBANK V1 tweets	TWEEBANK V1 tokens	TWEEBANK V2 tweets	TWEEBANK V2 tokens
train	639	9,310	1,639	24,753
dev.	–	–	710	11,742
test	201	2,839	1,201	19,112
total	840	12,149	3,550	55,607

Table 2: Statistics of TWEEBANK V2 and comparison with TWEEBANK V1.

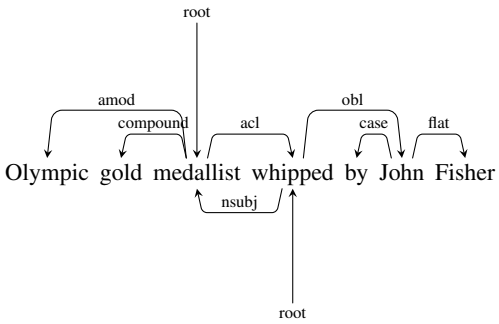


Figure 3: An example of disagreement; one annotator's parse is shown above, disagreeing arcs from the other annotator are shown below. This is a real example in our annotation.

were further refined by the authors of this paper to increase the coverage of our guidelines and solve inconsistencies between different annotators during this exercise. In the second stage, a tokenizer, a POS tagger, and a parser were trained on the annotated data from the first stage (1,050 tweets in total), and used to automatically analyze the sampled 2,500 tweets. Authors of this paper manually corrected the parsed data and finally achieved 3,550 labeled tweets.[11] Newly created annotations are split into train, development, and test sets and appended to the original splits of TWEEBANK V1. Statistics of our annotations and data splits are shown in Table 2.

We report the inter-annotator agreement between the annotators in the second stage. There is very little disagreement on the tokenization annotation. The agreement rate is 96.6% on POS, 88.8% on unlabeled dependencies, and 84.3% on labeled dependencies. Further analysis shows the major disagreements on POS involve entity names (30.6%) and topical hashtags (18.1%). Taking the example in Figure 1, "Fix you" can be understood as a verbal phrase but also as the name of the Coldplay's single and tagged as proper noun. An example of a disagreement on dependencies is shown in Figure 3. Depending on whether this is an example of a zero copula construction, or a clause-modified noun, either annotation is plausible.

[11] Manual annotation was done with Arborator (Gerdes, 2013), a web platform for drawing dependency trees.

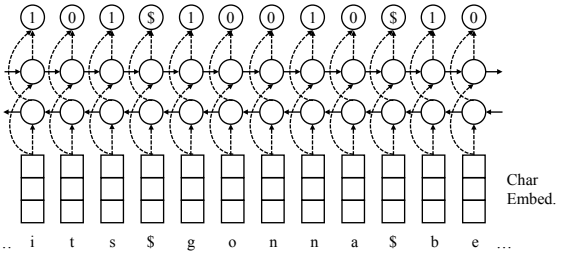


Figure 4: The bi-LSTM tokenizer that segments '*its gonna be*' into '*it s gon na be*'.

3 Parsing Pipeline

We present a pipeline system to parse tweets into Universal Dependencies. We evaluate each component individually, and the system as a whole.

3.1 Tokenizer

Tokenization, as the initial step of many NLP tasks, is non-trivial for informal tweets, which include hashtags, at-mentions, and emoticons (O'Connor et al., 2010). Context is often required for tokenization decisions; for example, the asterisk in *4*3* is a separate token signifying multiplication, but the asterisk in *sh*t* works as a mask to evoke censorship and should not be segmented.

We introduce a new character-level bidirectional LSTM (bi-LSTM) sequence-labeling model (Huang et al., 2015; Ma and Hovy, 2016) for tokenization. Our model takes the raw sentence and tags each character in this sentence as whether it is the beginning of a word (1 as the beginning and 0 otherwise). Figure 4 shows the architecture of our tokenization model. Space is treated as an input but deterministically assigned a special tag $.

Experimental results. Our preliminary results showed that our model trained on the combination of UD_English-EWT and TWEEBANK V2 outperformed the one trained only on the UD_English-EWT or TWEEBANK V2, consistent with previous work on dialect treebank parsing (Wang et al., 2017). So we trained our tokenizer on the training portion of TWEEBANK V2 combined with the UD_English-EWT training set and tested on the TWEEBANK V2 test set. We report F_1 scores, combining precision and recall for token identification. Table 3 shows the tokenization results,

System	F_1
Stanford CoreNLP	97.3
Twokenizer	94.6
UDPipe v1.2	97.4
our bi-LSTM tokenizer	98.3

Table 3: Tokenizer comparison on the TWEEBANK V2 test set.

System	Accuracy
Stanford CoreNLP	90.6
Owoputi et al., 2013 (greedy)	93.7
Owoputi et al., 2013 (CRF)	94.6
Ma and Hovy, 2016	92.5

Table 4: POS tagger comparison on gold-standard tokens in the TWEEBANK V2 test set.

Tokenization System	F_1
Stanford CoreNLP	92.3
our bi-LSTM tokenizer (§3.1)	93.3

Table 5: Owoputi et al. (2013) POS tagging performance with automatic tokenization on the TWEEBANK V2 test set.

compared to other available tokenizers. Stanford CoreNLP (Manning et al., 2014) and Twokenizer (O'Connor et al., 2010)[12] are rule-based systems and were not adapted to the UD tokenization scheme. The UDPipe v1.2 (Straka and Straková, 2017) model was re-trained on the same data as our system. Compared with UDPipe, we use an LSTM instead of a GRU in our model and we also use a larger size for hidden units (64 vs. 20), which has stronger representational power. Our bi-LSTM tokenizer achieves the best accuracy among all these tokenizers. These results speak to the value of statistical modeling in tokenization for informal texts.

3.2 Part-of-Speech Tagger

Part-of-speech tagging for tweets has been extensively studied (Ritter et al., 2011; Gimpel et al., 2011; Derczynski et al., 2013; Owoputi et al., 2013; Gui et al., 2017). We therefore consider existing POS taggers for tweets instead of developing our own. On the annotation scheme designed in §2.3, based on UD and adapted for Twitter, we compared several existing systems: the Stanford CoreNLP tagger, Owoputi et al.'s (2013) word cluster–enhanced tagger (both greedy and CRF variants), and Ma and Hovy's (2016) neural network tagger which achieves the state-of-the-art performance on PTB. Gui et al. (2017) presented a state-of-the-art neural tagger for Twitter, but their implementation works only with the PTB tagset, so we exclude it. All compared systems were re-trained on the combination of the UD_English-EWT and TWEEBANK V2 training sets. We use Twitter-specific GloVe embeddings released by Pennington et al. (2014) in all neural taggers and parsers.[13]

Experimental results. We tested the POS taggers on the TWEEBANK V2 test set. Results with gold-standard tokenization are shown in Table 4. Careful feature engineering and Brown et al. (1992) clusters help Owoputi et al.'s (2013) feature-based POS taggers to outperform Ma and Hovy's (2016) neural network model.

Results of the Owoputi et al. (2013) tagger with non-greedy inference on automatically tokenized data are shown in Table 5. We see that errors in tokenization do propagate, but tagging performance is above 93% with our tokenizer.

3.3 Parser

Social media applications typically require processing large volumes of data, making speed an important consideration. We therefore begin with the neural greedy stack LSTM parser introduced by Ballesteros et al. (2015), which can parse a sentence in linear time and harnesses character representations to construct word vectors, which should help mitigate the challenge of spelling variation. We encourage the reader to refer their paper for more details about the model.

In our initial experiments, we train our parser on the combination of UD_English-EWT and TWEEBANK V2 training sets. Gold-standard tokenization and automatic POS tags are used. Automatic POS tags are assigned with 5-fold jackknifing. Hyperparameters are tuned on the TWEEBANK V2 development set. Unlabeled attachment score and labeled attachment score (including punctuation) are reported. All the experiments were run on a Xeon E5-2670 2.6 GHz machine.

Reimers and Gurevych (2017) and others have

[12] We use the updated version of Twokenizer from Owoputi et al. (2013).

[13] http://nlp.stanford.edu/data/glove.twitter.27B.zip

System	UAS	LAS	Kt/s
Kong et al. (2014)	81.4	76.9	0.3
Dozat et al. (2017)	81.8	77.7	1.7
Ballesteros et al. (2015)	80.2	75.7	2.3
Ensemble (20)	83.4	79.4	0.2
Distillation ($\alpha = 1.0$)	81.8	77.6	2.3
Distillation ($\alpha = 0.9$)	82.0	77.8	2.3
Distillation w/ exploration	82.1	77.9	2.3

Table 6: Dependency parser comparison on TWEEBANK V2 test set, with automatic POS tags. We use Ballesteros et al. (2015) as our baseline and build the ensemble and distilling model over it. The "Kt/s" column shows the parsing speed evaluated by thousands of tokens the model processed per second.

pointed out that neural network training is nondeterministic and depends on the seed for the random number generator. Our preliminary experiments confirm this finding, with a gap of 1.4 LAS on development data between the best (76.2) and worst (74.8) runs. To control for this effect, we report the average of five differently-seeded runs, for each of our models and the compared ones.

Initial results. The first section of Table 6 compares the stack LSTM with TWEEBOPARSER (the system of Kong et al., 2014) and the state-of-the-art parser in the CoNLL 2017 evaluations, due to Dozat et al. (2017). Kong et al.'s (2014) parser is a graph-based parser with lexical features and word cluster and it uses dual decomposition for decoding. The parser in Dozat et al. (2017) is also a graph-based parser but includes character-based word representations and uses a biaffine classifier to predict whether an attachment exists between two words. Both of the compared systems require superlinear runtime due to graph-based parsing. They are re-trained on the same data as our system. Our baseline lags behind by nearly two LAS points but runs faster than both of them.

Ensemble. Due to ambiguity in the training data—which most loss functions are not robust to (Frénay and Verleysen, 2014), including the log loss we use, following Ballesteros et al. (2015)—and due to the instability of neural network training, we follow Dietterich (2000) and consider an ensemble of twenty parsers trained using different random initialization. To parse at test time, the transition probabilities of the twenty members of the ensemble are averaged. The result achieves LAS of 79.4, outperforming all three systems above (Table 6).

Distillation. The shortcoming of the 20-parser ensemble is, of course, that it requires twenty times the runtime of a single greedy parser, making it the slowest system in our comparison. Kuncoro et al. (2016) proposed the distillation of 20 greedy transition-based parser into a single *graph-based* parser; they transformed the votes of the ensemble into a structured loss function. However, as Kuncoro et al. pointed out, it is not straightforward to use a structured loss in a *transition-based* parsing algorithm. Because fast runtime is so important for NLP on social media, we introduce a new way to distill our greedy ensemble into a single transition-based parser (the first such attempt, to our knowledge).

Our approach applies techniques from Hinton et al. (2015) and Kim and Rush (2016) to parsing. Note that training a transition-based parser typically involves the transformation of the training data into a sequence of "oracle" state-action pairs. Let $q(a \mid s)$ denote the distilled model's probability of an action a given parser state s; let $p(a \mid s)$ be the probability under the ensemble (i.e., the average of the 20 separately-trained ensemble members). To train the distilled model, we minimize the interpolation between their distillation loss and the conventional log loss:

$$\begin{aligned}\operatorname{argmin}_q \quad & \alpha \sum_i \underbrace{\sum_a -p(a \mid s_i) \cdot \log q(a \mid s_i)}_{\text{distillation loss}} \\ & + (1-\alpha) \sum_i \underbrace{-\log q(a_i \mid s_i)}_{\text{log loss}}\end{aligned} \tag{1}$$

Distilling from this parser leads to a single greedy transition-based parser with 77.8 LAS—better than past systems but worse than our more expensive ensemble. The effect of α is illustrated in Figure 5; generally paying closer attention to the ensemble, rather than the conventional log loss objective, leads to better performance.

Learning from exploration. When we set $\alpha = 1$, we eliminate the oracle from the estimation procedure (for the distilled model). This presents an opportunity to learn with *exploration*, by randomly sampling transitions from the ensemble, found useful in recent methods for training greedy models that use dynamic oracles (Goldberg and Nivre, 2012, 2013; Kiperwasser and Goldberg,

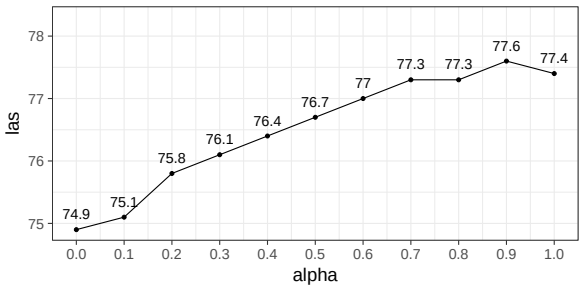


Figure 5: The effect of α on distillation.

Pipeline stage	Score	Ours	SOTA
Tokenization	F_1	98.3	97.3
POS tagging	F_1	93.3	92.2
UD parsing	*LAS* F_1	74.0	71.4

Table 7: Evaluating our pipeline against a state-of-the-art pipeline.

2016; Ballesteros et al., 2016). We find that this approach outperforms the conventional distillation model, coming in 1.5 points behind the ensemble (last line of Table 6).

Pipeline evaluation. Finally, we report our full pipeline's performance in Table 7. We also compare our model with a pipeline of the state-of-the-art systems (labeled "SOTA"): Stanford CoreNLP tokenizer,[14] Owoputi et al.'s (2013) tagger, and Dozat et al.'s (2017) parser. Our system differs from the state-of-the-art pipeline in the tokenization and parser components. From Table 7, our pipeline outperforms the state of the art when evaluated in pipeline manner. The results also emphasize the importance of tokenization: without gold tokenization UD parsing performance drops by about four points.

4 Conclusion

We study the problem of parsing tweets into Universal Dependencies. We adapt the UD guidelines to cover special constructions in tweets and create the TWEEBANK V2, which has 55,607 tokens. We characterize the disagreements among our annotators and argue that inherent ambiguity in this genre makes consistent annotation a challenge. Using this new treebank, we build a pipeline system to parse tweets into UD. We also propose a new method to distill an ensemble of 20 greedy parsers into a single one to overcome annotation noise without sacrificing efficiency. Our parser achieves an improvement of 2.2 in LAS over a strong baseline and outperforms other state-of-the-art parsers in both accuracy and speed.

[14] We choose the Stanford CoreNLP tokenizer in the spirit of comparing rule-based and statistical methods.

Acknowledgments

We thank Elizabeth Clark, Lucy Lin, Nelson Liu, Kelvin Luu, Phoebe Mulcaire, Hao Peng, Maarten Sap, Chenhao Tan, and Sam Thomson at the University of Washington, and Austin Blodgett, Lucia Donatelli, Joe Garman, Emma Manning, Angela Yang, and Yushi Zhang at Georgetown University for their annotation efforts in the first round. We are grateful for the support from Lingpeng Kong at the initial stage of this project. We also thank the anonymous reviewers for their helpful comments and suggestions. This work was supported by the National Key Basic Research Program of China via grant 2014CB340503 and the National Natural Science Foundation of China (NSFC) via grant 61632011.

References

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Many languages, one parser. *TACL* 4.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. In *Proc. of EMNLP*.

Miguel Ballesteros, Yoav Goldberg, Chris Dyer, and Noah A. Smith. 2016. Training with exploration improves a greedy stack LSTM parser. In *Proc. of EMNLP*.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics* 18(4).

Marie-Catherine de Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. 2014. Universal Stanford Dependencies: A cross-linguistic typology. In *LREC*.

Leon Derczynski, Alan Ritter, Sam Clark, and Kalina Bontcheva. 2013. Twitter part-of-speech tagging for all: Overcoming sparse and noisy data. In *Proc. of RANLP 2013*.

Thomas G. Dietterich. 2000. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning* 40(2):139–157. https://doi.org/10.1023/A:1007607513941.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the CoNLL 2017 shared task. In *Proc. of CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.

Jacob Eisenstein. 2013. What to do about bad language on the internet. In *Proc. of NAACL*.

Jennifer Foster, Özlem Çetinoğlu, Joachim Wagner, Joseph Le Roux, Stephen Hogan, Joakim Nivre, Deirdre Hogan, and Josef Van Genabith. 2011. #hardtoparse: POS tagging and parsing the Twitterverse. In *Proc. of the 5th AAAI Conference on Analyzing Microtext*.

Benoît Frénay and Michel Verleysen. 2014. Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks and Learning Systems* 25:845–869.

Kim Gerdes. 2013. Collaborative dependency annotation. In *Proc. of DepLing*.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proc. of ACL*.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *Proc. of COLING*.

Yoav Goldberg and Joakim Nivre. 2013. Training deterministic parsers with non-deterministic oracles. *TACL* 1.

Tao Gui, Qi Zhang, Haoran Huang, Minlong Peng, and Xuanjing Huang. 2017. Part-of-speech tagging for Twitter with adversarial neural networks. In *Proc. of EMNLP*.

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

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR* abs/1503.02531.

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

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proc. of EMNLP*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL* 4.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A. Smith. 2014. A dependency parser for tweets. In *Proc. of EMNLP*.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2016. Distilling an ensemble of greedy dependency parsers into one MST parser. In *Proc. of EMNLP*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proc. of ACL*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL System Demonstrations*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* 19(2):313–330.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proc. of LREC*.

Brendan O'Connor, Michel Krieger, and David Ahn. 2010. TweetMotif: Exploratory search and topic summarization for Twitter.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proc. of NAACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proc. of EMNLP*.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proc. of LREC*.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In *Proc. of EMNLP*.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: An experimental study. In *Proc. of EMNLP*.

Manuela Sanguinetti, Cristina Bosco, Alessandro Mazzei, Alberto Lavelli, and Fabio Tamburini. 2017. Annotating Italian social media texts in Universal Dependencies. In *Proc. of Depling*. Pisa, Italy.

Nathan Schneider, Brendan O'Connor, Naomi Saphra, David Bamman, Manaal Faruqui, Noah A. Smith, Chris Dyer, and Jason Baldridge. 2013. A framework for (under)specifying dependency syntax without overloading annotators. In *Proc. of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*.

Milan Straka and Jana Straková. 2017. Tokenizing, POS tagging, lemmatizing and parsing UD 2.0 with UDPipe. In *Proc. of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.

Hongmin Wang, Yue Zhang, GuangYong Leonard Chan, Jie Yang, and Hai Leong Chieu. 2017. Universal Dependencies parsing for colloquial Singaporean English. In *Proc. of ACL*.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. of IWPT*.

Robust Multilingual Part-of-Speech Tagging via Adversarial Training

Michihiro Yasunaga **Jungo Kasai** **Dragomir Radev**
Department of Computer Science, Yale University
{michihiro.yasunaga,jungo.kasai,dragomir.radev}@yale.edu

Abstract

Adversarial training (AT)[1] is a powerful regularization method for neural networks, aiming to achieve robustness to input perturbations. Yet, the specific effects of the robustness obtained from AT are still unclear in the context of natural language processing. In this paper, we propose and analyze a neural POS tagging model that exploits AT. In our experiments on the Penn Treebank WSJ corpus and the Universal Dependencies (UD) dataset (27 languages), we find that AT not only improves the overall tagging accuracy, but also 1) prevents over-fitting well in low resource languages and 2) boosts tagging accuracy for rare / unseen words. We also demonstrate that 3) the improved tagging performance by AT contributes to the downstream task of dependency parsing, and that 4) AT helps the model to learn cleaner word representations. 5) The proposed AT model is generally effective in different sequence labeling tasks. These positive results motivate further use of AT for natural language tasks.

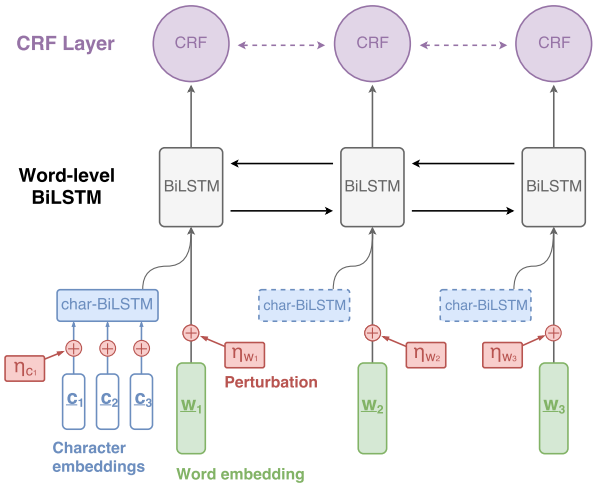


Figure 1: Illustration of our architecture for adversarial POS tagging. Given a sentence, we input the normalized word embeddings ($\underline{\boldsymbol{w}}_1, \underline{\boldsymbol{w}}_2, \underline{\boldsymbol{w}}_3$) and character embeddings (showing $\underline{\boldsymbol{c}}_1, \underline{\boldsymbol{c}}_2, \underline{\boldsymbol{c}}_3$ for w_1). Each word is represented by concatenating its word embedding and its character-level BiLSTM output. They are fed into the main BiLSTM-CRF network for POS tagging. In adversarial training, we compute and add the worst-case perturbation η to all the input embeddings for regularization.

1 Introduction

Recently, neural network-based approaches have become popular in many natural language processing (NLP) tasks including tagging, parsing, and translation (Chen and Manning, 2014; Bahdanau et al., 2015; Ma and Hovy, 2016). However, it has been shown that neural networks tend to be locally unstable and even tiny perturbations to the original inputs can mislead the models (Szegedy et al., 2014). Such maliciously perturbed inputs are called *adversarial examples*. *Adversarial training* (Goodfellow et al., 2015) aims to improve the robustness of a model to input perturbations by training on both unmodified examples and adversarial examples. Previous work (Goodfellow et al., 2015; Shaham et al., 2015) on image recognition has demonstrated the enhanced robustness of their models to unseen images via adversarial training and has provided theoretical explanations of the regularization effects.

Despite its potential as a powerful regularizer, adversarial training (AT) has yet to be explored extensively in natural language tasks. Recently, Miyato et al. (2017) applied AT on text classification, achieving state-of-the-art accuracy. Yet, the specific effects of the robustness obtained from AT are still unclear in the context of NLP. For example, research studies have yet to answer questions such as 1) how can we interpret perturbations or robustness on natural language inputs? 2) how are they related to linguistic factors like vocabulary statis-

[1]We distinguish AT from Generative Adversarial Networks (GANs).

Proceedings of NAACL-HLT 2018, pages 976–986
New Orleans, Louisiana, June 1 - 6, 2018.

tics? 3) are the effects of AT language-dependent? Answering such questions is crucial to understand and motivate the application of adversarial training on natural language tasks.

In this paper, spotlighting a well-studied core problem of NLP, we propose and carefully analyze a neural part-of-speech (POS) tagging model that exploits adversarial training. With a BiLSTM-CRF model (Huang et al., 2015; Ma and Hovy, 2016) as our baseline POS tagger, we apply adversarial training by considering perturbations to input word/character embeddings. In order to demystify the effects of adversarial training in the context of NLP, we conduct POS tagging experiments on multiple languages using the Penn Treebank WSJ corpus (Englsih) and the Universal Dependencies dataset (27 languages), with thorough analyses of the following points:

- Effects on different target languages
- Vocabulary statistics and tagging accuracy
- Influence on downstream tasks
- Representation learning of words

In our experiments, we find that our adversarial training model consistently outperforms the baseline POS tagger, and even achieves state-of-the-art results on 22 languages. Furthermore, our analyses reveal the following insights into adversarial training in the context of NLP:

- The regularization effects of adversarial training (AT) are general across different languages. AT can prevent overfitting especially well when training examples are scarce, providing an effective tool to process low resource languages.
- AT can boost the tagging performance for rare/unseen words and increase the sentence-level accuracy. This positively affects the performance of down-stream tasks such as dependency parsing, where low sentence-level POS accuracy can be a bottleneck (Manning, 2011).
- AT helps the network learn cleaner word embeddings, showing stronger correlations with their POS tags.

We argue that the effects of AT can be interpreted from the perspective of natural language. Finally, we demonstrate that the proposed AT model is generally effective across different sequence labeling tasks. This work therefore provides a strong motivation and basis for utilizing adversarial training in NLP tasks.

2 Related Work

2.1 POS Tagging

Part-of-speech (POS) tagging is a fundamental NLP task that facilitates downstream tasks such as syntactic parsing. While current state-of-the-art POS taggers (Ling et al., 2015; Ma and Hovy, 2016) yield accuracy over 97.5% on PTB-WSJ, there still remain issues. The per token accuracy metric is easy since taggers can easily assign correct POS tags to highly unambiguous tokens, such as punctuation (Manning, 2011). Sentence-level accuracy serves as a more realistic metric for POS taggers but it still remains low. Another problem with current POS taggers is that their accuracy deteriorates drastically on low resource languages and rare words (Plank et al., 2016). In this work, we demonstrate that adversarial training (AT) can mitigate these issues.

It is empirically shown that POS tagging performance can greatly affect downstream tasks such as dependency parsing (Dozat et al., 2017). In this work, we also demonstrate that the improvements obtained from our AT POS tagger actually contribute to dependency parsing. Nonetheless, parsing with gold POS tags still yields better results, bolstering the view that POS tagging is an essential task in NLP that needs further development.

2.2 Adversarial Training

The concept of adversarial training (Szegedy et al., 2014; Goodfellow et al., 2015) was originally introduced in the context of image classification to improve the robustness of a model by training on input images with malicious perturbations. Previous work (Goodfellow et al., 2015; Shaham et al., 2015; Wang et al., 2017) has provided a theoretical framework to understand adversarial examples and the regularization effects of adversarial training (AT) in image recognition.

Recently, Miyato et al. (2017) applied AT to a natural language task (text classification) by extending the concept of adversarial perturbations to word embeddings. Wu et al. (2017) further explored the possibility of AT in relation extraction. Both report improved performance on their tasks via AT, but the specific effects of AT have yet to be analyzed. In our work, we aim to address this issue by providing detailed analyses on the effects of AT from the perspective of NLP, such as different languages, vocabulary statistics, word embedding distribution, and aim to motivate future research

that exploits AT in NLP tasks.

AT is related to other regularization methods that add noise to data such as dropout (Srivastava et al., 2014) and its variant for NLP tasks, word dropout (Iyyer et al., 2015). Xie et al. (2017) discuss various data noising techniques for language modeling. While these methods produce random noise, AT generates perturbations that the current model is particularly vulnerable to, and thus is claimed to be effective (Goodfellow et al., 2015).

It should be noted that while related in name, adversarial training (AT) differs from Generative Adversarial Networks (GANs) (Goodfellow et al., 2014). GANs have already been applied to NLP tasks such as dialogue generation (Li et al., 2017) and transfer learning (Kim et al., 2017; Gui et al., 2017). Adversarial training also differs from adversarial *evaluation*, recently proposed for reading comprehension tasks (Jia and Liang, 2017).

3 Method

In this section, we introduce our baseline POS tagging model and explain how we implement adversarial training on top.

3.1 Baseline POS Tagging Model

Following the recent top-performing models for sequence labeling tasks (Plank et al., 2016; Lample et al., 2016; Ma and Hovy, 2016), we employ a Bi-directional LSTM-CRF model as our baseline (see Figure 1 for an illustration).

Character-level BiLSTM. Prior work has shown that incorporating character-level representations of words can boost POS tagging accuracy by capturing morphological information present in each language. Major neural character-level models include the character-level CNN (Ma and Hovy, 2016) and (Bi)LSTM (Dozat et al., 2017). A Bi-directional LSTM (BiLSTM) (Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997) processes each sequence both forward and backward to capture sequential information, while preventing the vanishing / exploding gradient problem. We observed that the character-level BiLSTM outperformed the CNN by 0.1% on the PTB-WSJ development set, and hence in all of our experiments we use the character-level BiLSTM. Specifically, we generate a character-level representation for each word by feeding its character embeddings into the BiLSTM and obtaining the concatenated final states.

Word-level BiLSTM. Each word in a sentence is represented by concatenating its word embedding and its character-level representation. They are fed into another level of BiLSTM (word-level BiLSTM) to process the entire sentence.

CRF. In sequence labeling tasks it is beneficial to consider the correlations between neighboring labels and jointly decode the best chain of labels for a given sentence. With this motivation, we apply a conditional random field (CRF) (Lafferty et al., 2001) on top of the word-level BiLSTM to perform POS tag inference with global normalization, addressing the "label bias" problem. Specifically, given an input sentence, we pass the output sequence of the word-level BiLSTM to a first-order chain CRF to compute the conditional probability of the target label sequence:

$$p(\boldsymbol{y} \mid \boldsymbol{s}; \boldsymbol{\theta})$$

where $\boldsymbol{\theta}$ represents all of the model parameters (in the BiLSTMs and CRF), $\boldsymbol{s}$ and $\boldsymbol{y}$ denote the input embeddings and the target POS tag sequence, respectively, for the given sentence.

For training, we minimize the negative log-likelihood (loss function)

$$L(\boldsymbol{\theta}; \boldsymbol{s}, \boldsymbol{y}) = -\log p(\boldsymbol{y} \mid \boldsymbol{s}; \boldsymbol{\theta}) \quad (1)$$

with respect to the model parameters. Decoding searches for the POS tag sequence $\boldsymbol{y}^*$ with the highest conditional probability using the Viterbi algorithm. For more detail about the BiLSTM-CRF formulation, refer to Ma and Hovy (2016).

3.2 Adversarial Training

Adversarial training (Goodfellow et al., 2015) is a powerful regularization method, primarily explored in image recognition to improve the robustness of classifiers to input perturbations. Given a classifier, we first generate input examples that are very close to original inputs (so should yield the same labels) yet are likely to be misclassified by the current model. Specifically, these *adversarial examples* are generated by adding small perturbations to the inputs in the direction that significantly increases the loss function of the classifier (*worst-case* perturbations). Then, the classifier is trained on the mixture of clean examples and adversarial examples to improve the stability to input perturbations. In this work, we incorporate adversarial training into our baseline POS tagger, aiming to achieve better regularization effects and to provide their interpretations in the context of NLP.

Generating adversarial examples. Adversarial training (AT) considers continuous perturbations to inputs, so we define perturbations at the level of dense word / character embeddings rather than one-hot vector representations, similarly to Miyato et al. (2017). Specifically, given an input sentence, we consider the concatenation of all the word/character embeddings in the sentence: $\boldsymbol{s} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{c}_1, \boldsymbol{c}_2, \ldots]$. To prepare an adversarial example, we aim to generate the worst-case perturbation of a small bounded norm ϵ that maximizes the loss function L of the current model:

$$\boldsymbol{\eta} = \underset{\boldsymbol{\eta}': \|\boldsymbol{\eta}'\|_2 \leq \epsilon}{\arg\max}\ L(\hat{\boldsymbol{\theta}}; \boldsymbol{s} + \boldsymbol{\eta}', \boldsymbol{y})$$

where $\hat{\boldsymbol{\theta}}$ is the current value of the model parameters, treated as a constant, and $\boldsymbol{y}$ denotes the target labels. Since the exact computation of such $\boldsymbol{\eta}$ is intractable in complex neural networks, we employ the Fast Gradient Method (Liu et al., 2017; Miyato et al., 2017) i.e. first order approximation to obtain an approximate worst-case perturbation of norm ϵ, by a single gradient computation:

$$\boldsymbol{\eta} = \epsilon\, \boldsymbol{g}/\|\boldsymbol{g}\|_2, \text{ where } \boldsymbol{g} = \nabla_{\boldsymbol{s}} L(\hat{\boldsymbol{\theta}}; \boldsymbol{s}, \boldsymbol{y}) \quad (2)$$

ϵ is a hyperparameter to be determined in the development dataset. Note that the perturbation $\boldsymbol{\eta}$ is generated in the direction that significantly increases the loss L. We find such $\boldsymbol{\eta}$ against the current model parameterized by $\hat{\boldsymbol{\theta}}$, at each training step, and construct an adversarial example by

$$\boldsymbol{s}_{\text{adv}} = \boldsymbol{s} + \boldsymbol{\eta}$$

However, if we do not restrict the norm of word / character embeddings, the model could trivially learn embeddings of large norms to make the perturbations insignificant. To prevent this issue, we normalize word/character embeddings so that they have mean 0 and variance 1 for every entry, as in Miyato et al. (2017). The normalization is performed every time we feed input embeddings into the LSTMs and generate adversarial examples. To ensure a fair comparison, we also normalize input embeddings in our baseline model.

While Miyato et al. (2017) set the norm of a perturbation ϵ (Eq 2) to be a fixed value for all input sentences, to generate adversarial examples for an entire sentence of a variable length and to include character embeddings besides word embeddings, we make the perturbation size ϵ adaptive to the dimension of the concatenated input embedding $\boldsymbol{s} \in \mathbb{R}^D$. We set ϵ to be $\alpha\sqrt{D}$ (i.e., proportional to $\sqrt{D}$), as the expected squared norm of $\boldsymbol{s}$ after the embedding normalization is D. The scaling factor α is selected from $\{0.001, 0.005, 0.01, 0.05, 0.1\}$ based on the development performance in each treebank. We used 0.01 for PTB-WSJ and UD-Spanish, and 0.05 for the rest. Note that $\alpha = 0$ would generate no noise (identical to the baseline); if $\alpha = 1$, the generated adversarial perturbation would have a norm comparable to the original embedding, which could change the semantics of the input sentence (Wu et al., 2017). Hence, the optimal perturbation scale α should lie in between and be small enough to preserve the semantics of the original input.

Adversarial training. At each training step, we generate adversarial examples against the current model, and train on the mixture of clean examples and adversarial examples to achieve robustness to input perturbations. To this end, we define the loss function for adversarial training as:

$$\tilde{L} = \gamma L(\boldsymbol{\theta}; \boldsymbol{s}, \boldsymbol{y}) + (1 - \gamma) L(\boldsymbol{\theta}; \boldsymbol{s}_{\text{adv}}, \boldsymbol{y})$$

where $L(\boldsymbol{\theta}; \boldsymbol{s}, \boldsymbol{y})$, $L(\boldsymbol{\theta}; \boldsymbol{s}_{\text{adv}}, \boldsymbol{y})$ represent the loss from a clean example and the loss from its adversarial example, respectively, and γ determines the weighting between them. We used $\gamma = 0.5$ in all our experiments. This objective function can be optimized with respect to the model parameters $\boldsymbol{\theta}$, in the same manner as the baseline model.

4 Experiments

To fully analyze the effects of adversarial training, we train and evaluate our baseline/adversarial POS tagging models on both a standard English dataset and a multilingual dataset.

4.1 Datasets

As a standard English dataset, we use the Wall Street Journal (WSJ) portion of the Penn Treebank (PTB) (Marcus et al., 1993), containing 45 different POS tags. We adopt the standard split: sections 0-18 for training, 19-21 for development and 22-24 for testing (Collins, 2002; Manning, 2011).

For multilingual POS tagging experiments, to compare with prior work, we use treebanks from Universal Dependencies (UD) v1.2 (Nivre et al., 2015) (17 POS) with the given data splits. We experiment on languages for which pre-trained Polyglot word embeddings (Al-Rfou et al., 2013) are available, resulting in 27 languages listed in Table 2. We regard languages with less than 60k tokens of training data as low-resource (Table 2, bottom), as in Plank et al. (2016).

Model	Accuracy
Toutanova et al. (2003)	97.27
Manning (2011)	97.28
Collobert et al. (2011)	97.29
Søgaard (2011)	97.50
Ling et al. (2015)	**97.78**
Ma and Hovy (2016)	97.55
Yang et al. (2017)	97.55
Hashimoto et al. (2017)	97.55
Ours – Baseline (BiLSTM-CRF)	97.54
Ours – Adversarial	97.58

Table 1: POS tagging accuracy on the PTB-WSJ test set, with other top-performing systems.

4.2 Training & Evaluation Details

Model settings. We initialize word embeddings with 100-dimensional GloVe (Pennington et al., 2014) for English, and with 64-dimensional Polyglot (Al-Rfou et al., 2013) for other languages. We use 30-dimensional character embeddings, and set the state sizes of character/word-level BiLSTM to be 50, 200 for English, 50, 100 for low resource languages, and 50, 150 for other languages. The model parameters and character embeddings are randomly initialized, as in Ma and Hovy (2016). We apply dropout (Srivastava et al., 2014) to input embeddings and BiLSTM outputs for both baseline and adversarial training, with dropout rate 0.5.

Optimization. We train the model parameters and word/character embeddings by the mini-batch stochastic gradient descent (SGD) with batch size 10, momentum 0.9, initial learning rate 0.01 and decay rate 0.05. We also use a gradient clipping of 5.0 (Pascanu et al., 2012). The models are trained with early stopping (Caruana et al., 2001) based on the development performance.

Evaluation. We evaluate per token tagging accuracy on test sets. We repeat the experiment three times and report the statistical significance.

4.3 Results

PTB-WSJ dataset. Table 1 shows the POS tagging results. As expected, our baseline (BiLSTM-CRF) model (accuracy 97.54%) performs on par with other state-of-the-art systems. Built upon this baseline, our adversarial training (AT) model reaches accuracy 97.58% thanks to its regularization power, outperforming recent POS taggers except Ling et al. (2015). The improvement over the baseline is statistically significant, with p-value $<$ 0.05 on the t-test. We provide additional analysis on this result in later sections.

	Our Models		Plank et al. (2016)			Berend	Nguyen et
	Baseline	Adversarial	BiLSTM	TNT	CRF	(2017)	al. (2017)
bg	98.34	**98.53**	97.97	96.84	96.36	95.63	97.4
cs	98.70	**98.81**	98.24	96.82	96.56	95.83	–
da	96.63	**96.74**	96.35	94.29	93.83	93.32	95.8
de•	94.29	**94.35**	93.38	92.64	91.38	90.73	92.7
en	95.72	**95.82**	95.16	94.55	93.35	93.47	94.7
es	96.26	**96.44**	95.74	94.55	94.23	94.69	95.9
eu•	94.55	94.71	**95.51**	93.35	91.63	90.63	93.7
fa	97.38	**97.51**	97.49	95.98	95.65	96.11	96.8
fi•	94.54	95.40	**95.85**	93.59	90.32	89.19	94.6
fr	96.48	**96.63**	96.11	94.51	95.14	94.96	96.0
he	97.34	**97.43**	96.96	93.71	93.63	95.28	–
hi	97.12	**97.21**	97.10	94.53	96.00	96.09	96.4
hr•	96.12	96.32	**96.82**	94.06	93.16	93.53	–
id	93.95	**94.03**	93.41	93.16	92.96	92.02	93.1
it	98.04	**98.08**	97.95	96.16	96.43	96.28	97.5
nl	92.64	93.09	**93.30**	88.54	90.03	85.10	91.4
no	97.88	**98.08**	98.03	96.31	96.21	95.67	97.4
pl•	97.34	97.57	**97.62**	95.57	93.96	93.95	96.3
pt	97.94	**98.07**	97.90	96.27	96.32	95.50	97.5
sl•	97.81	**98.11**	96.84	94.92	94.77	92.70	97.1
sv	96.39	**96.70**	96.69	95.19	94.45	94.62	–
Avg	96.45	**96.65**	96.40	94.55	94.11	93.59	95.55
el	98.18	**98.24**	–	–	–	97.12	–
et•	90.79	**91.32**	–	–	–	86.30	–
ga	90.66	**91.11**	–	–	–	88.82	–
hu•	93.39	**94.02**	–	–	–	89.47	–
ro	91.24	**91.46**	–	–	–	88.99	–
ta	82.91	**83.16**	–	–	–	81.80	–
Avg	91.20	**91.55**	–	–	–	88.41	–

Table 2: POS tagging accuracy (test) for 27 UD v1.2 treebanks, with other recent works, Plank et al. (2016), Berend (2017) and Nguyen et al. (2017). For Plank et al. (2016), we include the traditional baselines TNT and CRF, and their state-of-the-art model that employs a multi-task BiLSTM. Languages with • are morphologically rich, and those at the bottom ('el' to 'ta') are low-resource, containing less than 60k tokens in their training sets.

Multilingual dataset (UD). Experimental results are summarized in Table 2. Our AT model shows clear advantages over the baseline in all of the 27 languages (average improvement ∼0.25%; see the two shaded columns). Considering that our baseline (BiLSTM-CRF) is already a top performing model for POS tagging, these improvements made by AT are substantial. The improvements are also statistically significant for all the languages, with p-value $<$ 0.05 on the t-test, suggesting that the regularization by AT is generally effective across different languages. Moreover, our AT model achieves state-of-the-art on nearly all of the languages, except the five where Plank et al. (2016)'s multi-task BiLSTM yielded better results. Among the five, most languages are morphologically rich (•).[2] We suspect that their joint training of word rarity may be of particular help in processing morphologically complex words.

[2]We followed the criteria of morphological richness used in Nguyen et al. (2017).

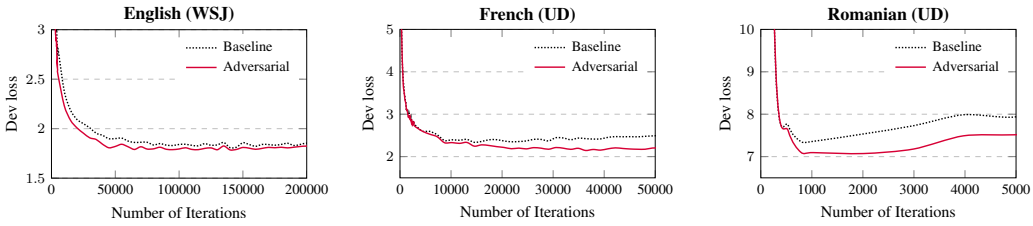


Figure 2: Learning curves for three representative languages (Romanian is low-resource). We show the transition of loss (defined in Eq 1) on the development sets.

English (WSJ)

Word Frequency	0	1-10	10-100	100-	Total
# Tokens	3240	7687	20908	97819	129654
Baseline	**92.25**	95.36	96.03	98.19	97.53
Adversarial	92.01	**95.52**	96.10	98.23	97.57

French (UD)

Word Frequency	0	1-10	10-100	100-	Total
# Tokens	356	839	1492	4523	7210
Baseline	87.64	94.05	94.03	98.43	96.48
Adversarial	87.92	**94.88**	94.03	98.50	96.63

Table 3: POS tagging accuracy (test) on different subsets of words, categorized by their frequency of occurrence in training. The second row shows the number of tokens in the test set that are in each category. The third and fourth rows show the performance of our two models. Better scores are underlined. The biggest improvement is **in bold**.

English (WSJ)

Word Frequency	0	1-10	10-100	100-	Total
# Tokens	6480	15374	41815	195637	259306
Baseline	97.76	97.71	97.80	97.45	97.53
Adversarial	**98.06**	97.71	97.89	97.47	97.57

French (UD)

Word Frequency	0	1-10	10-100	100-	Total
# Tokens	712	1678	2983	9045	14418
Baseline	95.08	97.08	97.58	96.11	96.48
Adversarial	**95.37**	97.26	97.79	96.23	96.63

Table 4: POS tagging accuracy (test) on *neighboring* words. We cluster all words in the test set in the same way as Table 3 and consider the tagging performance on the neighbors (left and right) of these words in the test text.

Additionally, we see that our AT model achieves notably large improvements over the baseline in resource-poor languages (the bottom of Table 2), with average improvement 0.35%, as compared to that for resource-rich languages, 0.20%. To further visualize the regularization effects, we present the learning curves for three representative languages, English (WSJ), French (UD-fr) and Romanian (UD-ro, low-resource), based on the development loss (see Figure 2). For all the three languages, we can observe that the AT model (red solid line) prevents overfitting better than the baseline (black dotted line), and this advantage is more significant in low resource languages. For example, in Romanian, the baseline model starts to increase development loss after 1,000 iterations even with dropout, whereas the AT model keeps improving until 2,500 iterations, achieving notably lower development loss (0.4 down). These results illustrate that AT can prevent overfitting especially well on small datasets and can augment the regularization power beyond dropout. AT can also be viewed as an effective means of data augmentation, where we generate and train with new examples the current model is particularly vulnerable to at every time step, enhancing the robustness of the model. AT can therefore be a promising tool to process low resource languages.

5 Analysis

In the previous sections, we demonstrated the regularization power of adversarial training (AT) on different languages, based on the overall POS tagging performance and learning curves. In this section, we conduct further analyses on the robustness of AT from NLP specific aspects such as word statistics, sequence modeling, downstream tasks, and word representation learning.

We find that AT can boost tagging accuracy on rare words and neighbors of unseen words (§5.1). Furthermore, this robustness against rare / unseen words leads to better sentence-level accuracy and downstream dependency parsing (§5.2). We illustrate these findings using two major languages, English (WSJ) and French (UD), which have substantially large training and testing data to discuss vocabulary statistics and sentence-level performance. Finally, we study the effects of AT on word representation learning (§5.3), and the applicability of AT to different sequential tasks (§5.4).

5.1 Word-level Analysis

Poor tagging accuracy on rare/unseen words is one of the bottlenecks in current POS taggers (Manning, 2011; Plank et al., 2016). Aiming to reveal

English (WSJ)

	Sentence-level Acc.	Stanford Parser UAS	Stanford Parser LAS	Parsey McParseface UAS	Parsey McParseface LAS
Baseline	59.08	91.53	89.30	91.68	87.92
Adversarial	**59.61**	**91.57**	**89.35**	**91.73**	**87.97**
(w/ gold tags)	–	(92.07)	(90.63)	(91.98)	(88.60)

French (UD)

	Sentence-level Acc.	Parsey Universal UAS	Parsey Universal LAS
Baseline	52.35	84.85	80.36
Adversarial	**53.36**	**85.01**	**80.55**
(w/ gold tags)	–	(85.05)	(80.75)

Table 5: Sentence-level accuracy and downstream dependency parsing performance by our baseline/adversarial POS taggers.

the effects of AT on rare/unseen words, we analyze tagging performance at the word level, considering vocabulary statistics.

Word frequency. To define rare/unseen words, we consider each word's frequency of occurrence in the training set. We categorize all words in the test set based on this frequency and study the test tagging accuracy for each group (see Table 3).[3] In both languages, the AT model achieves large improvements over the baseline on rare words (e.g., frequency 1-10 in training), as opposed to more frequent words. This result again corroborates the data augmentation power of AT under small training examples. On the other hand, we did not observe meaningful improvements on unseen words (frequency 0 in training). A possible explanation is that AT can facilitate the learning of words with at least a few occurrences in training (rare words), but is not particularly effective in inferring the POS tags of words for which no training examples are given (unseen words).

Neighboring words. One important characteristic of natural language tasks is the sequential nature of inputs (i.e., sequence of words), where each word influences the function of its neighboring words. Since our model uses BiLSTM-CRF for that reason, we also study the tagging performance on the neighbors of rare/unseen words, and analyze the effects of AT with the sequence model in mind. In Table 4, we cluster all words in the test set based on their frequency in training again, and consider the tagging accuracy on the neighbors (left and right) of these words in the test text. We observe that AT tends to achieve large improvements over the baseline on the neighbors of unseen words (training frequency 0), while the improvements on the neighbors of more frequent words remain moderate. Our AT model thus exhibits strong stability to uncertain neighbors, as compared to the baseline. We suspect that because we generate adversarial examples against entire input sentences, training with adversarial examples makes the model more robust not only to perturbations in each word but also to perturbations in its neighboring words, leading to greater stability to uncertain neighbors.

[3]To conduct the analysis, we picked the median result from the three repeated experiments.

5.2 Sentence-level & Downstream Analysis

In the word-level analysis, we showed that AT can boost tagging accuracy on rare words and the neighbors of unseen words, enhancing overall robustness on rare/unseen words. In this section, we discuss the benefit of our improved POS tagger in a major downstream task, dependency parsing.

Most of the recent state-of-the-art dependency parsers take predicted POS tags as input (e.g. Chen and Manning (2014); Andor et al. (2016); Dozat and Manning (2017)). Dozat et al. (2017) empirically show that their dependency parser gains significant improvements by using POS tags predicted by a Bi-LSTM POS tagger, while POS tags predicted by the UDPipe tagger (Straka et al., 2016) do not contribute to parsing performance as much. This observation illustrates that POS tagging performance has a great influence on dependency parsing, motivating the hypothesis that the POS tagging improvements gained from our adversarial training help dependency parsing.

To test the hypothesis, we consider three settings in dependency parsing of English and French: using POS tags predicted by the baseline model, using POS tags predicted by the AT model, and using gold POS tags. For English (PTB-WSJ), we first convert the treebank into Stanford Dependencies (SD) using Stanford CoreNLP (ver 3.8.0) (Manning et al., 2014), and then apply two well-known dependency parsers: Stanford Parser (ver 3.5.0) (Chen and Manning, 2014) and Parsey McParseface (SyntaxNet) (Andor et al., 2016). For French (UD), we use Parsey Universal from SyntaxNet. The three parsers are all publicly available and pre-trained on corresponding treebanks.

Table 5 shows the results of the experiments. We can observe improvements in both languages by using the POS tags predicted by our AT POS tagger. As Manning (2011) points out, when pre-

English (WSJ)

POS Cluster	NN	VB	JJ	RB	Avg.
1) Initial (GloVe)	0.243	0.426	0.220	0.549	0.359
2) Baseline	0.280	0.431	**0.309**	0.667	0.422
3) Adversarial	**0.281**	**0.436**	0.306	**0.675**	**0.424**

French (UD)

POS Cluster	NOUN	VERB	ADJ	ADV	Avg.
1) Initial (polyglot)	0.215	0.233	0.210	0.540	0.299
2) Baseline	0.258	0.271	0.262	0.701	0.373
3) Adversarial	**0.263**	**0.272**	**0.263**	**0.720**	**0.379**

Table 6: Cluster tightness evaluation for word embeddings, based on the cosine similarity measure. Higher scores indicate better clustering (cleaner word vector distribution). Each row corresponds to word vectors 1) at the beginning, 2) after baseline training, and 3) after adversarial training.

English (WSJ)

Perturbation scale α	0	0.001	0.01	0.05	0.1	0.5
Avg. cluster tightness	0.422	0.423	0.424	0.429	**0.436**	0.429

Table 7: Average cluster tightness for word embeddings trained with varied perturbation scale α (0 indicates baseline training).

dicted POS tags are used for downstream dependency parsing, a single bad mistake in a sentence can greatly damage the usefulness of the POS tagger. The robustness of our AT POS tagger against rare/unseen words helps to mitigate such an issue. This advantage can also be observed from the AT POS tagger's notably higher sentence-level accuracy than the baseline (see Table 5 left). Nonetheless, gold POS tags still yield better parsing results as compared to the baseline/AT POS taggers, supporting the claim that POS tagging needs further improvement for downstream tasks.

5.3 Effects on Representation Learning

Next, we perform an analysis on representation learning of words (word embeddings) for the English (PTB-WSJ) and French (UD) experiments. We hypothesize that adversarial training (AT) helps to learn better word embeddings so that the POS tag prediction of a word cannot be influenced by a small perturbation in the input embedding.

To verify this hypothesis, we cluster all words in the test set based on their correct POS tags[4] and evaluate the tightness of the word vector distribution within each cluster. We compare this clustering quality among the three settings: 1) beginning (initialized with GloVe or Polyglot), 2) after baseline training (50 epochs), and 3) after adversarial training (50 epochs), to study the effects of AT on word representation learning.

For evaluating the tightness of word vector distribution, we employ the cosine similarity metric, which is widely used as a measure of the closeness between two word vectors (e.g., Mikolov et al. (2013); Pennington et al. (2014)). To measure the tightness of each cluster, we compute the cosine similarity for every pair of words within, and then take the average. We also report the average tightness across all the clusters.

The evaluation results are summarized in Table 6. We report the tightness scores for the four major clusters: *noun*, *verb*, *adjective*, and *adverb* (from left to right). As can be seen from the table, for both languages, adversarial training (AT) results in cleaner word embedding distributions than the baseline, with a higher cosine similarity within each POS cluster, and with a clear advantage in the average tightness across all the clusters. In other words, the learned word vectors show stronger correlations with their POS tags. This result confirms that training with adversarial examples can help to learn cleaner word embeddings so that the meaning / grammatical function of a word cannot be altered by a small perturbation in its embedding. This analysis provides a means to interpret the robustness to input perturbations, from the perspective of NLP.

Relation with perturbation size ϵ. We also study how the size of added perturbations influences word representation learning in adversarial training. Recall that we set the norm of a perturbation ϵ to be $\alpha\sqrt{D}$, where D is the dimension of the concatenated input embeddings (see §3.2). For instance, $\alpha = 0$ would produce no noise; $\alpha = 1$ would generate a perturbation of a norm equivalent to the original word embeddings. We hypothesize that AT facilitates word representation learning when α is small enough to preserve the semantics of input words, but can hinder the learning when α is too large. To test the hypothesis, we repeat the clustering evaluation for word embeddings trained with varied perturbation scale α: 0, 0.001, 0.01, 0.05, 0.1, 0.5 (see Table 7). We observe that the quality of learned word embedding distribution keeps improving as α goes up from 0 to 0.1, but starts to drop around $\alpha = 0.5$. We also find that this optimal α in word embedding learning (i.e., 0.1) is larger than the α which

[4] We excluded words with multiple tags in the test text.

Model	F1
Tsuruoka et al. (2011)	93.81
Collobert et al. (2011)	94.32
Yang et al. (2017)	94.66
Suzuki and Isozaki (2008)	95.15
Søgaard and Goldberg (2016)	95.56
Hashimoto et al. (2017)	95.77
Peters et al. (2017)	**96.37**
Ours – Baseline (BiLSTM-CRF)	95.18
Ours – Adversarial	95.25

Table 8: Chunking F1 scores on the CoNLL-2000 task, with other top performing models.

Model	F1
Collobert et al. (2011)	89.59
Huang et al. (2015)	90.10
Chiu and Nichols (2016)	90.91
Lample et al. (2016)	90.94
Luo et al. (2015)	91.20
Ma and Hovy (2016)	91.21
Peters et al. (2017)	**91.93**
Ours – Baseline (BiLSTM-CRF)	91.22
Ours – Adversarial	91.56

Table 9: NER F1 scores on the CoNLL-2003 (English) task, with other top performing models.

yielded the best tagging performance on development sets (i.e., 0.01 or 0.05). A possible explanation is that while word embeddings can adapt to relatively large α (e.g., 0.1) during training, as adversarial perturbations are generated at the embedding level, such α could change the semantics of the input from the current tagging model's perspective and hinder the training of *tagging*.

5.4 Other Sequence Labeling Tasks

Finally, to further confirm the applicability of AT, we experiment with our BiLSTM-CRF AT model in different sequence labeling tasks: chunking and named entity recognition (NER).

Chunking can be performed as a sequence labeling task that assigns a chunking tag (B-NP, I-VP, etc.) to each word. We conduct experiments on the CoNLL 2000 shared task with the standard data split: PTB-WSJ Sections 15-18 for training and 20 for testing. We use Section 19 as the development set and employ the IOBES tagging scheme, following Hashimoto et al. (2017).

NER aims to assign an entity type to each word, such as *person*, *location*, *organization*, and *misc*. We conduct experiments on the CoNLL-2003 (English) shared task (Tjong Kim Sang and De Meulder, 2003), adopting the IOBES tagging scheme as in (Lample et al., 2016; Ma and Hovy, 2016).

The results are summarized in Table 8 and 9. AT enhanced F1 score from the baseline BiLSTM-CRF model's 95.18 to 95.25 for chunking, and from 91.22 to 91.56 for NER, also significantly outperforming Ma and Hovy (2016). These improvements made by AT are bigger than that for English POS tagging, most likely due to the larger room for improvement in chunking and NER. The improvements are again statistically significant, with p-value < 0.05 on the t-test. The experimental results suggest that the proposed adversarial training scheme is generally effective across different sequence labeling tasks.

Our BiLSTM-CRF AT model did not reach the performance by Hashimoto et al. (2017)'s multi-task model and Peters et al. (2017)'s state-of-the-art system that incorporates pretrained language models. It would be interesting future work to combine the strengths of these joint models (e.g., syntactic and semantic aids) and adversarial training (e.g., robustness).

6 Conclusion

We proposed and carefully analyzed a POS tagging model that exploits adversarial training (AT). In our multilingual experiments, we find that AT achieves substantial improvements on all the languages tested, especially on low resource ones. AT also enhances the robustness to rare/unseen words and sentence-level accuracy, alleviating the major issues of current POS taggers, and contributing to the downstream task, dependency parsing. Furthermore, our analyses on different languages, word/neighbor statistics and word representation learning reveal the effects of AT from the perspective of NLP. The proposed AT model is applicable to general sequence labeling tasks. This work therefore provides a strong basis and motivation for utilizing AT in natural language tasks.

Acknowledgements

We would like to thank Rui Zhang, Jonathan Kummerfeld, Yutaro Yamada, as well as all the anonymous reviewers for their helpful feedback and suggestions on this work.

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. In *CoNLL*.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav

Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *ACL*.

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

Gábor Berend. 2017. Sparse coding of neural word embeddings for multilingual sequence labeling. *TACL* .

Rich Caruana, Steve Lawrence, and C Lee Giles. 2001. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *NIPS*.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *EMNLP*.

Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. In *TACL*.

Michael Collins. 2002. Discriminative training methods for hidden markov models. In *EMNLP*.

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

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *ICLR*.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the conll 2017 shared task. In *CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. pages 20–30.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NIPS*.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *ICLR*.

Tao Gui, Qi Zhang, Haoran Huang, Minlong Peng, and Xuanjing Huang. 2017. Part-of-speech tagging for twitter with adversarial neural networks. In *EMNLP*.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple NLP tasks. In *EMNLP*.

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

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991* .

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *ACL*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *EMNLP*.

Joo-Kyung Kim, Young-Bum Kim, Ruhi Sarikaya, and Eric Fosler-Lussier. 2017. Cross-lingual transfer learning for pos tagging without cross-lingual resources. In *EMNLP*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL*.

Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. In *EMNLP*.

Wang Ling, Tiago Luís, Luís Marujo, Rámon Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *EMNLP*.

Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. 2017. Delving into transferable adversarial examples and black-box attacks. In *ICLR*.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint entity recognition and disambiguation. In *EMNLP*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *ACL*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*.

Christopher D Manning. 2011. Part-of-speech tagging from 97% to 100%: is it time for some linguistics? *Computational Linguistics and Intelligent Text Processing* pages 171–189.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2017. Adversarial training methods for semi-supervised text classification. In *ICLR*.

Dat Quoc Nguyen, Mark Dras, and Mark Johnson. 2017. A Novel Neural Network Model for Joint POS Tagging and Graph-based Dependency Parsing. In *CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.

Joakim Nivre, Željko Agić, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Cristina Bosco, Sam Bowman, Giuseppe G. A. Celano, Miriam Connor, Marie-Catherine de Marneffe, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Tomaž Erjavec, Richárd Farkas, Jennifer Foster, Daniel Galbraith, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Yoav Goldberg, Berta Gonzales, Bruno Guillaume, Jan Hajič, Dag Haug, Radu Ion, Elena Irimia, Anders Johannsen, Hiroshi Kanayama, Jenna Kanerva, Simon Krek, Veronika Laippala, Alessandro Lenci, Nikola Ljubešić, Teresa Lynn, Christopher Manning, Cătălina Mărănduc, David Mareček, Héctor Martínez Alonso, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Shunsuke Mori, Hanna Nurmi, Petya Osenova, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Prokopis Prokopidis, Sampo Pyysalo, Loganathan Ramasamy, Rudolf Rosa, Shadi Saleh, Sebastian Schuster, Wolfgang Seeker, Mojgan Seraji, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Kiril Simov, Aaron Smith, Jan Štěpánek, Alane Suhr, Zsolt Szántó, Takaaki Tanaka, Reut Tsarfaty, Sumire Uematsu, Larraitz Uria, Viktor Varga, Veronika Vincze, Zdeněk Žabokrtský, Daniel Zeman, and Hanzhi Zhu. 2015. Universal dependencies 1.2. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063* .

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

Matthew E Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *ACL*.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *ACL*.

M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *Trans. Sig. Proc.* 45(11):2673–2681.

Uri Shaham, Yutaro Yamada, and Sahand Negahban. 2015. Understanding adversarial training: Increasing local stability of neural nets through robust optimization. *arXiv preprint arXiv:1511.05432* .

Anders Søgaard. 2011. Semi-supervised condensed nearest neighbor for part-of-speech tagging. In *ACL-HLT*.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *ACL*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15:1929–1958.

Milan Straka, Jan Hajic, and Jana Straková. 2016. Udpipe: Trainable pipeline for processing conll-u files performing tokenization, morphological analysis, pos tagging and parsing. In *LREC*.

Jun Suzuki and Hideki Isozaki. 2008. Semi-supervised sequential labeling and segmentation using gigaword scale unlabeled data. *ACL-HLT* .

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *ICLR*.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *CoNLL*.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *HLT-NAACL*.

Yoshimasa Tsuruoka, Yusuke Miyao, and Jun'ichi Kazama. 2011. Learning with lookahead: Can history-based models rival globally optimized models? In *CoNLL*.

Beilun Wang, Ji Gao, and Yanjun Qi. 2017. A theoretical framework for robustness of (deep) classifiers against adversarial samples. In *ICLR*.

Yi Wu, David Bamman, and Stuart Russell. 2017. Adversarial training for relation extraction. In *EMNLP*.

Ziang Xie, Sida I Wang, Jiwei Li, Daniel Levy, Aiming Nie, Dan Jurafsky, and Andrew Y Ng. 2017. Data noising as smoothing in neural network language models. In *ICLR*.

Zhilin Yang, Ruslan Salakhutdinov, and William W. Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. In *ICLR*.

Universal Dependency Parsing for Hindi-English Code-switching

Irshad Ahmad Bhat
LTRC, IIIT-H,
Hyderabad, India
irshad.bhat@iiit.ac.in

Riyaz Ahmad Bhat
Interaction Labs,
Bangalore, India
rbhat@interactions.com

Manish Shrivastava
LTRC, IIIT-H,
Hyderabad, India
m.shrivastava@iiit.ac.in

Dipti Misra Sharma
LTRC, IIIT-H,
Hyderabad, India
dipti@iiit.ac.in

Abstract

Code-switching is a phenomenon of mixing grammatical structures of two or more languages under varied social constraints. The code-switching data differ so radically from the benchmark corpora used in NLP community that the application of standard technologies to these data degrades their performance sharply. Unlike standard corpora, these data often need to go through additional processes such as language identification, normalization and/or back-transliteration for their efficient processing. In this paper, we investigate these indispensable processes and other problems associated with syntactic parsing of code-switching data and propose methods to mitigate their effects. In particular, we study dependency parsing of code-switching data of Hindi and English multilingual speakers from Twitter. We present a treebank of Hindi-English code-switching tweets under Universal Dependencies scheme and propose a neural stacking model for parsing that efficiently leverages part-of-speech tag and syntactic tree annotations in the code-switching treebank and the preexisting Hindi and English treebanks. We also present normalization and back-transliteration models with a decoding process tailored for code-switching data. Results show that our neural stacking parser is 1.5% LAS points better than the augmented parsing model and our decoding process improves results by 3.8% LAS points over the first-best normalization and/or back-transliteration.

1 Introduction

Code-switching[1] (henceforth CS) is the juxtaposition, within the same speech utterance, of grammatical units such as words, phrases, and clauses belonging to two or more different languages (Gumperz, 1982). The phenomenon is prevalent in multilingual societies where speakers share more than one language and is often prompted by multiple social factors (Myers-Scotton, 1995). Moreover, code-switching is mostly prominent in colloquial language use in daily conversations, both online and offline.

Most of the benchmark corpora used in NLP for training and evaluation are based on edited monolingual texts which strictly adhere to the norms of a language related, for example, to orthography, morphology, and syntax. Social media data in general and CS data, in particular, deviate from these norms implicitly set forth by the choice of corpora used in the community. This is the reason why the current technologies often perform miserably on social media data, be it monolingual or mixed language data (Solorio and Liu, 2008b; Vyas et al., 2014; Çetinoğlu et al., 2016; Gimpel et al., 2011; Owoputi et al., 2013; Kong et al., 2014). CS data offers additional challenges over the monolingual social media data as the phenomenon of code-switching transforms the data in many ways, for example, by creating new lexical forms and syntactic structures by mixing morphology and syntax of two languages making it much more diverse than any monolingual corpora (Çetinoğlu et al., 2016). As the current computational models fail to cater to the complexities of CS data, there is often a need for dedicated techniques tailored to its specific characteristics.

Given the peculiar nature of CS data, it has been widely studied in linguistics literature (Poplack, 1980; Gumperz, 1982; Myers-Scotton, 1995), and more recently, there has been a surge in studies concerning CS data in NLP as well (Solorio and Liu, 2008a,a; Vyas et al., 2014; Sharma et al., 2016; Rudra et al., 2016; Joshi et al., 2016; Bhat et al., 2017; Chandu et al., 2017; Rijhwani et al.,

[1] Code-mixing is another term in the linguistics literature used interchangeably with code-switching. Both terms are often used to refer to the same or similar phenomenon of mixed language use.

Proceedings of NAACL-HLT 2018, pages 987–998
New Orleans, Louisiana, June 1 - 6, 2018.

2017; Guzmán et al., 2017, and others). Besides the individual computational works, a series of shared-tasks and workshops on preprocessing and shallow syntactic analysis of CS data have also been conducted at multiple venues such as Empirical Methods in NLP (EMNLP 2014 and 2016), International Conference on NLP (ICON 2015 and 2016) and Forum for Information Retrieval Evaluation (FIRE 2015 and 2016). Most of these works have attempted to address preliminary tasks such as language identification, normalization and/or back-transliteration as these data often need to go through these additional processes for their efficient processing. In this paper, we investigate these indispensable processes and other problems associated with syntactic parsing of code-switching data and propose methods to mitigate their effects. In particular, we study dependency parsing of Hindi-English code-switching data of multilingual Indian speakers from Twitter. Hindi-English code-switching presents an interesting scenario for the parsing community. Mixing among typologically diverse languages will intensify structural variations which will make parsing more challenging. For example, there will be many sentences containing: (1) both SOV and SVO word orders[2], (2) both head-initial and head-final genitives, (3) both prepositional and postpositional phrases, etc. More importantly, none among the Hindi and English treebanks would provide any training instance for these mixed structures within individual sentences. In this paper, we present the first code-switching treebank that provides syntactic annotations required for parsing mixed-grammar syntactic structures. Moreover, we present a parsing pipeline designed explicitly for Hindi-English CS data. The pipeline comprises of several modules such as a language identification system, a back-transliteration system, and a dependency parser. The gist of these modules and our overall research contributions are listed as follows:

- back-transliteration and normalization models based on encoder-decoder frameworks with sentence decoding tailored for code-switching data;
- a dependency treebank of Hindi-English code-switching tweets under Universal Dependencies scheme; and
- a neural parsing model which learns POS tagging and parsing jointly and also incorporates knowledge from the monolingual treebanks using neural stacking.

[2]Order of Subject, Object and Verb in transitive sentences.

2 Preliminary Tasks

As preliminary steps before parsing of CS data, we need to identify the language of tokens and normalize and/or back-transliterate them to enhance the parsing performance. These steps are indispensable for processing CS data and without them the performance drops drastically as we will see in Results Section. We need normalization of non-standard word forms and back-transliteration of Romanized Hindi words for addressing out-of-vocabulary problem, and lexical and syntactic ambiguity introduced due to contracted word forms. As we will train separate normalization and back-transliteration models for Hindi and English, we need language identification for selecting which model to use for inference for each word form separately. Moreover, we also need language information for decoding best word sequences.

2.1 Language Identification

For language identification task, we train a multilayer perceptron (MLP) stacked on top of a recurrent bidirectional LSTM (Bi-LSTM) network as shown in Figure 1.

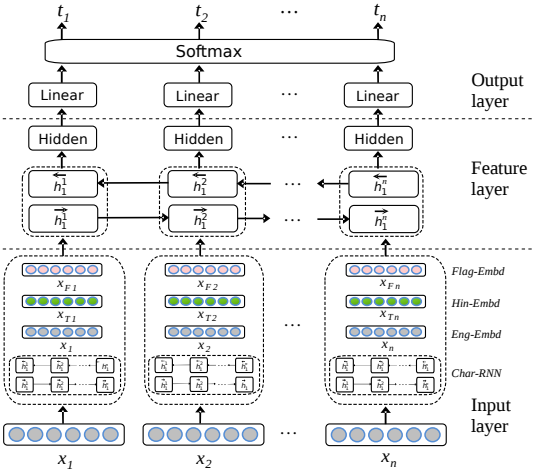


Figure 1: Language identification network

We represent each token by a concatenated vector of its English embedding, back-transliterated Hindi embedding, character Bi-LSTM embedding and flag embedding (English dictionary flag and word length flag with length bins of 0-3, 4-6, 7-10, and 10-all). These concatenated vectors are passed to a Bi-LSTM network to generate a sequence of

hidden representations which encode the contextual information spread across the sentence. Finally, output layer uses the feed-forward neural network with a softmax function for a probability distribution over the language tags. We train the network on our CS training set concatenated with the data set provided in ICON 2015[3] shared task (728 Facebook comments) on language identification and evaluate it on the datasets from Bhat et al. (2017). We achieved the state-of-the-art performance on both development and test sets (Bhat et al., 2017). The results are shown in Table 1.

Label	Precision	Recall	F1-Score	count
hi	97.76	98.09	97.92	1465
en	96.87	98.83	97.84	1283
ne	94.33	79.17	86.08	168
acro	92.00	76.67	83.64	30
univ	99.71	1.00	99.86	349
average	97.39	97.42	97.36	3295
(Bhat et al., 2017)	-	96.10	-	-

Table 1: Language Identification results on CS test set.

2.2 Normalization and Back-transliteration

We learn two separate but similar character-level models for normalization-cum-transliteration of noisy Romanized Hindi words and normalization of noisy English words. We treat both normalization and back-transliteration problems as a general sequence to sequence learning problem. In general, our goal is to learn a mapping for non-standard English and Romanized Hindi word forms to standard forms in their respective scripts. In case of Hindi, we address the problem of normalization and back-transliteration of Romanized Hindi words using a single model. We use the attention-based encoder-decoder model of Luong (Luong et al., 2015) with global attention for learning. For Hindi, we train the model on the transliteration pairs (87,520) from the Libindic transliteration project[4] and Brahmi-Net (Kunchukuttan et al., 2015) which are further augmented with noisy transliteration pairs (1,75,668) for normalization. Similarly, for normalization of noisy English words, we train the model on noisy word forms (4,29,715) synthetically generated from the English vocabulary. We use simple rules such as dropping non-initial vowels and replacing consonants based on their phonological proximity to generate synthetic data for normalization. Figure 2 shows some of the noisy forms generated from standard word forms using simple and finite rules which include vowel elision (`please` → `pls`), interchanging similar consonants and vowels (`cousin` → `couzin`), replacing consonant or vowel clusters with a single letter (`Twitter` → `Twiter`), etc. From here onwards, we will refer to both normalization and back-transliteration as normalization.

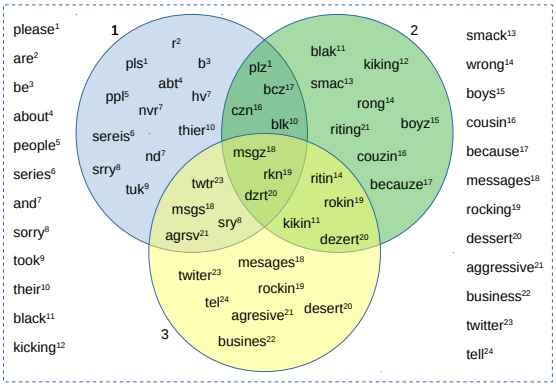


Figure 2: Synthetic normalization pairs generated for a sample of English words using hand crafted rules.

At inference time, our normalization models will predict the most likely word form for each input word. However, the single-best output from the model may not always be the best option considering an overall sentential context. Contracted word forms in social media content are quite often ambiguous and can represent different standard word forms. For example, noisy form '`pt`' can expand to different standard word forms such as '`put`', '`pit`', '`pat`', '`pot`' and '`pet`'. The choice of word selection will solely depend on the sentential context. To select contextually relevant forms, we use exact search over n-best normalizations from the respective models extracted using beam-search decoding. The best word sequence is selected using the Viterbi decoding over b^n word sequences scored by a trigram language model. b is the size of beam-width and n is the sentence length. The language models are trained on the monolingual data of Hindi and English using KenLM toolkit (Heafield et al., 2013). For each word, we extract five best normalizations (b=5). Decoding the best word sequence is a non-trivial problem for CS data due to lack of normalized and back-transliterated CS data for training a language model. One obvious solution is to apply decoding on individual language fragments in a CS sentence (Dutta et al., 2015). One major prob-

[3] http://ltrc.iiit.ac.in/icon2015/

[4] https://github.com/libindic/indic-trans

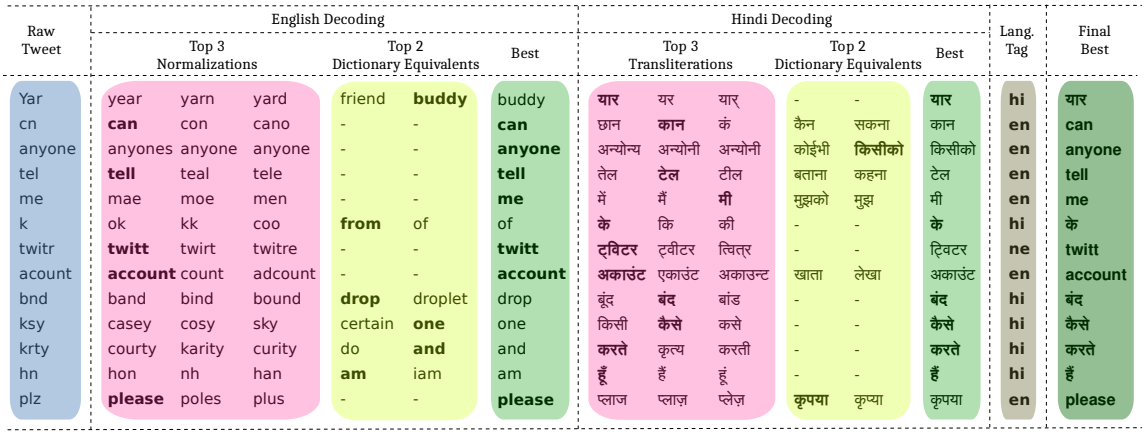


Figure 3: The figure shows a 3-step decoding process for the sentence *"Yar cn anyone tel me k twitr account bnd ksy krty hn plz" (Friend can anyone tell me how to close twitter account please).*

lem with this approach is that the language models used for scoring are trained on complete sentences but are applied on sentence fragments. Scoring individual CS fragments might often lead to wrong word selection due to incomplete context, particularly at fragment peripheries. We solve this problem by using a 3-step decoding process that works on two separate versions of a CS sentence, one in Hindi, and one in English. In the first step, we replace first-best back-transliterated forms of Hindi words by their translation equivalents using a Hindi-English bilingual lexicon.[5] An exact search is used over the top '5' normalizations of English words, the translation equivalents of Hindi words and the actual word itself. In the second step, we decode best word sequence over Hindi version of the sentence by replacing best English word forms decoded from the first step by their translation equivalents. An exact search is used over the top '5' normalizations of Hindi words, the dictionary equivalents of decoded English words and the original words. In the final step, English and Hindi words are selected from their respective decoded sequences using the predicted language tags from the language identification system. Note that the bilingual mappings are only used to aid the decoding process by making the CS sentences lexically monolingual so that the monolingual language models could be used for scoring. They are not used in the final decoded output. The overall decoding process is shown in Figure 3.

Both of our normalization and back-transliteration systems are evaluated on the evaluation set of Bhat et al. (2017). Results of our systems are reported in Table 3 with a comparison of accuracies based on the nature of decoding used. The results clearly show the significance of our 3-step decoding over first-best and fragment-wise decoding.

Data-set	Hindi				English			
	Tokens	FB	FW	3-step	Tokens	FB	FW	3-step
Dev	1549	82.82	87.28	**90.01**	34	82.35	**88.23**	**88.23**
Test	1465	83.54	88.19	**90.64**	28	71.42	75.21	**81.71**

Table 2: Normalization accuracy based on the number of noisy tokens in the evaluation set. FB = First Best, and FW = Fragment Wise

3 Universal Dependencies for Hindi-English

Recently Bhat et al. (2017) provided a CS dataset for the evaluation of their parsing models which they trained on the Hindi and English Universal Dependency (UD) treebanks. We extend this dataset by annotating 1,448 more sentences. Following Bhat et al. (2017) we first sampled CS data from a large set of tweets of Indian language users that we crawled from Twitter using Tweepy[6]–a Twitter API wrapper. We then used a language identification system trained on ICON dataset (see Section 2) to filter Hindi-English CS tweets from the crawled Twitter data. Only those tweets were selected that satisfied a minimum ratio of 30:70(%) code-switching. From this dataset, we manually selected 1,448 tweets for annotation. The selected tweets are thoroughly checked for code-switching ratio. For POS tagging and dependency annotation, we used Version 2 of Universal dependency guidelines (De Marneffe et al., 2014),

[5] An off-the-shelf MT system would have been appropriate for this task, however, we would first need to adapt it to CS data which in itself is a non-trivial task.

[6] http://www.tweepy.org/

while language tags are assigned based on the tag set defined in (Solorio et al., 2014; Jamatia et al., 2015). The dataset was annotated by two expert annotators who have been associated with annotation projects involving syntactic annotations for around 10 years. Nonetheless, we also ensured the quality of the manual annotations by carrying an inter-annotator agreement analysis. We randomly selected a dataset of 150 tweets which were annotated by both annotators for both POS tagging and dependency structures. The inter-annotator agreement has a 96.20% accuracy for POS tagging and a 95.94% UAS and a 92.65% LAS for dependency parsing.

We use our dataset for training while the development and evaluation sets from Bhat et al. (2017) are used for tuning and evaluation of our models. Since the annotations in these datasets follow version 1.4 of the UD guidelines, we converted them to version 2 by using carefully designed rules. The statistics about the data are given in Table 3.

Data-set	Sentences	Tokens	Hi	En	Ne	Univ	Acro
Train	1,448	20,203	8,363	8,270	698	2,730	142
Dev	225	3,411	1,549	1,300	151	379	32
Test	225	3,295	1,465	1,283	168	349	30

Table 3: Data Statistics. Dev set is used for tuning model parameters, while Test set is used for evaluation.

4 Dependency Parsing

We adapt Kiperwasser and Goldberg (2016) transition-based parser as our base model and incorporate POS tag and monolingual parse tree information into the model using neural stacking, as shown in Figures 4 and 6.

4.1 Parsing Algorithm

Our parsing models are based on an arc-eager transition system (Nivre, 2003). The arc-eager system defines a set of configurations for a sentence $w_1, \ldots, w_n$, where each configuration $c = (S, B, A)$ consists of a stack S, a buffer B, and a set of dependency arcs A. For each sentence, the parser starts with an initial configuration where $S = [ROOT]$, $B = [w_1, \ldots, w_n]$ and $A = \emptyset$ and terminates with a configuration c if the buffer is empty and the stack contains the ROOT. The parse trees derived from transition sequences are given by A. To derive the parse tree, the arc-eager system defines four types of transitions (t): Shift, Left-Arc, Right-Arc, and Reduce.

We use the training by exploration method of Goldberg and Nivre (2012) for decoding a transition sequence which helps in mitigating error propagation at evaluation time. We also use pseudo-projective transformations of Nivre and Nilsson (2005) to handle a higher percentage of non-projective arcs in the CS data (∼2%). We use the most informative scheme of head+path to store the transformation information.

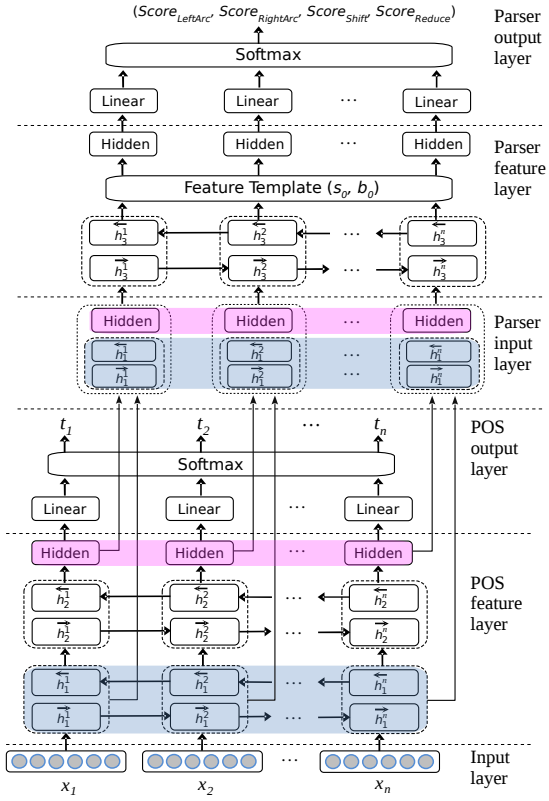


Figure 4: POS tagging and parsing network based on stack-propagation model proposed in (Zhang and Weiss, 2016).

4.2 Base Models

Our base model is a stack of a tagger network and a parser network inspired by stack-propagation model of Zhang and Weiss (2016). The parameters of the tagger network are shared and act as a regularization on the parsing model. The model is trained by minimizing a joint negative log-likelihood loss for both tasks. Unlike Zhang and Weiss (2016), we compute the gradients of the log-loss function simultaneously for each training instance. While the parser network is updated given the parsing loss only, the tagger network is updated with respect to both tagging and parsing losses. Both tagger and parser networks comprise of an input layer, a feature layer, and an output layer as shown in Figure 4. Following Zhang and Weiss (2016), we refer to this model as stack-prop.

Tagger network: The input layer of the tagger encodes each input word in a sentence by concatenating a pre-trained word embedding with its character embedding given by a character Bi-LSTM. In the feature layer, the concatenated word and character representations are passed through two stacked Bi-LSTMs to generate a sequence of hidden representations which encode the contextual information spread across the sentence. The first Bi-LSTM is shared with the parser network while the other is specific to the tagger. Finally, output layer uses the feed-forward neural network with a softmax function for a probability distribution over the Universal POS tags. We only use the forward and backward hidden representations of the focus word for classification.

Parser Network: Similar to the tagger network, the input layer encodes the input sentence using word and character embeddings which are then passed to the shared Bi-LSTM. The hidden representations from the shared Bi-LSTM are then concatenated with the dense representations from the feed-forward network of the tagger and passed through the Bi-LSTM specific to the parser. This ensures that the tagging network is penalized for the parsing error caused by error propagation by back-propagating the gradients to the shared tagger parameters (Zhang and Weiss, 2016). Finally, we use a non-linear feed-forward network to predict the labeled transitions for the parser configurations. From each parser configuration, we extract the top node in the stack and the first node in the buffer and use their hidden representations from the parser specific Bi-LSTM for classification.

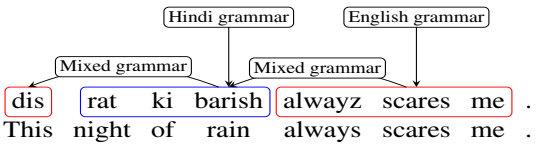


Figure 5: Code-switching tweet showing grammatical fragments from Hindi and English.

4.3 Stacking Models

It seems reasonable that limited CS data would complement large monolingual data in parsing CS data and a parsing model which leverages both data would significantly improve parsing performance. While a parsing model trained on our limited CS data might not be enough to accurately parse the individual grammatical fragments of Hindi and English, the preexisting Hindi and English treebanks are large enough to provide sufficient annotations to capture their structure. Similarly, parsing model(s) trained on the Hindi and English data may not be able to properly connect the divergent fragments of the two languages as the model lacks evidence for such mixed structures in the monolingual data. This will happen quite often as Hindi and English are typologicalls very diverse (see Figure 5).

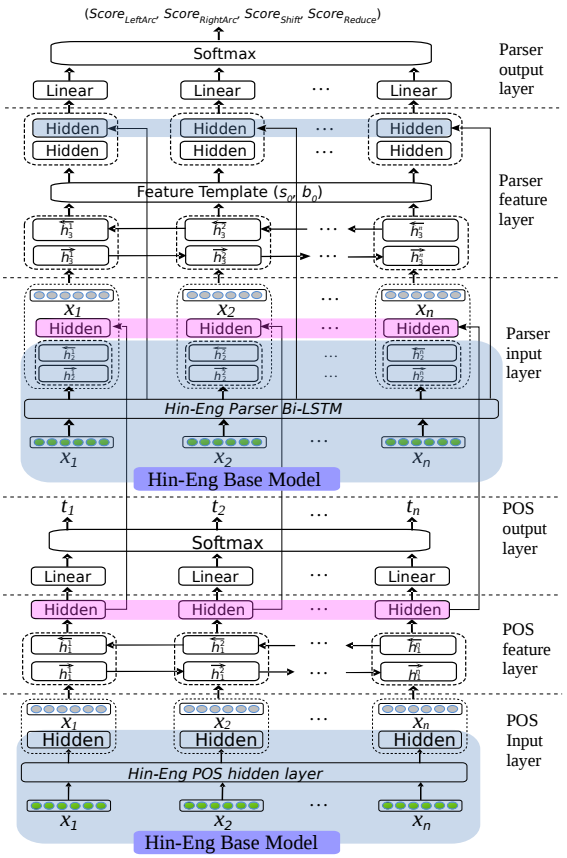


Figure 6: Neural Stacking-based parsing architecture for incorporating monolingual syntactic knowledge.

As we discussed above, we adapted feature-level neural stacking (Zhang and Weiss, 2016; Chen et al., 2016) for joint learning of POS tagging and parsing. Similarly, we also adapt this stacking approach for incorporating the monolingual syntactic knowledge into the base CS model. Recently, Wang et al. (2017) used neural stacking for injecting syntactic knowledge of English into a graph-based Singlish parser which lead to significant improvements in parsing performance. Unlike Wang et al. (2017), our base stacked models will allow us to transfer the POS tagging knowledge as well along the parse tree knowledge.

As shown in Figure 6, we transfer both POS tagging and parsing information from the source

model trained on augmented Hindi and English data. For tagging, we augment the input layer of the CS tagger with the MLP layer of the source tagger. For transferring parsing knowledge, hidden representations from the parser specific Bi-LSTM of the source parser are augmented with the input layer of the CS parser which already includes the hidden layer of the CS tagger, word and character embeddings. In addition, we also add the MLP layer of the source parser to the MLP layer of the CS parser. The MLP layers of the source parser are generated using raw features from CS parser configurations. Apart from the addition of these learned representations from the source model, the overall CS model remains similar to the base model shown in Figure 4. The tagging and parsing losses are back-propagated by traversing back the forward paths to all trainable parameters in the entire network for training and the whole network is used collectively for inference.

5 Experiments

We train all of our POS tagging and parsing models on training sets of the Hindi and English UD-v2 treebanks and our Hindi-English CS treebank. For tuning and evaluation, we use the development and evaluation sets from Bhat et al. (2017). We conduct multiple experiments in gold and predicted settings to measure the effectiveness of the sub-modules of our parsing pipeline. In predicted settings, we use the POS taggers separately trained on the Hindi, English and CS training sets. All of our models use word embeddings from transformed Hindi and English embedding spaces to address the problem of lexical differences prevalent in CS sentences.

5.1 Hyperparameters

Word Representations For language identification, POS tagging and parsing models, we include the lexical features in the input layer of our neural networks using 64-dimension pre-trained word embeddings, while we use randomly initialized embeddings within a range of $[-0.1, +0.1]$ for non-lexical units such as POS tags and dictionary flags. We use 32-dimensional character embeddings for all the three models and 32-dimensional POS tag embeddings for pipelined parsing models. The distributed representation of Hindi and English vocabulary are learned separately from the Hindi and English monolingual corpora. The English monolingual data contains around 280M sentences, while the Hindi data is comparatively smaller and contains around 40M sentences. The word representations are learned using Skip-gram model with negative sampling which is implemented in `word2vec` toolkit (Mikolov et al., 2013). We use the projection algorithm of Artetxe et al. (2016) to transform the Hindi and English monolingual embeddings into same semantic space using a bilingual lexicon ($\sim$63,000 entries). The bilingual lexicon is extracted from ILCI and Bojar Hindi-English parallel corpora (Jha, 2010; Bojar et al., 2014). For normalization models, we use 32-dimensional character embeddings uniformly initialized within a range of $[-0.1, +0.1]$.

Hidden dimensions The POS tagger specific Bi-LSTMs have 128 cells while the parser specific Bi-LSTMs have 256 cells. The Bi-LSTM in the language identification model has 64 cells. The character Bi-LSTMs have 32 cells for all three models. The hidden layer of MLP has 64 nodes for the language identification network, 128 nodes for the POS tagger and 256 nodes for the parser. We use hyperbolic tangent as an activation function in all tasks. In the normalization models, we use single layered Bi-LSTMs with 512 cells for both encoding and decoding of character sequences.

Learning For language identification, POS tagging and parsing networks, we use momentum SGD for learning with a minibatch size of 1. The LSTM weights are initialized with random orthonormal matrices as described in (Saxe et al., 2013). We set the dropout rate to 30% for POS tagger and parser Bi-LSTM and MLP hidden states while for language identification network we set the dropout to 50%. All three models are trained for up to 100 epochs, with early stopping based on the development set.

In case of normalization, we train our encoder-decoder models for 25 epochs using vanilla SGD. We start with a learning rate of 1.0 and after 8 epochs reduce it to half for every epoch. We use a mini-batch size of 128, and the normalized gradient is rescaled whenever its norm exceeds 5. We use a dropout rate of 30% for the Bi-LSTM.

Language identification, POS tagging and parsing code is implemented in DyNet (Neubig et al., 2017) and for normalization without decoding, we use Open-NMT toolkit for neural machine translation (Klein et al., 2017). All

the code is available at https://github.com/irshadbhat/nsdp-cs and the data is available at https://github.com/CodeMixedUniversalDependencies/UD_Hindi_English.

6 Results

In Table 4, we present the results of our main model that uses neural stacking for learning POS tagging and parsing and also for knowledge transfer from the Bilingual model. Transferring POS tagging and syntactic knowledge using neural stacking gives 1.5% LAS[7] improvement over a naive approach of data augmentation. The Bilingual model which is trained on the union of Hindi and English data sets is least accurate of all our parsing models. However, it achieves better or near state-of-the-art results on the Hindi and English evaluation sets (see Table 5). As compared to the best system in CoNLL 2017 Shared Task on Universal Dependencies (Zeman et al., 2017; Dozat et al., 2017), our results for English are around 3% better in LAS, while for Hindi only 0.5% LAS points worse. The CS model trained only on the CS training data is slightly more accurate than the Bilingual model. Augmenting the CS data to Hindi-English data complements their syntactic structures relevant for parsing mixed grammar structures which are otherwise missing in the individual datasets. The average improvements of around ∼5% LAS clearly show their complementary nature.

Model	Gold (LID+TRN)		Auto (LID+TRN)	
	UAS	LAS	UAS	LAS
Bilingual	75.26	65.41	73.29	63.18
CS	76.69	66.90	75.84	64.94
Augmented	80.39	71.27	78.95	69.51
Neural Stacking	**81.50**	**72.44**	**80.23**	**71.03**
(Bhat et al., 2017)	74.16	64.11	66.18	54.40

Table 4: Accuracy of different parsing models on the evaluation set. POS tags are jointly predicted with parsing. LID = Language tag, TRN = Transliteration/normalization.

Table 6 summarizes the POS tagging results on the CS evaluation set. The tagger trained on the CS training data is 2.5% better than the Bilingual tagger. Adding CS training data to Hindi and English train sets further improves the accuracy by 1%. However, our stack-prop tagger achieves the highest accuracy of 90.53% by leveraging POS information from Bilingual tagger using neural stacking.

[7]The improvements discussed in the running text are for the models that are evaluated in auto settings.

Data-set	Pipeline					Stack-prop		
	Gold POS		Auto POS					
	UAS	LAS	POS	UAS	LAS	POS	UAS	LAS
Hindi	95.66	93.08	97.52	94.08	90.69	**97.65**	**94.36**	**91.02**
English	89.95	87.96	95.75	87.71	84.59	**95.80**	**88.30**	**85.30**

Table 5: POS and parsing results for Hindi and English monolingual test sets using pipeline and stack-prop models.

Model	Gold (LID+TRN)		Auto (LID+TRN)	
	Pipeline	SP	Pipeline	SP
Bilingual	88.36	88.12	86.71	86.27
CS	90.32	90.38	89.12	89.19
Augmented	91.20	91.50	90.02	90.20
Neural Stacking	**91.76**	**91.90**	**90.36**	**90.53**
(Bhat et al., 2017)	86.00		85.30	

Table 6: POS tagging accuracies of different models on CS evaluation set. SP = stack-prop.

Pipeline vs Stack-prop Table 7 summarizes the parsing results of our pipeline models which use predicted POS tags as input features. As compared to our stack-prop models (Table 4), pipeline models are less accurate (average 1% LAS improvement across models) which clearly emphasizes the significance of back-propagating the parsing loss to tagging parameters as well.

Model	Gold (LID+TRN+POS)		Auto (LID+TRN+POS)	
	UAS	LAS	UAS	LAS
Bilingual	82.29	73.79	72.09	61.18
CS	82.73	73.38	75.20	64.64
Augmented	85.66	77.75	77.98	69.16
Neural Stacking	**86.87**	**78.57**	**78.90**	**69.45**

Table 7: Accuracy of different parsing models on the test set using predicted language tags, normalized/back-transliterated words and predicted POS tags. POS tags are predicted separately before parsing. In Neural Stacking model, only parsing knowledge from the Bilingual model is transferred.

Significance of normalization We also conducted experiments to evaluate the impact of normalization on both POS tagging and parsing. The results are shown in Table 8. As expected, tagging and parsing models that use normalization without decoding achieve an average of 1% improvement over the models that do not use normalization at all. However, our 3-step decoding leads to higher gains in tagging as well as parsing accuracies. We achieved around 2.8% improvements in tagging and around 4.6% in parsing over the models that use first-best word forms from the normalization models. More importantly, there is a mod-

erate drop in accuracy (1.4% LAS points) caused due to normalization errors (see results in Table 4 for gold vs auto normalization).

System	POS	UAS	LAS
No Normalization	86.98	76.25	66.02
First Best	87.74	78.26	67.22
3-step Decoding	**90.53**	**80.23**	**71.03**

Table 8: Impact of normalization and back-transliteration on POS tagging and parsing models.

Monolingual vs Cross-lingual Embeddings We also conducted experiments with monolingual and cross-lingual embeddings to evaluate the need for transforming the monolingual embeddings into a same semantic space for processing of CS data. Results are shown in Table 9. Cross-lingual embeddings have brought around ∼0.5% improvements in both tagging and parsing. Cross-lingual embeddings are essential for removing lexical differences which is one of the problems encountered in CS data. Addressing the lexical differences will help in better learning by exposing syntactic similarities between languages.

Embedding	POS	UAS	LAS
Monolingual	90.07	79.46	70.53
Crosslingual	**90.53**	**80.23**	**71.03**

Table 9: Impact of monolingual and cross-lingual embeddings on stacking model performance.

7 Conclusion

In this paper, we have presented a dependency parser designed explicitly for Hindi-English CS data. The parser uses neural stacking architecture of Zhang and Weiss (2016) and Chen et al. (2016) for learning POS tagging and parsing and for knowledge transfer from Bilingual models trained on Hindi and English UD treebanks. We have also presented normalization and back-transliteration models with a decoding process tailored for CS data. Our neural stacking parser is 1.5% LAS points better than the augmented parsing model and 3.8% LAS points better than the one which uses first-best normalizations.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 2289–2294.

Irshad Bhat, Riyaz A. Bhat, Manish Shrivastava, and Dipti Sharma. 2017. Joining hands: Exploiting monolingual treebanks for parsing of code-mixing data. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, Valencia, Spain, pages 324–330.

Ondřej Bojar, Vojtěch Diatka, Pavel Rychlý, Pavel Straňák, Vít Suchomel, Aleš Tamchyna, and Daniel Zeman. 2014. HindEnCorp - Hindi-English and Hindi-only Corpus for Machine Translation. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. European Language Resources Association (ELRA), Reykjavik, Iceland.

Özlem Çetinoğlu, Sarah Schulz, and Ngoc Thang Vu. 2016. Challenges of computational processing of code-switching. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*. Association for Computational Linguistics, Austin, Texas, pages 1–11.

Khyathi Raghavi Chandu, Manoj Chinnakotla, Alan W Black, and Manish Shrivastava. 2017. Webshodh: A code mixed factoid question answering system for web. In *International Conference of the Cross-Language Evaluation Forum for European Languages*. Springer, pages 104–111.

Hongshen Chen, Yue Zhang, and Qun Liu. 2016. Neural network for heterogeneous annotations. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 731–741.

Marie-Catherine De Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D Manning. 2014. Universal stanford dependencies: A cross-linguistic typology. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*. volume 14, pages 4585–92.

Timothy Dozat, Peng Qi, and Christopher D Manning. 2017. Stanford's graph-based neural dependency parser at the conll 2017 shared task. *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies* pages 20–30.

Sukanya Dutta, Tista Saha, Somnath Banerjee, and Sudip Kumar Naskar. 2015. Text normalization in code-mixed social media text. In *Proceedings of the*

2nd International Conference on Recent Trends in Information Systems (ReTIS).

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*. Association for Computational Linguistics, pages 42–47.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *Proceedings of the 24th International Conference on Computational Linguistics*. pages 959–976.

John J Gumperz. 1982. *Discourse strategies*, volume 1. Cambridge University Press.

Gualberto Guzmán, Joseph Ricard, Jacqueline Serigos, Barbara E Bullock, and Almeida Jacqueline Toribio. 2017. Metrics for modeling code-switching across corpora. *Proceedings of Interspeech 2017* pages 67–71.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H Clark, and Philipp Koehn. 2013. Scalable modified kneser-ney language model estimation. In *ACL (2)*. pages 690–696.

Anupam Jamatia, Björn Gambäck, and Amitava Das. 2015. Part-of-speech tagging for code-mixed english-hindi twitter and facebook chat messages. page 239.

Girish Nath Jha. 2010. The TDIL program and the Indian language corpora initiative (ILCI). In *Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC 2010). European Language Resources Association (ELRA)*.

Aditya Joshi, Ameya Prabhu, Manish Shrivastava, and Vasudeva Varma. 2016. Towards sub-word level compositions for sentiment analysis of hindi-english code mixed text. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 2482–2491.

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

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Proc. ACL*. https://doi.org/10.18653/v1/P17-4012.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, and Chris Dyer. 2014. A dependency parser for tweets. In *Proceedings of Conference on Empirical Methods In Natural Language Processing (EMNLP)*. volume 1001, page 1012.

Anoop Kunchukuttan, Ratish Puduppully, and Pushpak Bhattacharyya. 2015. Brahmi-net: A transliteration and script conversion system for languages of the indian subcontinent.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* .

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Carol Myers-Scotton. 1995. *Social motivations for codeswitching: Evidence from Africa*. Oxford University Press.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980* .

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*.

Olutobi Owoputi, Brendan OConnor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of NAACL-HLT*. pages 380–390.

Shana Poplack. 1980. Sometimes ill start a sentence in spanish y termino en español: toward a typology of code-switching1. *Linguistics* 18(7-8):581–618.

Shruti Rijhwani, Royal Sequiera, Monojit Choudhury, Kalika Bali, and Chandra Shekhar Maddila. 2017. Estimating code-switching on twitter with a novel generalized word-level language detection technique. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 1971–1982.

Koustav Rudra, Shruti Rijhwani, Rafiya Begum, Kalika Bali, Monojit Choudhury, and Niloy Ganguly.

2016. Understanding language preference for expression of opinion and sentiment: What do hindi-english speakers do on twitter? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1131–1141. https://aclweb.org/anthology/D16-1121.

Andrew M Saxe, James L McClelland, and Surya Ganguli. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120* .

Arnav Sharma, Sakshi Gupta, Raveesh Motlani, Piyush Bansal, Manish Shrivastava, Radhika Mamidi, and Dipti M. Sharma. 2016. Shallow parsing pipeline - hindi-english code-mixed social media text. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 1340–1345.

Thamar Solorio, Elizabeth Blair, Suraj Maharjan, Steve Bethard, Mona Diab, Mahmoud Gonheim, Abdelati Hawwari, Fahad AlGhamdi, Julia Hirshberg, Alison Chang, and Pascale Fung. 2014. Overview for the first shared task on language identification in code-switched data. In *Proceedings of the First Workshop on Computational Approaches to Code-Switching. EMNLP 2014, Conference on Empirical Methods in Natural Language Processing, Octobe, 2014, Doha, Qatar*.

Thamar Solorio and Yang Liu. 2008a. Learning to predict code-switching points. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 973–981.

Thamar Solorio and Yang Liu. 2008b. Part-of-speech tagging for english-spanish code-switched text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1051–1060.

Yogarshi Vyas, Spandana Gella, Jatin Sharma, Kalika Bali, and Monojit Choudhury. 2014. Pos tagging of english-hindi code-mixed social media content. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. volume 14, pages 974–979.

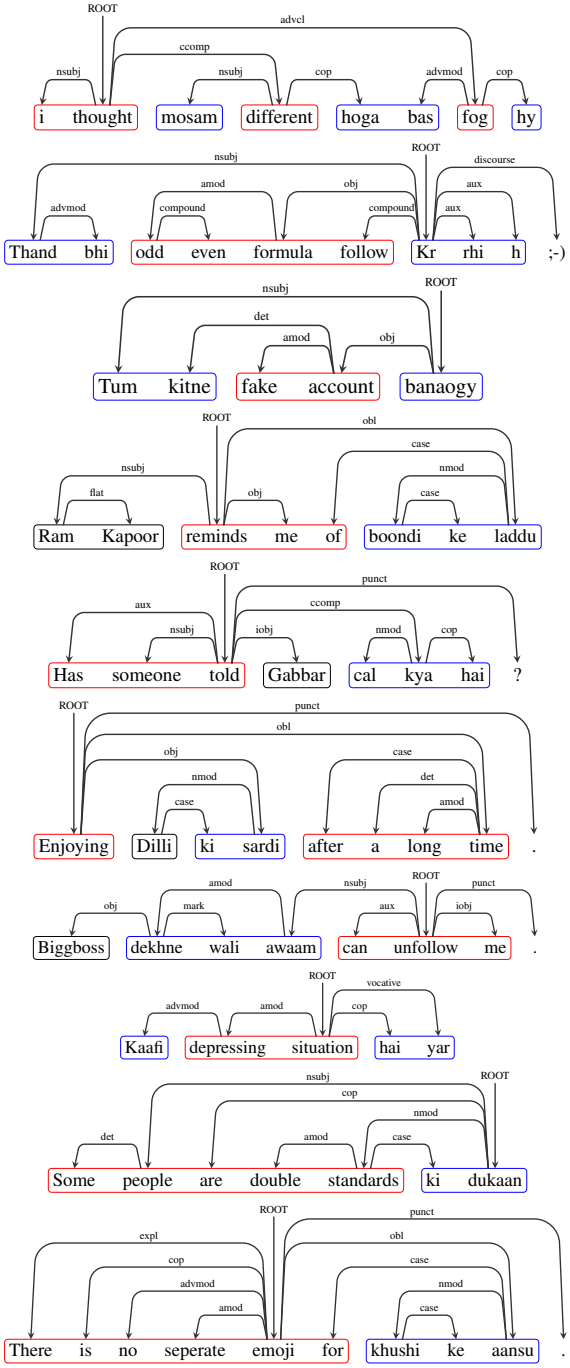
Hongmin Wang, Yue Zhang, GuangYong Leonard Chan, Jie Yang, and Hai Leong Chieu. 2017. Universal dependencies parsing for colloquial singaporean english. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1732–1744.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gokirmak, Anna Nedoluzhko, Silvie Cinkova, Jan Hajic jr., Jaroslava Hlavacova, Václava Kettnerová, Zdenka Uresova, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria dePaiva, Kira Droganova, Héctor Martínez Alonso, Çağr Çöltekin, Umut Sulubacak, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadová, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonca, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 2017. Conll 2017 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics, Vancouver, Canada, pages 1–19.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1557–1566.

A Supplemental Material

A.1 Example Annotations from our CS Treebank



What's Going On in Neural Constituency Parsers? An Analysis

David Gaddy, Mitchell Stern, and Dan Klein
Computer Science Division
University of California, Berkeley
{dgaddy,mitchell,klein}@berkeley.edu

Abstract

A number of differences have emerged between modern and classic approaches to constituency parsing in recent years, with structural components like grammars and feature-rich lexicons becoming less central while recurrent neural network representations rise in popularity. The goal of this work is to analyze the extent to which information provided directly by the model structure in classical systems is still being captured by neural methods. To this end, we propose a high-performance neural model (92.08 F1 on PTB) that is representative of recent work and perform a series of investigative experiments. We find that our model implicitly learns to encode much of the same information that was explicitly provided by grammars and lexicons in the past, indicating that this scaffolding can largely be subsumed by powerful general-purpose neural machinery.

1 Introduction

In the past several years, many aspects of constituency parsing and natural language processing in general have changed. Grammars, which were once the central component of many parsers, have played a continually decreasing role. Rich lexicons and handcrafted lexical features have become less common as well. On the other hand, recurrent neural networks have gained traction as a powerful and general purpose tool for representation. So far, not much has been shown about how neural networks are able to compensate for the removal of the structures used in past models. To gain insight, we introduce a parser that is representative of recent trends and analyze its learned representations to determine what information it captures and what is important for its strong performance.

Our parser is a natural extension of recent work in constituency parsing. We combine a common span representation based on recurrent neural networks with a novel, simplified scoring model. In addition, we replace the externally predicted part-of-speech tags used in some recent systems with character-level word representations. Our parser achieves a test F1 score of 92.08 on section 23 of the Penn Treebank, exceeding the performance of many other state-of-the-art models evaluated under comparable conditions. Section 2 describes our model in detail.

The remainder of the paper is focused on analysis. In Section 3, we look at the decline of grammars and output correlations. Past work in constituency parsing used context-free grammars with production rules governing adjacent labels (or more generally production-factored scores) to propagate information and capture correlations between output decisions (Collins, 1997; Charniak and Johnson, 2005; Petrov and Klein, 2007; Hall et al., 2014). Many recent parsers no longer have explicit grammar production rules, but still use information about other predictions, allowing them to capture output correlations (Dyer et al., 2016; Choe and Charniak, 2016). Beyond this, there are some parsers that use no context for bracket scoring and only include mild output correlations in the form of tree constraints (Cross and Huang, 2016b; Stern et al., 2017). In our experiments, we find that we can accurately predict parents from the representation given to a child. Since a simple classifier can predict the information provided by parent-child relations, this explains why the information no longer needs to be specified explicitly. We also show that we can completely remove output correlations from our model with a variant of our parser that makes independent span label decisions without any tree constraints while maintaining high F1 scores and mostly producing trees.

In Section 4, we look at lexical representations. In the past, parsers used a variety of cus-

Proceedings of NAACL-HLT 2018, pages 999–1010
New Orleans, Louisiana, June 1 - 6, 2018.

tom lexical representations, such as word shape features, prefixes, suffixes, and special tokens for categories like numerals (Klein and Manning, 2003; Petrov and Klein, 2007; Finkel et al., 2008). Character-level models have shown promise in parsing and other NLP tasks as a way to remove the complexity of these lexical features (Ballesteros et al., 2015; Ling et al., 2015b; Kim et al., 2016; Coavoux and Crabbé, 2017; Liu and Zhang, 2017). We compare the performance of character-level representations and externally predicted part-of-speech tags and show that these two sources of information seem to fill a similar role. We also perform experiments showing that the representations learned with character-level models contain information that was hand-specified in some other models.

Finally, in Section 5 we look at the surface context captured by recurrent neural networks. Many recent parsers use LSTMs, a popular type of recurrent neural network, to combine and summarize context for making decisions (Choe and Charniak, 2016; Cross and Huang, 2016a; Dyer et al., 2016; Stern et al., 2017). Before LSTMs became common in parsing, systems that included surface features used a fixed-size window around the fenceposts at each end of a span (Charniak and Johnson, 2005; Finkel et al., 2008; Hall et al., 2014; Durrett and Klein, 2015), and the inference procedure handled most of the propagation of information from the rest of the sentence. We perform experiments showing that LSTMs capture far-away surface context and that this information is important for our parser's performance. We also provide evidence that word order of the far-away context is important and that the amount of context alone does not account for all of the gains seen with LSTMs.

Overall, we find that the same sources of information that were effective for grammar-driven parsers are also captured by parsers based on recurrent neural networks.

2 Parsing Model

In this section, we propose a span-based parsing model that combines components from several recent neural architectures for constituency parsing and other natural language tasks. While this system is primarily introduced for the purpose of our analysis, it also performs well as a parser in its own right, exhibiting some gains over comparable work. Our model is in many respects similar to the chart parser of Stern et al. (2017), but features a number of simplifications and improvements.

2.1 Overview

Abstractly, our model consists of a single scoring function $s(i, j, \ell)$ that assigns a real-valued score to every label ℓ for each span (i, j) in an input sentence. We take the set of available labels to be the collection of all nonterminals and unary chains observed in the training data, treating the latter as atomic units. The score of a tree T is defined as a sum over internal nodes of labeled span scores:

$$s(T) = \sum_{(i,j,\ell) \in T} s(i, j, \ell).$$

We note that, in contrast with many other chart parsers, our model can directly score n-ary trees without the need for binarization or other tree transformations. Under this setup, the parsing problem is to find the tree with the highest score:

$$\hat{T} = \operatorname*{argmax}_{T} s(T).$$

Our concrete implementation of $s(i, j, \ell)$ can be broken down into three pieces: word representation, span representation, and label scoring. We discuss each of these in turn.

2.2 Word Representation

One popular way to represent words is the use of word embeddings. We have a separate embedding for each word type in the training vocabulary and map all unknown words at test time to a single `<UNK>` token. In addition to word embeddings, character-level representations have also been gaining traction in recent years, with common choices including recurrent, convolutional, or bag-of-n-gram representations. These alleviate the unknown word problem by working with smaller, more frequent units, and readily capture morphological information not directly accessible through word embeddings. Character LSTMs in particular have proved useful in constituency parsing (Coavoux and Crabbé, 2017), dependency parsing (Ballesteros et al., 2015), part-of-speech tagging (Ling et al., 2015a), named entity recognition (Lample et al., 2016), and machine translation (Ling et al., 2015b), making them a natural choice for our system. We obtain a character-level representation for a word by running it through a bidirectional character LSTM and concatenating the final forward and backward outputs.

The complete representation of a given word is the concatenation of its word embedding and its character LSTM representation. While past work has also used sparse indicator features (Finkel et al., 2008) or part-of-speech tags predicted by an external system (Cross and Huang, 2016b) for additional word-level information, we find these to be unnecessary in the presence of a robust character-level representation.

2.3 Span Representation

To build up to spans, we first run a bidirectional LSTM over the sequence of word representations for an input sentence to obtain context-sensitive forward and backward representations $\mathbf{f}_i$ and $\mathbf{b}_i$ for each fencepost i. We then follow past work in dependency parsing (Wang and Chang, 2016) and constituency parsing (Cross and Huang, 2016b; Stern et al., 2017) in representing the span (i, j) by the concatenation of the corresponding forward and backward span differences:

$$\mathbf{r}_{ij} = [\mathbf{f}_j - \mathbf{f}_i, \mathbf{b}_i - \mathbf{b}_j].$$

See Figure 1 for an illustration.

2.4 Label Scoring

Finally, we implement the label scoring function by feeding the span representation through a one-layer feedforward network whose output dimensionality equals the number of possible labels. The score of a specific label ℓ is the corresponding component of the output vector:

$$s(i, j, \ell) = [\mathbf{W}_2\, g(\mathbf{W}_1 \mathbf{r}_{ij} + \mathbf{z}_1) + \mathbf{z}_2]_\ell\,,$$

where g is an elementwise ReLU nonlinearity.

2.5 Inference

Even though our model operates on n-ary trees, we can still employ a CKY-style algorithm for efficient globally optimal inference by introducing an auxiliary empty label $\varnothing$ with $s(i, j, \varnothing) = 0$ for all (i, j) to handle spans that are not constituents. Under this scheme, every binarization of a tree with empty labels at intermediate dummy nodes will have the same score, so an arbitrary binarization can be selected at training time with no effect on learning. We contrast this with the chart parser of Stern et al. (2017), which assigns different scores to different binarizations of the same underlying tree and in theory may exhibit varying performance depending on the method chosen for conversion.

With this change in place, let $s_{\text{best}}(i, j)$ denote the score of the best subtree spanning (i, j). For spans of length one, we need only consider the choice of label:

$$s_{\text{best}}(i, i+1) = \max_\ell s(i, i+1, \ell).$$

For general spans (i, j), we have the following recursion:

$$s_{\text{best}}(i, j) = \max_\ell s(i, j, \ell) + \max_k \left[s_{\text{best}}(i, k) + s_{\text{best}}(k, j) \right].$$

That is, we can independently select the best label for the current span and the best split point, where the score of a split is the sum of the best scores for the corresponding subtrees.

To parse the full sentence, we compute $s_{\text{best}}(0, n)$ using a bottom-up chart decoder, then traverse backpointers to recover the tree achieving that score. Nodes assigned the empty label are omitted during the reconstruction process to obtain the full n-ary tree. The overall complexity of this approach is $\mathcal{O}(n^3 + Ln^2)$, where n is the number of words and L is the total number of labels. We note that because our system does not use a grammar, there is no constant for the number of grammar rules multiplying the $\mathcal{O}(n^3)$ term as in traditional CKY parsing. In practice, the $\mathcal{O}(n^2)$ evaluations of the span scoring function corresponding to the $\mathcal{O}(Ln^2)$ term dominate runtime.

2.6 Training

As is common for structured prediction problems (Taskar et al., 2005), we use margin-based training to learn a model that satisfies the constraints

$$s(T^*) \geq s(T) + \Delta(T, T^*)$$

for each training example, where T^* denotes the gold output, T ranges over all valid trees, and Δ is the Hamming loss on labeled spans. Our training objective is the hinge loss:

$$\max\left(0,\, \max_T \left[s(T) + \Delta(T, T^*) \right] - s(T^*)\right).$$

This is equal to 0 when all constraints are satisfied, or the magnitude of the largest margin violation otherwise.

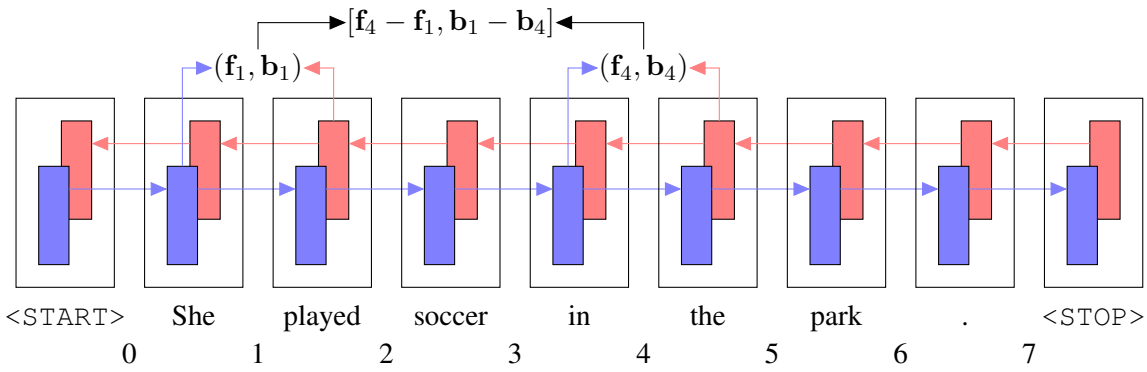


Figure 1: Span representations are computed by running a bidirectional LSTM over the input sentence and taking differences of the output vectors at the two endpoints. Here we illustrate the process for the span $(1, 4)$ corresponding to "played soccer in" in the example sentence.

Since Δ decomposes over spans, the inner loss-augmented decode $\max_T [s(T) + \Delta(T, T^*)]$ can be performed efficiently using a slight modification of the dynamic program used for inference. In particular, we replace $s(i, j, \ell)$ with $s(i, j, \ell) + 1[\ell \neq \ell^*_{ij}]$, where ℓ^*_{ij} is the label of span (i, j) in the gold tree T^*.

2.7 Results

We use the Penn Treebank (Marcus et al., 1993) for our experiments with the standard splits of sections 2-21 for training, section 22 for development, and section 23 for testing. Details about our model hyperparameters and training prodecure can be found in Appendix A.

Across 10 trials, our model achieves an average development F1 score of 92.22 on section 22 of the Penn Treebank. We use this as our primary point of comparison in all subsequent analysis. The model with the best score on the development set achieves a test F1 score of 92.08 on section 23 of the Penn Treebank, exceeding the performance of other recent state-of-the-art discriminative models which do not use external data or ensembling.[1]

3 Output Correlations

Output correlations are information about compatibility between outputs in a structured prediction model. Since outputs are all a function of the input, output correlations are not necessary for prediction when a model has access to the entire input. In practice, however, many models throughout NLP have found them useful (Collins, 1997; Lafferty et al., 2001; Koo and Collins, 2010), and Liang et al. (2008) provides theoretical results suggesting they may be useful for learning efficiently. In constituency parsing, there are two primary forms of output correlation typically captured by models. The first is correlations between label decisions, which often are captured by either production scores or the history in an incremental tree-creation procedure. The second, more subtle correlation comes from the enforcement of tree constraints, since the inclusion of one bracket can affect whether or not another bracket can be present. We explore these two classes of output correlations in Sections 3.1 and 3.2 below.

3.1 Parent Classification

The base parser introduced in Section 2 scores labeled brackets independently then uses a dynamic program to select a set of brackets that forms the highest-scoring tree. This independent labeling is an interesting departure from classical parsing work where correlations between predicted labels played a central role. It is natural to wonder why modeling label correlations isn't as important as it once was. Is there something about the neural representation that allows us to function without it? One possible explanation is that the neural machinery, in particular the LSTM, is handling much of the reconciliation between labels that was previously handled by an inference procedure. In other words, instead of using local information to suggest several brackets and letting the grammar handle interactions between them, the LSTM may be making decisions about brackets already in its latent state, allowing it to use the result of these decisions to inform other bracketings.

One way to explore this hypothesis would be

[1] Code for our parser is available at https://github.com/dgaddy/parser-analysis.

to evaluate whether the parser's learned representations could be used to predict parent labels of nodes in the tree. If the label of a node's parent can be predicted with high accuracy from the representation of its span, then little of the information about parent-child relations provided explicitly by a grammar has been lost. For this experiment, we freeze the input and LSTM parameters of our base model and train a new label scoring network to predict the label of a span's parent rather than the label of the span itself. We only predict parent labels for spans that have a bracket in the gold tree, so that all but the top level spans will have non-empty labels. The new network is trained with a margin loss.

After training on the standard training sections of the treebank, the network was able to correctly predict 92.3% of parent labels on the development set. This is fairly accurate, which supports the hypothesis that the representation knows a substantial amount about surrounding context in the output tree. For comparison, given only a span's label, the best you can do for predicting the parent is 43.3% with the majority class conditioned on the current label.

3.2 Independent Span Decisions

Like other recent parsers that do not capture correlations between output labels (Cross and Huang, 2016b; Stern et al., 2017), our base parser still does have some output correlations captured by the enforcement of tree constraints. In this section, we set out to determine the importance of these output correlations by making a version of the parser where they are removed. Although parsers are typically designed to form trees, the bracketing F1 measure used to evaluate parsers is still defined on non-tree outputs. To remove all output correlations from our parser, we can simply remove the tree constraint and independently make decisions about whether to include a bracketed span. The architecture is identical to the one described in Section 2, producing a vector of label scores for each span. We choose the label with the maximum score as the label for a span. As before, we fix the score of the empty label at zero, so if all other label scores are negative, the span will be left out of the set of predicted brackets. We train with independent margin losses for each span.

Ignoring tree well-formedness, the development F1 score of this independent span selection parser is 92.20, effectively matching the performance of the tree-constrained parser. In addition, we find that 94.5% of predicted bracketings for development set examples form valid trees, even though we did not explicitly encourage this. This high performance shows that our parser can function well even without modeling any output correlations.

4 Lexical Representation

In this section, we investigate several common choices for lexical representations of words and their role in neural parsing.

4.1 Alternate Word Representations

We compare the performance of our base model, which uses word embeddings and a character LSTM, with otherwise identical parsers that use other combinations of lexical representations. The results of these experiments are summarized in Table 1. First, we remove the character-level representations from our model, leaving only the word embeddings. We find that development performance drops from 92.22 F1 to 91.44 F1, showing that word embeddings alone do not capture sufficient information for state-of-the-art performance. Then, we replace the character-level representations with embeddings of part-of-speech tags predicted by the Stanford tagger (Toutanova et al., 2003). This model achieves a comparable development F1 score of 92.09, but unlike our base model relies on outputs from an external system. Next, we train a model which includes all three lexical representations: word embeddings, character LSTM representations, and part-of-speech tag embeddings. We find that development performance is nearly identical to the base model at 92.24 F1, suggesting that character representations and predicted part-of-speech tags provide much of the same information. Finally, we remove word embeddings and rely completely on character-level embeddings. After retuning the character LSTM size, we find that a slightly larger character LSTM can make up for the loss in word-level embeddings, giving a development F1 of 92.24.

4.2 Predicting Word Features

Past work in constituency parsing has demonstrated that indicator features on word shapes, suffixes, and similar attributes provide useful infor-

Word and Character LSTM	92.22
Word Only	91.44
Word and Tag	92.09
Word, Tag, and Character LSTM	92.24
Character Only	92.24

Table 1: Development F1 scores on section 22 of the Penn Treebank for different lexical representations.

mation beyond the identity of a word itself, especially for rare and unknown tokens (Finkel et al., 2008; Hall et al., 2014). We hypothesize that the character-level LSTM in our model learns similar information without the need for manual supervision. To test this, we take the word representations induced by the character LSTM in our parser as fixed word encodings, and train a small feed-forward network to predict binary word features defined in the Berkeley Parser (Petrov and Klein, 2007). We randomly split the vocabulary of the Penn Treebank into two subsets, using 80% of the word types for training and 20% for testing.

We find that the character LSTM representations allow for previously handcrafted indicator features to be predicted with accuracies of 99.7% or higher in all cases. The fact that this simple classifier performs so well indicates that the information contained in these features is readily available from our model's character-level encodings. A detailed breakdown of accuracy by feature can be found in Appendix B.

5 Context in the Sentence LSTM

In this section, we analyze where the information in the sentence-level LSTM hidden vectors comes from. Since the LSTM representations we use to make parsing decisions come from the fenceposts on each side of a span, we would like to understand whether they only capture information from the immediate vicinity of the fenceposts or if they also contain more distant information. Although an LSTM is theoretically capable of incorporating an arbitrarily large amount of context, it is unclear how much context it actually captures and whether this context is important for parsing accuracy.

5.1 Derivative Analysis

First, we would like to know if the LSTM features capture distant information. For this experiment, we use derivatives as a measure of sensitivity to changes in an input. If the derivative of a value with respect to a particular input is high, then that input has a large impact on the final value. For a particular component of an LSTM output vector, we compute its gradient with respect to each LSTM input vector, calculate the ℓ_2-norms of the gradients, and bucket the results according to distance from the output position. This process is repeated for every output position of each sentence in the development set, and the results are averaged within each bucket. Due to the scale of the required computation, we only use a subset of the output vector components to compute the average, sampling one at random per output vector.

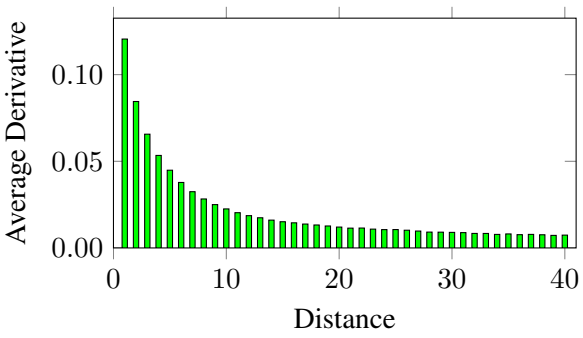


Figure 2: Average derivative of the LSTM output with respect to its input as a function of distance. The output is most sensitive to the closest words, but the tail of the distribution is fairly heavy, indicating that far-away words also have substantial impact.

Figure 2 illustrates how the average gradient norm is affected by the distance between the LSTM input and output. As would be expected, the closest input vectors have the largest effect on the hidden state. However, the tail of values is fairly heavy, with substantial gradient norms even for inputs 40 words away. This shows that far-away inputs do have an effect on the LSTM representation.

5.2 Truncation Analysis

Next, we investigate whether information in the LSTM representation about far-away inputs is actually important for parsing performance. To do so, we remove distant context information from our span encoding, representing spans by features obtained from LSTMs that are run on fixed-sized windows of size k around each fencepost. Figure 3 illustrates this truncated representation. Since the truncated representation also removes information about the size and position of the span in addition to the context words, we learn a position-dependent cell state initialization for each of the

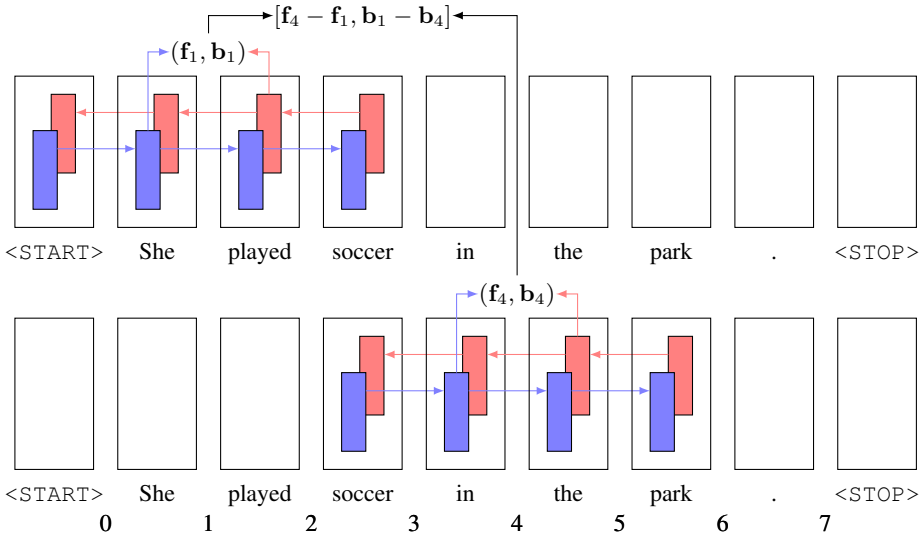


Figure 3: An example of creating a truncated span representation for the span "played soccer in" with context size $k = 2$. This representation is used to investigate the importance of information far away from the fenceposts of a span.

two LSTM directions to give a more fair comparison to the full LSTM. The use of a fixed-sized context window is reminiscent of prior work by Hall et al. (2014) and Durrett and Klein (2015), but here we use an LSTM instead of sparse features. We train parsers with different values of k and observe how their performance varies. All other architecture details and hyperparameters are the same as for the original model.

The blue points in Figure 4 show how the context size k affects parser performance for $k \in \{2, 3, 5, 10, 20, 30\}$. As with the derivative analysis, although most of the weight is carried by the nearby inputs, a nontrivial fraction of performance is due to context more than 10 words away.

5.3 Word Order

Now that we have established that long-distance information is important for parsing performance, we would like to know whether the order of the far-away words is important. Is the LSTM capturing far-away structure, or is the information more like a bag-of-words representation summarizing the words that appear?

To test the importance of order, we train a parser where information about the order of far-away words is destroyed. As illustrated in Figure 5, we run a separate LSTM over the entire sentence for each fencepost, shuffling the input depending on the particular fencepost being represented. We randomly shuffle words outside a context window

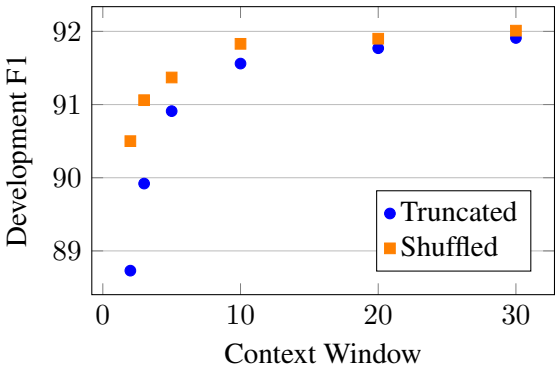


Figure 4: Development F1 as the amount of context given to the sentence-level LSTM varies. The blue points represent parser performance when the LSTM is truncated to a window around the fenceposts, showing that far-away context is important. The orange points represent performance when the full context is available but words outside a window around the fenceposts are shuffled, showing that the order of far-away context is also important.

of size k around the fencepost of interest, keeping words on the left and the right separate so that directional information is preserved but exact positions are lost.

The orange points in Figure 4 show the performance of this experiment with different context sizes k. We observe that including shuffled distant words is substantially better than truncating them completely. On the other hand, shuffling does cause performance to degrade relative to the base parser even when the unshuffled win-

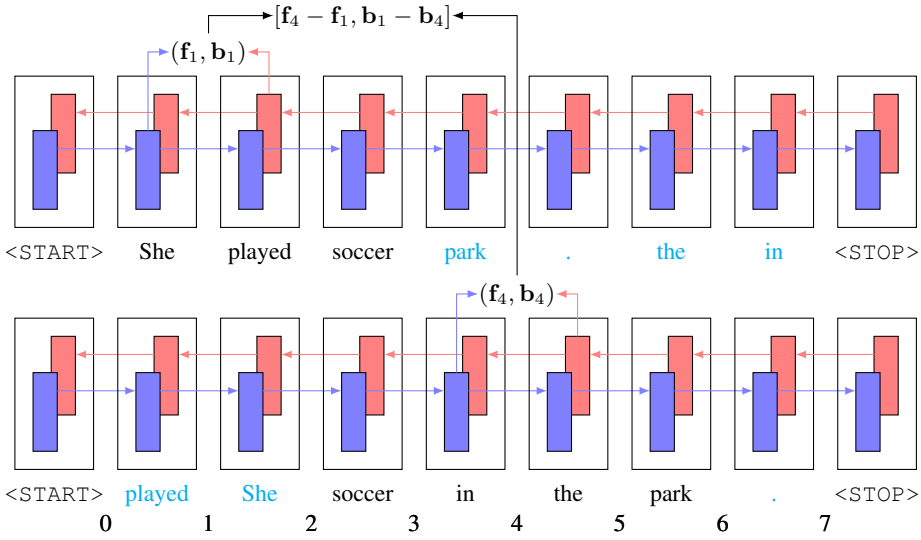


Figure 5: An example of creating a shuffled span representation for the span "played soccer in" with context size $k = 2$. The light blue words are outside the context window and are shuffled randomly. Shuffled representations are used to explore whether the order of far-away words is important.

dow is moderately large, indicating that the LSTM is propagating information that depends on the order of words in far-away positions.

5.4 LSTMs vs. Feedforward

Finally, we investigate whether the LSTM architecture itself is important for reasons other than just the amount of context it can capture. Like any architecture, the LSTM introduces particular inductive biases that affect what gets learned, and these could be important for parser performance. We run a version of the truncation experiment from Section 5.2 where we use a feedforward network in place of a sentence-level LSTM to process the surrounding context of each fencepost. The input to the network is the concatenation of the word representations that would be used as inputs for the truncated LSTM, and the output is a vector of the same size as the LSTM-based representation. As in Section 5.2, we wish to give our representation information about span size and position, so we also include a learned fencepost position embedding in the concatenated inputs to the network. We focus on context window size $k = 3$ for this experiment. We search among networks with one, two, or three hidden layers that are one, two, or four times the size of the LSTM hidden state.

Of all the feedforward networks tried, the maximum development performance was 83.39 F1, compared to 89.92 F1 for the LSTM-based truncation. This suggests that some property of the LSTM makes it better suited for the task of summarizing context than a flat feedforward network.

6 Related Analysis Work

Here we review other works that have performed similar analyses to ours in parsing and other areas of NLP. See Section 2 for a description of how our parser is related to other parsers.

Similar to our independent span prediction in Section 3.2, several works have found that their models still produce valid outputs for the majority of inputs even after relaxing well-formedness constraints. In dependency parsing, Zhang et al. (2017) and Chorowski et al. (2016) found that selecting dependency heads independently often resulted in valid trees for their parsers (95% and 99.5% of outputs form trees, respectively). In constituency parsing, the parser of Vinyals et al. (2015), which produced linearized parses token by token, was able to output valid constituency trees for the majority of sentences (98.5%) even though it was not constrained to do so.

Several other works have investigated what information is being captured within LSTM representations. Chawla et al. (2017) performed analysis of bidirectional LSTM representations in the context of named entity recognition. Although they were primarily interested in finding specific word types that were important for making decisions, they also analyzed how distance affected a word's impact. Shi et al. (2016) and Linzen et al.

(2016) perform analysis of LSTM representations in machine translation and language modeling respectively to determine whether syntactic information is present. Some of their techniques involve classification of features from LSTM hidden states, similar to our analysis in Sections 3.1 and 4.2.

In Section 5.4, we found that replacing an LSTM with a feedforward network hurt performance. Previously, Chelba et al. (2017) had similar findings in language modeling, where using LSTMs truncated to a particular distance improved performance over feedforward networks that were given the same context.

7 Conclusion

In this paper, we investigated the extent to which information provided directly by model structure in classical constituency parsers is still being captured by neural methods. Because neural models function in a substantially different way than classical systems, it could be that they rely on different information when making their decisions. Our findings suggest that, to the contrary, the neural systems are learning to capture many of the same knowledge sources that were previously provided, including the parent-child relations encoded in grammars and the word features induced by lexicons.

Acknowledgments

This work is supported by the DARPA Explainable Artificial Intelligence (XAI) program and the UC Berkeley Savio computational cluster. The second author is supported by an NSF Graduate Research Fellowship.

References

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 349–359. https://doi.org/10.18653/v1/D15-1041.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*. Association for Computational Linguistics, pages 173–180. http://www.aclweb.org/anthology/P05-1022.

Kushal Chawla, Sunil Kumar Sahu, and Ashish Anand. 2017. Investigating how well contextual features are captured by bi-directional recurrent neural network models. *arXiv preprint arXiv:1709.00659* .

Ciprian Chelba, Mohammad Norouzi, and Samy Bengio. 2017. N-gram language modeling using recurrent neural network estimation. *arXiv preprint arXiv:1703.10724* .

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 2331–2336. https://doi.org/10.18653/v1/D16-1257.

Jan Chorowski, Michał Zapotoczny, and Paweł Rychlikowski. 2016. Read, tag, and parse all at once, or fully-neural dependency parsing. *arXiv preprint arXiv:1609.03441* .

Maximin Coavoux and Benoit Crabbé. 2017. Multilingual lexicalized constituency parsing with word-level auxiliary tasks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, pages 331–336. http://aclweb.org/anthology/E17-2053.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *35th Annual Meeting of the Association for Computational Linguistics*. http://www.aclweb.org/anthology/P97-1003.

James Cross and Liang Huang. 2016a. Incremental parsing with minimal features using bi-directional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 32–37. https://doi.org/10.18653/v1/P16-2006.

James Cross and Liang Huang. 2016b. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1–11. https://doi.org/10.18653/v1/D16-1001.

Greg Durrett and Dan Klein. 2015. Neural crf parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 302–312. https://doi.org/10.3115/v1/P15-1030.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 199–209. https://doi.org/10.18653/v1/N16-1024.

Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning. 2008. Efficient, feature-based, conditional random field parsing. In *Proceedings of ACL-08: HLT*. Association for Computational Linguistics, Columbus, Ohio, pages 959–967. http://www.aclweb.org/anthology/P08-1109.

David Hall, Greg Durrett, and Dan Klein. 2014. Less grammar, more features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 228–237. https://doi.org/10.3115/v1/P14-1022.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press, pages 2741–2749.

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

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*. http://www.aclweb.org/anthology/P03-1054.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 1–11. http://www.aclweb.org/anthology/P10-1001.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data .

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 260–270. https://doi.org/10.18653/v1/N16-1030.

Percy Liang, Hal Daumé III, and Dan Klein. 2008. Structure compilation: trading structure for features. In *Proceedings of the 25th international conference on Machine learning*. ACM, pages 592–599.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015a. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1520–1530. https://doi.org/10.18653/v1/D15-1176.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015b. Character-based neural machine translation. *arXiv preprint arXiv:1511.04586* .

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association of Computational Linguistics* 4:521–535. http://www.aclweb.org/anthology/Q16-1037.

Jiangming Liu and Yue Zhang. 2017. Shift-reduce constituent parsing with neural lookahead features. *Transactions of the Association of Computational Linguistics* 5:45–58. http://www.aclweb.org/anthology/Q17-1004.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.* 19(2):313–330. http://dl.acm.org/citation.cfm?id=972470.972475.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980* .

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*. Association for Computational Linguistics, pages 404–411. http://www.aclweb.org/anthology/N07-1051.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural mt learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1526–1534. https://doi.org/10.18653/v1/D16-1159.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 818–827. https://doi.org/10.18653/v1/P17-1076.

Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. 2005. Learning structured prediction models: A large margin approach. In *Proceedings of the 22nd international conference on Machine learning*. ACM, pages 896–903.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, NAACL '03, pages 173–180. https://doi.org/10.3115/1073445.1073478.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*. pages 2773–2781.

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 2306–2315. https://doi.org/10.18653/v1/P16-1218.

Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. 2017. Dependency parsing as head selection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, pages 665–676. http://www.aclweb.org/anthology/E17-1063.

A Model Hyperparameters and Training Details

Component	Dimensions	Layers
Word Embeddings	100	
Character Embeddings	50	
Character LSTM	100	1
Sentence LSTM	250	2
Label Feedforward Network	250	1

Table 2: The sizes of the components used in our model.

Our model hyperparameters are summarized in Table 2. We train using the Adam optimizer (Kingma and Ba, 2014) with its default hyperparameters for 40 epochs. We evaluate on the development set 4 times per epoch, selecting the model with the highest overall development performance as our final model. When performing a word embedding lookup during training, we randomly replace words by the `<UNK>` token with probability $1/(1 + \text{freq}(w))$, where $\text{freq}(w)$ is the frequency of a word w in the training set. We apply dropout with probability 0.4 before and inside each layer of each LSTM. Our system is implemented in Python using DyNet (Neubig et al., 2017).

B Character LSTM Word Feature Classification

Binary Feature	Majority Class	Char-LSTM Classifier	Binary Feature	Majority Class	Char-LSTM Classifier
all-letters	77.22%	99.77%	suffix = “s”	82.65%	99.99%
has-letter	89.18%	99.97%	suffix = “ed”	92.52%	99.98%
all-lowercase	56.95%	99.95%	suffix = “ing”	93.26%	99.95%
has-lowercase	85.85%	99.90%	suffix = “ion”	97.75%	99.93%
all-uppercase	96.68%	99.90%	suffix = “er”	96.42%	99.97%
has-uppercase	67.77%	99.97%	suffix = “est”	99.63%	99.98%
all-digits	98.38%	99.99%	suffix = “ly”	97.56%	99.99%
has-digit	87.90%	99.91%	suffix = “ity”	99.30%	99.94%
all-punctuation	99.93%	99.98%	suffix = “y”	92.97%	99.93%
has-punctuation	79.04%	99.75%	suffix = “al”	98.48%	99.92%
has-dash	88.89%	99.95%	suffix = “ble”	99.30%	99.90%
has-period	92.55%	99.95%	suffix = “e”	89.57%	99.99%
has-comma	98.02%	99.97%			

Table 3: Classification accuracy for various binary word features using the character LSTM representations for words induced by a pre-trained parser. Performance substantially exceeds that of a majority class classifier in all cases, reaching 99.7% or higher for all features. The majority class is `True` for the first four features in the left column and `False` for the rest.

Deep Generative Model for Joint Alignment and Word Representation

Miguel Rios **Wilker Aziz** **Khalil Sima'an**
Institute for Logic, Language, and Computation
University of Amsterdam
{m.riosgaona, w.aziz, k.simaan}@uva.nl

Abstract

This work exploits translation data as a source of semantically relevant learning signal for models of word representation. In particular, we exploit equivalence through translation as a form of distributional context and jointly learn how to embed and align with a deep generative model. Our EMBEDALIGN model embeds words in their complete observed context and learns by marginalisation of latent lexical alignments. Besides, it embeds words as posterior probability densities, rather than point estimates, which allows us to compare words in context using a measure of overlap between distributions (e.g. KL divergence). We investigate our model's performance on a range of lexical semantics tasks achieving competitive results on several standard benchmarks including natural language inference, paraphrasing, and text similarity.

1 Introduction

Natural language processing applications often count on the availability of word representations trained on large textual data as a means to alleviate problems such as data sparsity and lack of linguistic resources (Collobert et al., 2011; Socher et al., 2011; Tu et al., 2017; Bowman et al., 2015).

Traditional approaches to inducing word representations circumvent the need for explicit semantic annotation by capitalising on some form of indirect semantic supervision. A typical example is to fit a binary classifier to detect whether or not a target word is likely to co-occur with neighbouring words (Mikolov et al., 2013). If the binary classifier represents a word as a continuous vector, that vector will be trained to be discriminative of the contexts it co-occurs with, and thus words in similar contexts will have similar representations.

The underlying assumption is that context (e.g. neighbouring words) stands for the meaning of the target word (Harris, 1954; Firth, 1957). The success of this *distributional hypothesis* hinges on the definition of context and different models are based on different definitions. Importantly, the nature of the context determines the range of linguistic properties the representations may capture (Levy and Goldberg, 2014b). For example, Levy and Goldberg (2014a) propose to use syntactic context derived from dependency parses. They show that their representations are much more discriminative of syntactic function than models based on immediate neighbourhood (Mikolov et al., 2013).

In this work, we take lexical translation as indirect semantic supervision (Diab and Resnik, 2002). Effectively we make two assumptions. First, that every word has a foreign equivalent that stands for its meaning. Second, that we can find this equivalent in translation data through lexical alignments.[1] For that we induce both a latent mapping between words in a bilingual sentence pair and distributions over latent word representations.

To summarise our contributions:

- we model a joint distribution over sentence pairs that generates data from latent word representations and latent lexical alignments;
- we embed words in context mining positive correlations from translation data;
- we find that foreign observations are necessary for generative training, but test time predictions can be made monolingually;
- we apply our model to a range of semantic natural language processing tasks showing its usefulness.

Code available from https://github.com/uva-slpl/embedalign

MR and WA contributed equally.

[1] These assumptions are not new to the community, but in this work they lead to a novel model which reaches more applications. §4 expands on the relation to other uses of bilingual data for word representation.

Proceedings of NAACL-HLT 2018, pages 1011–1023
New Orleans, Louisiana, June 1 - 6, 2018.

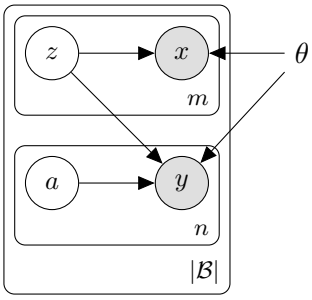


Figure 1: A sequence x_1^m is generated conditioned on a sequence of random embeddings z_1^m; generating the foreign sequence y_1^n further requires latent lexical alignments a_1^n.

2 EMBEDALIGN

In a nutshell, we model a distribution over pairs of sentences expressed in two languages, namely, a language of interest L1, and an auxiliary language L2 which our model uses to mine some learning signal. Our model, EMBEDALIGN, is governed by a simple generative story:

1. sample a length m for a sentence in L1 and a length n for a sentence in L2;
2. generate a sequence $z_1, \ldots, z_m$ of d-dimensional random embeddings by sampling independently from a standard Gaussian prior;
3. generate a word observation x_i in the vocabulary of L1 conditioned on the random embedding z_i;
4. generate a sequence $a_i, \ldots, a_n$ of n random alignments—each maps from a position a_j in x_1^m to a position j in the L2 sentence;
5. finally, generate an observation y_j in the vocabulary of L2 conditioned on the random embedding z_{a_j} that stands for x_{a_j}.

The model is parameterised by neural networks and parameters are estimated to maximise a lower-bound on log-likelihood of joint observations. In the following, we present the model formally (§2.1), discuss efficient training (§2.2), and concrete architectures (§2.3).

2.1 Probabilistic model

Notation We use block capitals (e.g. X) for random variables, lowercase letters (e.g. x) for assignments, and the shorthand X_1^m for a sequence $X_1, \ldots, X_m$. Boldface letters are reserved for deterministic vectors (e.g. $\mathbf{v}$) and matrices (e.g. $\mathbf{W}$). Finally, $\mathbb{E}[f(Z); \alpha]$ denotes the expected value of $f(z)$ under a density $q(z|\alpha)$.

We model a joint distribution over bilingual parallel data, i.e., L1–L2 sentence pairs. An observation is a pair of random sequences $\langle X_1^m, Y_1^n \rangle$, where a random variable X (Y) takes on values in the vocabulary of L1 (L2). For ease of exposition, the length m (n) of each sequence is assumed observed throughout. The L1 sentence is generated one word at a time from a random sequence of latent embeddings Z_1^m, each Z taking on values in $\mathbb{R}^d$. The L2 sentence is generated one word at a time given a random sequence of latent alignments A_1^n, where $A_j \in \{1, \ldots, m\}$ is the position in the L1 sentence to which y_j aligns.[2]

For $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, n\}$ the generative story is

$$Z_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{1a}$$
$$X_i | z_i \sim \text{Cat}(f(z_i; \theta)) \tag{1b}$$
$$A_j | m \sim \mathcal{U}(1/m) \tag{1c}$$
$$Y_j | z_1^m, a_j \sim \text{Cat}(g(z_{a_j}; \theta)) \tag{1d}$$

and Figure 1 is a graphical depiction of our model. We map from latent embeddings to categorical distributions over either vocabulary using a neural network whose parameters are deterministic and collectively denote by θ (the *generative parameters*). The marginal likelihood of a sentence pair is shown in Equation (2).

$$P_\theta(x_1^m, y_1^n | m, n) = \int p(z_1^m) \prod_{i=1}^{m} P_\theta(x_i | z_i) \times \prod_{j=1}^{n} \sum_{a_j=1}^{m} P(a_j | m) P_\theta(y_j | z_{a_j}) \mathrm{d}z_1^m \tag{2}$$

Due to the conditional independences of our model, it is trivial to marginalise lexical alignments for any given latent embeddings z_1^m, but marginalising the embeddings themselves is intractable. Thus, we employ amortised mean field variational inference using the inference model

$$q_\phi(z_1^m | x_1^m) \triangleq \prod_{i=1}^{m} \mathcal{N}(z_i | \mathbf{u}_i, \text{diag}(\mathbf{s}_i \odot \mathbf{s}_i)) \tag{3}$$

where each factor is a diagonal Gaussian. We map from x_1^m to a sequence $\mathbf{u}_1^m$ of independent posterior

[2] We pad L1 sentences with NULL to account for untranslatable L2 words (Brown et al., 1993). Instead, Schulz et al. (2016) generate untranslatable words from L2 context—an alternative we leave for future work.

mean (or location) vectors, where $\mathbf{u}_i \triangleq \mu(\mathbf{h}_i; \phi)$, as well as a sequence $\mathbf{s}_1^m$ of independent standard deviation (or scale) vectors, where $\mathbf{s}_i \triangleq \sigma(\mathbf{h}_i; \phi)$, and $\mathbf{h}_1^m = \text{enc}(x_1^m; \phi)$ is a deterministic encoding of the L1 sequence (we discuss concrete architectures in §2.3). All mappings are realised by neural networks whose parameters are collectively denoted by ϕ (the *variational parameters*). Note that we choose to approximate the posterior without conditioning on y_1^n. This allows us to use the inference model for *monolingual* prediction in absence of L2 data.

Variational ϕ and generative θ parameters are jointly point-estimated to attain a local optimum of the evidence lowerbound (Jordan et al., 1999):

$$\log P_\theta(x_1^m, y_1^n | m, n) \geq \sum_{i=1}^{m} \mathbb{E}\left[\log P_\theta(x_i|Z_i); \mathbf{u}_i, \mathbf{s}_i\right] + \sum_{j=1}^{n} \mathbb{E}\left[\log \sum_{a_j=1}^{m} P(a_j|m) P_\theta(y_j|Z_{a_j}); \mathbf{u}_1^m, \mathbf{s}_1^m\right] - \sum_{i=1}^{m} \text{KL}\left[\mathcal{N}(\mathbf{u}_i, \text{diag}(\mathbf{s}_i \odot \mathbf{s}_i)) || \mathcal{N}(\mathbf{0}, \mathbf{I})\right] \quad . \tag{4}$$

The variational family is location-scale, thus we can rely on stochastic optimisation (Robbins and Monro, 1951) and automatic differentiation (Baydin et al., 2015) with reparameterised gradient estimates (Kingma and Welling, 2014; Rezende et al., 2014; Titsias and Lázaro-Gredilla, 2014). Moreover, because the Gaussian density is an exponential family, the KL terms in (4) are available in closed-form (Kingma and Welling, 2014, Appendix B).

2.2 Efficient training

The likelihood terms in the ELBO (4) require evaluating two softmax layers over rather large vocabularies. This makes training prohibitively slow and calls for efficient approximation. We employ an approximation proposed by Botev et al. (2017) termed *complementary sum sampling* (CSS), which we review in this section.

Consider the likelihood term $\log P(X = \mathsf{x}|z)$ that scores an observation x given a sampled embedding z—we use serif font x to distinguish a particular observation from an arbitrary event $x \in \mathcal{X}$ in the support. The exact class probability

$$P(X = \mathsf{x}|z) = \frac{\exp(u(z, \mathsf{x}))}{\sum_{x \in \mathcal{X}} \exp(u(z, x))} \tag{5}$$

requires a normalisation over the complete support. CSS works by splitting the support into two sets, a set $\mathcal{C}$ that is explicitly summed over and must include the *positive* class x, and another set $\mathcal{N}$ that is a subset of the complement set $\mathcal{X} \setminus C$. We obtain an estimate for the normaliser

$$\sum_{x \in \mathcal{C}} \exp(u(z, x)) + \sum_{x \in \mathcal{N}} \kappa(x) \exp(u(z, x)) \tag{6}$$

by importance- or Bernoulli-sampling from the support using a proposal distribution $Q(X)$, where $\kappa(x)$ corrects for bias as $\mathcal{N}$ tends to the entire complement set. In this paper, we design $\mathcal{C}$ and $\mathcal{N}$ per training mini-batch: we take $\mathcal{C}$ to consist of all unique words in a mini-batch of training samples and $\mathcal{N}$ to consist of 10^3 negative classes uniformly sampled from the complement set $\mathcal{X} \setminus \mathcal{C}$, in which case $\kappa(x) = 10^{-3}|\mathcal{X} \setminus \mathcal{C}|$.[3]

CSS makes it particularly easy to approximate likelihood terms such as those with respect to L2 in Equation (4). Because those terms depend on a marginalisation over alignments, an approximation must give support to all words in the sequence y_1^n. With CSS this is extremely simple, we just need to make sure all unique words in y_1^n are in the set $\mathcal{C}$—which our mini-batch procedure does guarantee. Botev et al. (2017) show that CSS is rather stable and superior to the most popular softmax approximations. Besides being simple to implement, CSS also addresses a few problems with other approximations. To name a few: unlike importance sampling approximations, CSS converges to the exact softmax with bounded computation (it takes as many samples as there are classes). Unlike hierarchical softmax, CSS only affects training, that is, at test time we simply use the entire support instead of the approximation.

Without a softmax approximation, inference for our model would take time proportional to $O(m \times v_{\mathsf{x}} + m \times v_{\mathsf{y}} + m \times n)$ where v_{x} (v_{y}) corresponds to the size of the vocabulary of L1 (L2). The first term ($m \times v_{\mathsf{x}}$) corresponds to projecting from m latent embeddings to m categorical distributions over the vocabulary of L1. The second term ($m \times v_{\mathsf{y}}$) corresponds to projecting the same m latent embeddings to m categorical distributions over the vocabulary of L2. Finally, the third term ($m \times n$) is due to marginalisation of alignments.

[3] We sample uniformly from the complement set until we have 10^3 unique classes. We realise this operation outside the computation graph providing $\mathcal{C}$ and $\mathcal{N}$ as inputs to each training iteration, but a GPU-based solution is also possible.

Note, however, that with the CSS approximation we drop the dependency on vocabulary sizes (as the combined sizes of $\mathcal{C}$ and $\mathcal{N}$ is an independent constant). Moreover, if inference is performed on GPU, the squared term ($m \times n \approx m^2$) is amortised due to parallelism. Thus, while training our model is somewhat slower than monolingual models of word representation, which typically run in $O(m)$, it is not at all impracticably slower.

2.3 Architectures

Here we present the neural network architectures that parameterise the different generative and variational components of §2.1. Refer to Appendix B for an illustration.

Generative model We have two generative components, namely, a categorical distribution over the vocabulary of L1 and another over the vocabulary of L2. We predict the parameter (event probabilities) of each distribution with an affine transformation of a latent embedding followed by the softmax nonlinearity to ensure normalisation:

$$f(z_i; \theta) = \text{softmax}\left(\mathbf{W}_1 z_i + \mathbf{b}_1\right) \tag{7a}$$

$$g(z_{a_j}; \theta) = \text{softmax}\left(\mathbf{W}_2 z_{a_j} + \mathbf{b}_2\right) \tag{7b}$$

where $\mathbf{W}_1 \in \mathbb{R}^{v_x \times d}$, $\mathbf{b}_1 \in \mathbb{R}^{v_x}$, $\mathbf{W}_2 \in \mathbb{R}^{v_y \times d}$, $\mathbf{b}_2 \in \mathbb{R}^{v_y}$, and v_x (v_y) is the size of the vocabulary of L1 (L2). With the approximation of §2.2, we replace the L1 softmax layer (7a) by $\exp\left(z_i^\top \mathbf{c}_x + b_x\right)$ normalised by the CSS estimate (6) at training, and similarly for the L2 softmax layer (7b). In that case, we have parameters for $\mathbf{c}_x, \mathbf{c}_y \in \mathbb{R}^d$—deterministic embeddings for x and y, respectively—as well as bias terms $b_x, b_y \in \mathbb{R}$.

Inference model We predict approximate posterior parameters using two independent transformations

$$\mathbf{u}_i = \mathbf{M}_1 \mathbf{h}_i + \mathbf{d}_1 \tag{8a}$$

$$\mathbf{s}_i = \text{softplus}(\mathbf{M}_2 \mathbf{h}_i + \mathbf{d}_2) \tag{8b}$$

of a shared representation $\mathbf{h}_i \in \mathbb{R}^{d_x}$ of the ith word in the L1 sequence x_1^m—where $\mathbf{M}_1, \mathbf{M}_2 \in \mathbb{R}^{d \times d_x}$ are projection matrices, $\mathbf{d}_1, \mathbf{d}_2 \in \mathbb{R}^d$ are bias vectors, and the softplus nonlinearity ensures that standard deviations are non-negative. To obtain the deterministic encoding $\mathbf{h}_1^m$, we employ two different architectures: (1) a bag-of-words (BOW) encoder, where $\mathbf{h}_i$ is a deterministic projection of x_i onto $\mathbb{R}^{d_x}$; and (2) a bidirectional (BIRNN) encoder, where $\mathbf{h}_i$ is the element-wise sum of two LSTM hidden states (ith step) that process the sequence in opposite directions. We use 128 units for deterministic embeddings, and 100 units for LSTMs (Hochreiter and Schmidhuber, 1997) and latent representations (i.e. $d = 100$).

3 Experiments

We start the section describing the data used to estimate our model's parameters as well as details about the optimiser. The remainder of the section presents results on various benchmarks.

Training data We train our model on bilingual parallel data. In particular, we use parliament proceedings (Europarl-v7) (Koehn, 2005) from two language pairs: English-French and English-German.[4] We employed very minimal preprocessing, namely, tokenisation and lowercasing using scripts from MOSES (Koehn et al., 2007), and have discarded sentences longer than 50 tokens. Table 1 lists more information about the training data, including the English-French Giga web corpus (Bojar et al., 2014) which we use in §3.4.[5]

Corpus	Sentence pairs	Tokens
Europarl EN-FR	1.7	42.5
Europarl EN-DE	1.7	43.5
Giga EN-FR	18.3	419.6

Table 1: Training data size (in millions).

Optimiser For all architectures, we use the Adam optimiser (Kingma and Ba, 2014) with a learning rate of 10^{-3}. Except where explicitly indicated, we

- train our models for 30 epochs using mini batches of 100 sentence pairs;
- use validation alignment error rate for model selection;
- train every model 10 times with random *Glorot* initialisation (Glorot and Bengio, 2010) and report mean and standard deviation;
- anneal the KL terms using the following schedule: we use a scalar α from 0 to 1 with additive steps of size 10^{-3} every 500 updates.

[4] The proposed model is not limited to these language pairs.

[5] As we investigate various configurations and train every model 10 times to inspect variance in results, we conduct most of the experiments on the more manageable Europarl.

This means that at the beginning of the training, we allow the model to overfit to the likelihood terms, but towards the end we are optimising the true ELBO (Bowman et al., 2016).

It is also important to highlight that we do not employ regularisation techniques (such as batch normalisation, dropout, or L_2 penalty) for they did not seem to yield consistent results.

3.1 Word alignment

Since our model leverages learning signal from parallel data by marginalising latent lexical alignments, we use alignment error rate to double check whether the model learns sensible word correspondences. Intrinsic assessment of word alignment quality requires manual annotation. For English-French, we use the NAACL English-French hand-aligned data (37 sentence pairs for validation and 447 for test) (Mihalcea and Pedersen, 2003). For English-German, we use the data by Padó and Lapata (2006) (98 sentence pairs for validation and 987 for test). Alignment quality is then measured in terms of alignment error rate (AER) (Och and Ney, 2000)—an F-measure over predicted alignment links. For prediction we condition on the posterior means $\mathbb{E}[Z_1^m]$ which is just the predicted variational means $\mathbf{u}_1^m$ and select the `L1` position for which $P(y_j, a_j|\mathbf{u}_1^m)$ is maximum (a form of approximate Viterbi alignment).

Model	`L1` accuracy	`L2` accuracy	↓AER
BoW	95.59 ± 2.22	5.69 ± 2.07	35.41 ± 1.16
BoW$_\alpha$	99.87 ± 0.22	6.16 ± 0.39	30.94 ± 2.49
BiRNN	95.72 ± 1.28	7.31 ± 0.64	34.32 ± 1.08
BiRNN$_\alpha$	99.97 ± 0.09	7.25 ± 0.62	29.18 ± 1.91

Table 2: English-French validation ↑accuracy and ↓AER results.

Model	`L1` accuracy	`L2` accuracy	↓AER
BoW	93.51 ± 0.56	10.09 ± 0.20	53.66 ± 0.36
BoW$_\alpha$	97.72 ± 2.28	9.71 ± 0.63	52.81 ± 1.47
BiRNN	99.78 ± 0.18	8.63 ± 0.35	55.55 ± 0.67
BiRNN$_\alpha$	99.96 ± 0.05	8.32 ± 0.29	52.32 ± 1.77

Table 3: English-German validation ↑accuracy and ↓AER results.

We start by analysing validation results and selecting amongst a few variants of EmbedAlign. We investigate the use of annealing and the use of a bidirectional encoder in the variational approximation. Table 2 (3) lists ↓AER for En-Fr (En-De) as well as accuracy of word prediction. It is clear that both annealing (systems decorated with subscript α) and bidirectional representations improve the results across the board. In the rest of the paper we still investigate whether or not recurrent encoders help, but we always report results based on annealing.

In order to establish baselines for our models we report IBM models 1 and 2 (Brown et al., 1993). In a nutshell, IBM models 1 and 2 both estimate the conditional $P(y_j|x_1^m) = \sum_{a_j=1}^m P(a_j|m)P(y_j|x_{a_j})$ by marginalisation of latent lexical alignments. The only difference between the two models is the prior over alignments, which is uniform for IBM1 and categorical for IBM2. An important difference between IBM models and EmbedAlign concerns the lexical distribution. IBM models are parameterised with independent categorical parameters, while our model instead is parameterised by a neural network. IBM models condition on a single categorical event x_{a_j}, namely, the word aligned to. Our model instead conditions on the latent embedding z_{a_j} that stands for the word aligned to.

In order to establish even stronger conditional alignment models, we embed the conditioning words and replace IBM1's independent parameters by a neural network (single hidden layer MLP). We call this model a *neural IBM1* (or NIBM for short). Note that in an IBM model, the sequence x_1^m is never modelled, therefore we can condition on it without restrictions. For that reason, we also experiment with a bidirectional LSTM encoder and condition lexical distributions on its hidden states.

Model	En-Fr	En-De
IBM1	32.45	46.71
IBM2	22.61	40.11
NIBM$_{\text{BoW}}$	27.35 ± 0.19	46.22 ± 0.07
NIBM$_{\text{BiRNN}}$	25.57 ± 0.40	43.37 ± 0.11
EmbedAlign$_{\text{BoW}}$	30.97 ± 2.53	49.46 ± 1.72
EmbedAlign$_{\text{BiRNN}}$	29.43 ± 1.84	48.09 ± 2.12

Table 4: Test ↓AER.

Table 4 shows AER for test predictions. First observe that neural models outperform classic IBM1 by far, some of them even approach IBM2's performance. Next, observe that bidirectional encodings make NIBM much stronger at inducing good word-

to-word correspondences. EMBEDALIGN cannot catch up with NIBM, but that is not necessarily surprising. Note that NIBM is a conditional model, thus it can use all of its capacity to better explain L2 data. EMBEDALIGN, on the other hand, has to find a compromise between generating both streams of the data. To make that point a bit more obvious, Table 5 (6) lists accuracy of word prediction for EN-FR (EN-DE). Note that, without sacrificing L2 accuracy, and sometimes even improving it, EMBEDALIGN achieves very high L1 accuracy. This still does not imply that induced representations have captured aspects of lexical semantics such as word senses. All this means is that we have induced features that are jointly good at reconstructing both streams of the data one word at time. Of course it is tempting to conclude that our models must be capturing some useful generalisations. For that, the next sections will investigate a range of semantic NLP tasks.

Model	L1 accuracy	L2 accuracy
NIBM_{BoW}	-	7.21 ± 0.16
$\text{NIBM}_{\text{BiRNN}}$	-	6.47 ± 0.45
$\text{EMBALIGN}_{\text{BoW}}$	98.90 ± 0.41	7.08 ± 0.34
$\text{EMBALIGN}_{\text{BiRNN}}$	99.21 ± 0.18	7.44 ± 0.61

Table 5: English-French ↑accuracy in test set

Model	L1 accuracy	L2 accuracy
NIBM_{BoW}	-	7.94 ± 0.03
$\text{NIBM}_{\text{BiRNN}}$	-	8.38 ± 0.10
$\text{EMBALIGN}_{\text{BoW}}$	96.86 ± 2.89	8.72 ± 0.39
$\text{EMBALIGN}_{\text{BiRNN}}$	99.32 ± 0.34	8.00 ± 0.12

Table 6: English-German ↑accuracy in test set

3.2 Lexical substitution task

The English lexical substitution task (LST) consists in selecting a substitute word for a target word *in context* (McCarthy and Navigli, 2009). In the most traditional variant of the task, systems are presented with a list of potential candidates and this list must be sorted by relatedness.

Dataset The LST dataset includes 201 target words present in 10 sentences/contexts each, along with a manually annotated list of potential replacements. The data are split in 300 instances for validation and 1,710 for test. Systems are evaluated by comparing the predicted ranking to the manual one in terms of generalised average precision (GAP) (Melamud et al., 2015).

Model	cos	KL
RANDOM	30.0	-
SKIPGRAM	44.9	-
BSG	-	46.1
EN_{BoW}	29.75 ± 0.55	27.93 ± 0.25
EN_{BiRNN}	21.31 ± 1.05	27.64 ± 0.40
$\text{EN-FR}_{\text{BoW}}$	42.72 ± 0.36	41.90 ± 0.35
$\text{EN-FR}_{\text{BiRNN}}$	42.19 ± 0.57	41.61 ± 0.55
$\text{EN-DE}_{\text{BoW}}$	41.90 ± 0.58	40.63 ± 0.55
$\text{EN-DE}_{\text{BiRNN}}$	42.07 ± 0.47	40.93 ± 0.59

Table 7: English ↑GAP on LST test data.

Prediction We use EMBEDALIGN to encode each candidate (in context) as a posterior Gaussian density. Note that this task dispenses with inferences about L2. Each candidate is compared to the target word in context through a measure of overlap between their inferred densities—we take KL divergence. We then rank candidates using this measure.

Table 7 lists GAP scores for variants of EMBEDALIN (bottom section) as well as some baselines and other established methods (top section). For comparison, we also compute GAP by sorting candidates in terms of cosine similarity, in which case we take the Gaussian mean as a summary of the density. The top section of the table contains systems reported by Melamud et al. (2015) (RANDOM and SKIPGRAM) and by Brazinskas et al. (2017) (BSG). Note that both SKIPGRAM (Mikolov et al., 2013) and BSG were trained on the very large ukWaC English corpus (Ferraresi et al., 2008). SKIPGRAM is known to perform remarkably well regardless of its apparent insensitivity to context (in terms of design). BSG is a close relative of our model which gives SKIPGRAM a Bayesian treatment (also by means of amortised variational inference) and is by design sensitive to context in a manner similar to EMBEDALIGN, that is, through its inferred posteriors.

Our first observation is that cosine seems to outperform KL slightly. Others have shown that KL can be used to predict directional entailment (Vilnis and McCallum, 2014; Brazinskas et al., 2017), since LST is closer to paraphrasing than to entailment directionality may be a distractor, but we

Model	MR	CR	SUBJ	MPQA	SST	TREC	MRPC	SICK-R	SICK-E	SST14
W2VEC	77.7	79.8	90.9	88.3	79.7	83.6	72.5/81.4	0.80	78.7	0.65/0.64
NMT	64.7	70.1	84.9	81.5	-	82.8	-/-	-	-	0.43/0.42
EN	57.6	66.2	70.9	71.8	58.0	62.9	70.3/80.1	0.62	73.7	0.54/0.55
EN-FR	63.5	71.5	78.9	82.3	65.1	62.1	71.4/80.5	0.69	75.9	0.69/0.59
EN-DE	64.0	68.9	77.9	81.8	65.1	59.5	71.2/80.5	0.69	74.8	0.62/0.61
COMBO	66.7	73.1	82.4	84.8	69.2	67.7	71.8/80.7	0.73	77.4	0.62/0.61

Table 8: English sentence evaluation results: the last four rows correspond to the mean of 10 runs with EMBEDALIGN models. All models, but W2VEC, employ bidirectional encoders.

leave it as a rather speculative point. One additional point worth highlighting: the middle section of Table 7. EN_{BoW} and EN_{BiRNN} show what happens when we do not give EMBEDALIGN `L2` supervision at training. That is, imagine the model of Figure 1 without the bottom plate. In that case, the model representations overfit for `L1` word-by-word prediction. Without the need to predict any notion of context (monolingual or otherwise), the representations drift away from semantic-driven generalisations and fail at lexical substitution.

3.3 Sentence Evaluation

Conneau et al. (2017) developed a framework to evaluate unsupervised sentence level representations trained on large amounts of data on a range of supervised NLP tasks. We assess our induced representations using their framework on the following benchmarks evaluated on classification ↑accuracy (MRPC is further evaluated on ↑F1)

MR classification of positive or negative movie reviews;

SST fined-grained labelling of movie reviews from the Stanford sentiment treebank (SST);

TREC classification of questions into k-classes;

CR classification of positive or negative product reviews;

SUBJ classification of a sentence into subjective or objective;

MPQA classification of opinion polarity;

SICK-E textual entailment classification;

MRPC paraphrase identification in the Microsoft paraphrase corpus;

as well as the following benchmarks evaluated on the indicated correlation metric(s)

SICK-R semantic relatedness between two sentences (↑Pearson);

SST-14 semantic textual similarity (↑Pearson/Spearman).

Prediction We use EMBEDALIGN to annotate every word in the training set of the benchmarks above with the posterior mean embedding in context. We then average embeddings in a sentence and give that as features to a logistic regression classifier trained with 5-fold cross validation.[6]

For comparison, we report a SKIPGRAM model (here indicated as W2VEC) as well as a model that uses the encoder of a neural machine translation system (NMT) trained on English-French Europarl data. In both cases, we report results by Conneau et al. (2017). Table 8 shows the results for all benchmarks.[7] We report EMBEDALIGN trained on either EN-FR or EN-DE. The last line (COMBO) shows what happens if we train logistic regression on the concatenation of embeddings inferred by both EMBEDALIGN models, that is, EN-FR and EN-DE. Note that these two systems perform sometimes better sometimes worse depending on the benchmark. There is no clear pattern, but differences may well come from some qualitative difference in the induced latent space. It is a known fact that different languages realise lexical ambiguities differently, thus representations induced towards different languages are likely to capture different generalisations.[8] As COMBO results show, the representations induced from different corpora are somewhat complementary. That same observation has guided paraphrasing models based on pivoting (Bannard and Callison-Burch, 2005). Once more we report a monolingual variant of EMBEDALIGN (indicated by EN) in an attempt to illustrate how crucial the

[6] `http://scikit-learn.org/stable/`

[7] In Appendix A we provide bar plots marked with error bars (2 standard deviations).

[8] We also acknowledge that our treatment of German is likely suboptimal due to the lack of subword features, as it can also be seen in AER results.

translation signal is.

3.4 Word similarity

Word similarity benchmarks are composed of word pairs which are manually ranked out of context. For completeness, we also tried evaluating our embeddings in such benchmarks despite our work being focussed on applications where context matters.

Prediction To assign an embedding for a word type, we infer Gaussian posteriors for all training instances of that type in context and aggregate the posterior means through an average (effectively collapsing all instances).

To cover the vocabulary of the typical benchmark, we have to use a much larger bilingual collection than Europarl. Based on the results of §3.1, we decided to proceed with English-French only—recall that models based on that pair performed better in terms of AER. Results in this section are based on EMBEDALIGN (with bidirectional variational encoder) trained on the Giga web corpus (see Table 1 for statistics). Due to the scale of the experiment, we report on a single run.

We trained on Giga with the same hyperparameters that we trained on Europarl, however, for 3 epochs instead of 30 (with this dataset an epoch amounts to $183,000$ updates). Again, we performed model selection on AER. Table 9 shows the results for several datasets using the framework of Faruqui and Dyer (2014a). Note that EMBEDALIGN was designed to make use of context information, thus this evaluation setup is a bit unnatural for our model. Still, it outperforms SKIPGRAM on 5 out of 13 benchmarks, in particular, on SIMLEX-999, whose relevance has been argued by Upadhyay et al. (2016). We also remark that this model achieves 0.25 test AER and 45.16 test GAP on lexical substitution—a considerable improvement compared to models trained on Europarl and reported in Tables 4 (AER) and 7 (GAP).

Dataset	SKIPGRAM	EMBEDALIGN
MTurk-771	0.5679	0.5229
SIMLEX-999	0.3131	0.3887
WS-353-ALL	0.6392	0.3968
YP-130	0.3992	0.4784
VERB-143	0.2728	0.4593
MEN-TR-3k	0.6462	0.4191
SimVerb-3500	0.2172	0.3539
RG-65	0.5384	0.6389
WS-353-SIM	0.6962	0.4509
RW-STANFORD	0.3878	0.3278
WS-353-REL	0.6094	0.3494
MC-30	0.6258	0.5559
MTurk-287	0.6698	0.3965

Table 9: Evaluation of English word embeddings out of context in terms of Spearman's rank correlation coefficient (↑). The first column is from (Faruqui and Dyer, 2014a).

4 Related work

Our model is inspired by lexical alignment models such as IBM1 (Brown et al., 1993), however, we generate words y_1^n from a latent vector representation z_1^m of x_1^m, rather than directly from the observation x_1^m. IBM1 takes `L1` sequences as conditioning context and does not model their distribution. Instead, we propose a joint model, where `L1` sentences are generated from latent embeddings.

There is a vast literature on exploiting multilingual context to strengthen the notion of synonymy captured by monolingual models. Roughly, the literature splits into two groups, namely, approaches that derive additional features and/or training objectives based on pre-trained alignments (Klementiev et al., 2012; Faruqui and Dyer, 2014b; Luong et al., 2015; Šuster et al., 2016), and approaches that promote a joint embedding space by working with sentence level representations that dispense with explicit alignments (Hermann and Blunsom, 2014; AP et al., 2014; Gouws et al., 2015; Hill et al., 2014).

The work of Kočiský et al. (2014) is closer to ours in that they also learn embeddings by marginalising alignments, however, their model is conditional—much like IBM models—and their embeddings are not part of the probabilistic model, but rather part of the architecture design. The joint formulation allows our latent embeddings to harvest learning signal from `L2` while still being driven by the learning signal from `L1`—in a conditional model the representations can become specific to alignment deviating from the purpose of well representing the original language. In §3 we show substantial evidence that our model performs better when using both learning signals.

Vilnis and McCallum (2014) first propose to map words into Gaussian densities instead of point estimates for better word representation. For example, a distribution can capture asymmetric relations that

a point estimate cannot. Brazinskas et al. (2017) recast the skip-gram model as a conditional variational auto-encoder. They induce a Gaussian density for each occurrence of a word in context, and for that their model is the closest to ours. Additionally, they estimate a Gaussian prior per word type thus representing both types and occurrences. Unlike our model, the Bayesian skip-gram is not trained generatively by reconstructing the data, but rather discriminatively by prediction of overlapping sets of neighbouring words.

5 Discussion

We have presented a generative model of word representation that learns from positive correlations implicitly expressed in translation data. In order to make these correlations surface, we induce and marginalise latent lexical alignments.

Embedding models such as CBOW and skip-gram (Mikolov et al., 2013) are essentially speaking supervised classifiers. This means they depend on somewhat artificial strategies to derive labelled data from monolingual corpora—words far from the central word still have co-occurred with it even though they are taken as negative evidence. Training our proposed model does not require a heuristic notion of negative training data. However, the model is also based on a somewhat artificial assumption: `L1` words do not necessarily need to have an `L2` equivalent and, even when they do, this equivalent need not be realised as a single word.

We have shown with extensive experiments that our model can induce representations useful to several tasks including but not limited to alignment (the task it most obviously relates to). We observed interesting results on semantic natural language processing benchmarks such as natural language inference, lexical substitution, paraphrasing, and sentiment classification.

We are currently expanding the notion of distributional context to multiple auxiliary foreign languages at once. This seems to only require minor changes to the generative story and could increase the model's disambiguation power dramatically. Another direction worth exploring is to extend the model's hierarchy with respect to how parallel sentences are generated. For example, modelling sentence level latent variables may capture global constraints and expose additional correlations to the model.

Acknowledgments

We thank Philip Schulz for comments on an earlier version of this paper as well as the anonymous NAACL reviewers. One of the Titan Xp cards used for this research was donated by the NVIDIA Corporation. This work was supported by the Dutch Organization for Scientific Research (NWO) VICI Grant nr. 277-89-002.

References

Sarath Chandar AP, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *Advances in Neural Information Processing Systems*. pages 1853–1861.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '05, pages 597–604. https://doi.org/10.3115/1219840.1219914.

Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. 2015. Automatic differentiation in machine learning: a survey. *arXiv preprint arXiv:1502.05767* .

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Baltimore, Maryland, USA, pages 12–58. http://www.aclweb.org/anthology/W/W14/W14-3302.

Aleksandar Botev, Bowen Zheng, and David Barber. 2017. Complementary sum sampling for likelihood approximation in large scale classification. In *Artificial Intelligence and Statistics*. pages 1030–1038.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 632–642. http://aclweb.org/anthology/D15-1075.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*. pages 10–21.

Arthur Brazinskas, Serhii Havrylov, and Ivan Titov. 2017. Embedding words as distributions with a bayesian skip-gram model. *Arxiv: 1711.11027* .

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics* 19(2):263–311.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.* 12:2493–2537. http://dl.acm.org/citation.cfm?id=1953048.2078186.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364* .

Mona Diab and Philip Resnik. 2002. An unsupervised method for word sense tagging using parallel corpora. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, pages 255–262. https://doi.org/10.3115/1073083.1073126.

Manaal Faruqui and Chris Dyer. 2014a. Community evaluation and exchange of word vectors at wordvectors.org. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, Baltimore, USA.

Manaal Faruqui and Chris Dyer. 2014b. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Gothenburg, Sweden, pages 462–471. http://www.aclweb.org/anthology/E14-1049.

Adriano Ferraresi, Eros Zanchetta, Marco Baroni, and Silvia Bernardini. 2008. Introducing and evaluating ukwac, a very large web-derived corpus of english. In *In Proceedings of the 4th Web as Corpus Workshop (WAC-4*.

J. R. Firth. 1957. A synopsis of linguistic theory 1930-1955. *Studies in Linguistic Analysis* pages 1–32.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. PMLR, Chia Laguna Resort, Sardinia, Italy, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256. http://proceedings.mlr.press/v9/glorot10a.html.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. Bilbowa: Fast bilingual distributed representations without word alignments. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*. PMLR, Lille, France, volume 37 of *Proceedings of Machine Learning Research*, pages 748–756. http://proceedings.mlr.press/v37/gouws15.html.

Zellig S. Harris. 1954. Distributional structure. *Word* 10(23):146–162.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. In *Proceedings of*

the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, Baltimore, Maryland, pages 58–68. http://www.aclweb.org/anthology/P14-1006.

Felix Hill, Kyunghyun Cho, Sebastien Jean, Coline Devin, and Yoshua Bengio. 2014. Embedding word similarity with neural machine translation. *arXiv preprint arXiv:1412.6448* .

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

MichaelI. Jordan, Zoubin Ghahramani, TommiS. Jaakkola, and LawrenceK. Saul. 1999. An introduction to variational methods for graphical models. *Machine Learning* 37(2):183–233.

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

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *International Conference on Learning Representations*.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *Proceedings of COLING 2012*. The COLING 2012 Organizing Committee, Mumbai, India, pages 1459–1474. http://www.aclweb.org/anthology/C12-1089.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*. AAMT, AAMT, Phuket, Thailand, pages 79–86. http://mt-archive.info/MTS-2005-Koehn.pdf.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '07, pages 177–180. http://dl.acm.org/citation.cfm?id=1557769.1557821.

Tomáš Kočiský, Karl Moritz Hermann, and Phil Blunsom. 2014. Learning Bilingual Word Representations by Marginalizing Alignments. In *Proceedings of ACL*.

Omer Levy and Yoav Goldberg. 2014a. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 302–308. http://www.aclweb.org/anthology/P14-2050.

Omer Levy and Yoav Goldberg. 2014b. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Ann Arbor, Michigan, pages 171–180. http://www.aclweb.org/anthology/W14-1618.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*. pages 151–159.

Diana McCarthy and Roberto Navigli. 2009. The english lexical substitution task. *Language Resources and Evaluation* 43(2):139–159.

Oren Melamud, Omer Levy, and Ido Dagan. 2015. A simple word embedding model for lexical substitution. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing, VS@NAACL-HLT 2015, June 5, 2015, Denver, Colorado, USA*. pages 1–7.

Rada Mihalcea and Ted Pedersen. 2003. An evaluation exercise for word alignment. In *Proceedings of the HLT-NAACL 2003 Workshop on Building and Using Parallel Texts: Data Driven Machine Translation and Beyond - Volume 3*. Association for

Computational Linguistics, Stroudsburg, PA, USA, HLT-NAACL-PARALLEL '03, pages 1–10. https://doi.org/10.3115/1118905.1118906.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., pages 3111–3119.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *38th Annual Meeting of the Association for Computational Linguistics, Hong Kong, China, October 1-8, 2000.*.

Sebastian Padó and Mirella Lapata. 2006. Optimal constituent alignment with edge covers for semantic projection. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Sydney, Australia, pages 1161–1168. https://doi.org/10.3115/1220175.1220321.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*. pages 1278–1286.

Herbert Robbins and Sutton Monro. 1951. A stochastic approximation method. *The Annals of Mathematical Statistics* 22(3):400–407.

Philip Schulz, Wilker Aziz, and Khalil Sima'an. 2016. Word alignment without null words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 169–174. http://anthology.aclweb.org/P16-2028.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Edinburgh, Scotland, UK., pages 151–161. http://www.aclweb.org/anthology/D11-1014.

Michalis Titsias and Miguel Lázaro-Gredilla. 2014. Doubly stochastic variational bayes for non-conjugate inference. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. pages 1971–1979.

Lifu Tu, Kevin Gimpel, and Karen Livescu. 2017. Learning to embed words in context for syntactic tasks. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*. Association for Computational Linguistics, Vancouver, Canada, pages 265–275. http://www.aclweb.org/anthology/W17-2632.

Shyam Upadhyay, Manaal Faruqui, Chris Dyer, and Dan Roth. 2016. Cross-lingual models of word embeddings: An empirical comparison. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1661–1670. http://www.aclweb.org/anthology/P16-1157.

Luke Vilnis and Andrew McCallum. 2014. Word representations via gaussian embedding. *arXiv preprint arXiv:1412.6623* .

Simon Šuster, Ivan Titov, and Gertjan van Noord. 2016. Bilingual learning of multi-sense embeddings with discrete autoencoders. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 1346–1356. http://www.aclweb.org/anthology/N16-1160.

A Multiple runs sentence evaluation

Figure 2 shows multiple runs of our proposed model on sentence evaluation. The first figure reports the mean and two standard deviations (error bars) for benchmarks based on accuracy (ACC), the second figure reports benchmarks based on F1, and finally the third figure reports benchmarks based on correlation metrics Spearman (S) and Pearson (P).

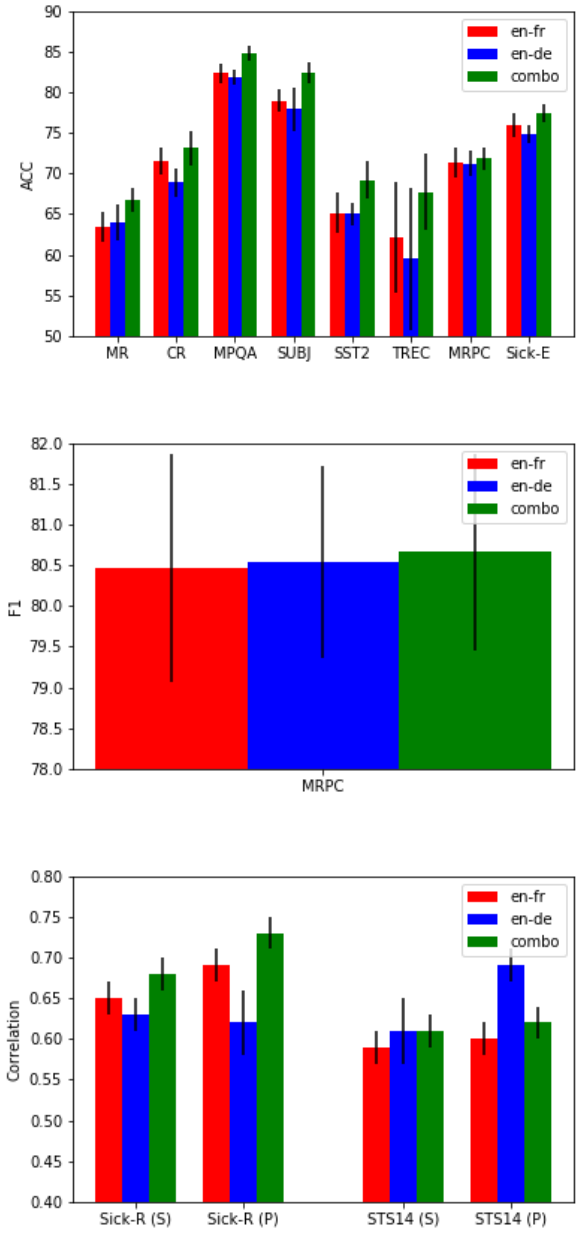


Figure 2: Mean and two standard deviations (error bars) for 10 runs of EMBEDALIGN on the sentence evaluation benchmarks.

B Architecture

Figure 3 shows the architecture for the inference and generative models in EMBEDALIGN, with BiRNN encoder (h).

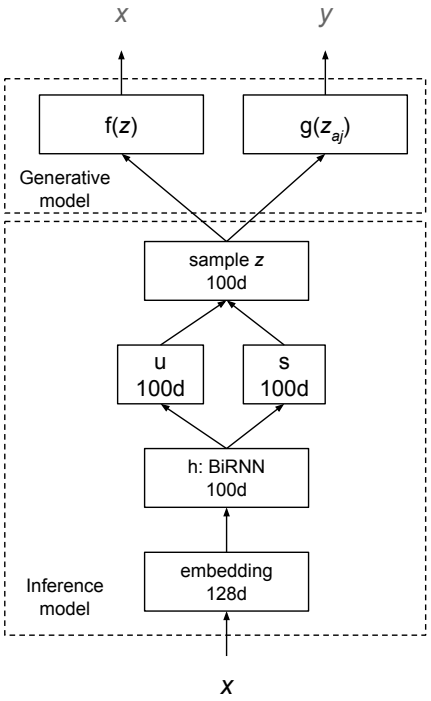


Figure 3: Architecture for EmbedAlign.

Learning Word Embeddings for Low-resource Languages by PU Learning

Chao Jiang
University of Virginia
cj7an@virginia.edu

Hsiang-Fu Yu
Amazon
rofuyu@cs.utexas.edu

Cho-Jui Hsieh
University of California Davis
chohsieh@ucdavis.edu

Kai-Wei Chang
University of California Los Angeles
kwchang@cs.ucla.edu

Abstract

Word embedding is a key component in many downstream applications in processing natural languages. Existing approaches often assume the existence of a large collection of text for learning effective word embedding. However, such a corpus may not be available for some low-resource languages. In this paper, we study how to effectively learn a word embedding model on a corpus with only a few million tokens. In such a situation, the co-occurrence matrix is sparse as the co-occurrences of many word pairs are unobserved. In contrast to existing approaches often only sample a few unobserved word pairs as negative samples, we argue that the zero entries in the co-occurrence matrix also provide valuable information. We then design a Positive-Unlabeled Learning (PU-Learning) approach to factorize the co-occurrence matrix and validate the proposed approaches in four different languages.

1 Introduction

Learning word representations has become a fundamental problem in processing natural languages. These semantic representations, which map a word into a point in a linear space, have been widely applied in downstream applications, including named entity recognition (Guo et al., 2014), document ranking (Nalisnick et al., 2016), sentiment analysis (Irsoy and Cardie, 2014), question answering (Antol et al., 2015), and image captioning (Karpathy and Fei-Fei, 2015).

Over the past few years, various approaches have been proposed to learn word vectors (e.g., (Pennington et al., 2014; Mikolov et al., 2013a; Levy and Goldberg, 2014b; Ji et al., 2015)) based on co-occurrence information between words observed on the training corpus. The intuition behind this is to represent words with similar vectors if they have similar contexts. To learn a good word embedding, most approaches assume a large collection of text is freely available, such that the estimation of word co-occurrences is accurate. For example, the Google Word2Vec model (Mikolov et al., 2013a) is trained on the Google News dataset, which contains around 100 billion tokens, and the GloVe embedding (Pennington et al., 2014) is trained on a crawled corpus that contains 840 billion tokens in total. However, such an assumption may not hold for low-resource languages such as Inuit or Sindhi, which are not spoken by many people or have not been put into a digital format. For those languages, usually, only a limited size corpus is available. Training word vectors under such a setting is a challenging problem.

One key restriction of the existing approaches is that they often mainly rely on the word pairs that are observed to co-occur on the training data. When the size of the text corpus is small, most word pairs are unobserved, resulting in an extremely sparse co-occurrence matrix (i.e., most entries are zero)[1]. For example, the text8[2] corpus has about 17,000,000 tokens and 71,000 distinct words. The corresponding co-occurrence matrix has more than five billion entries, but only about 45,000,000 are non-zeros (observed on the training corpus). Most existing approaches, such as Glove and Skip-gram, cannot handle a vast number of zero terms in the co-occurrence matrix; therefore, they only sub-sample a small subset of zero entries during the training.

In contrast, we argue that the *unobserved* word pairs can provide valuable information for training a word embedding model, especially when the co-occurrence matrix is very sparse. Inspired

[1]Note that the zero term can mean either the pairs of words cannot co-occur or the co-occurrence is not observed in the training corpus.

[2]http://mattmahoney.net/dc/text8.zip

Proceedings of NAACL-HLT 2018, pages 1024–1034
New Orleans, Louisiana, June 1 - 6, 2018.

by the success of Positive-Unlabeled Learning (PU-Learning) in collaborative filtering applications (Pan et al., 2008; Hu et al., 2008; Pan and Scholz, 2009; Qin et al., 2010; Paquet and Koenigstein, 2013; Hsieh et al., 2015), we design an algorithm to effectively learn word embeddings from both positive (observed terms) and unlabeled (unobserved/zero terms) examples. Essentially, by using the square loss to model the unobserved terms and designing an efficient update rule based on linear algebra operations, the proposed PU-Learning framework can be trained efficiently and effectively.

We evaluate the performance of the proposed approach in English[3] and other three resource-scarce languages. We collected unlabeled language corpora from Wikipedia and compared the proposed approach with popular approaches, the Glove and the Skip-gram models, for training word embeddings. The experimental results show that our approach significantly outperforms the baseline models, especially when the size of the training corpus is small.

Our key contributions are summarized below.

- We propose a PU-Learning framework for learning word embedding.
- We tailor the coordinate descent algorithm (Yu et al., 2017b) for solving the corresponding optimization problem.
- Our experimental results show that PU-Learning improves the word embedding training in the low-resource setting.

2 Related work

Learning word vectors. The idea of learning word representations can be traced back to Latent Semantic Analysis (LSA) (Deerwester et al., 1990) and Hyperspace Analogue to Language (HAL) (Lund and Burgess, 1996), where word vectors are generated by factorizing a word-document and word-word co-occurrence matrix, respectively. Similar approaches can also be extended to learn other types of relations between words (Yih et al., 2012; Chang et al., 2013) or entities (Chang et al., 2014). However, due to the limitation of the use of principal component analysis, these approaches are often less flexible. Besides, directly factorizing the co-occurrence matrix may cause the frequent words dominating the training objective.

In the past decade, various approaches have been proposed to improve the training of word embeddings. For example, instead of factorizing the co-occurrence count matrix, Bullinaria and Levy (2007); Levy and Goldberg (2014b) proposed to factorize *point-wise mutual information* (PMI) and *positive PMI* (PPMI) matrices as these metrics scale the co-occurrence counts (Bullinaria and Levy, 2007; Levy and Goldberg, 2014b). Skip-gram model with negative-sampling (SGNS) and Continuous Bag-of-Words models (Mikolov et al., 2013b) were proposed for training word vectors on a large scale without consuming a large amount of memory. GloVe (Pennington et al., 2014) is proposed as an alternative to decompose a weighted log co-occurrence matrix with a bias term added to each word. Very recently, *WordRank* model (Ji et al., 2015) has been proposed to minimize a ranking loss which naturally fits the tasks requiring ranking based evaluation metrics. Stratos et al. (2015) also proposed CCA (canonical correlation analysis)-based word embedding which shows competitive performance. All these approaches focus on the situations where a large text corpus is available.

Positive and Unlabeled (PU) Learning: Positive and Unlabeled (PU) learning (Li and Liu, 2005) is proposed for training a model when the positive instances are partially labeled and the unlabeled instances are mostly negative. Recently, PU learning has been used in many classification and collaborative filtering applications due to the nature of "implicit feedback" in many recommendation systems—users usually only provide positive feedback (e.g., purchases, clicks) and it is very hard to collect negative feedback.

To resolve this problem, a series of PU matrix completion algorithms have been proposed (Pan et al., 2008; Hu et al., 2008; Pan and Scholz, 2009; Qin et al., 2010; Paquet and Koenigstein, 2013; Hsieh et al., 2015; Yu et al., 2017b). The main idea is to assign a small uniform weight to all the missing or zero entries and factorize the corresponding matrix. Among them, Yu et al. (2017b) proposed an efficient algorithm for matrix factorization with PU-learning, such that the weighted matrix is constructed implicitly. In this paper, we

[3] Although English is not a resource-scarce language, we simulate the low-resource setting in an English corpus. In this way, we leverage the existing evaluation methods to evaluate the proposed approach.

$\mathcal{W}, \mathcal{C}$	vocabulary of central and context words
m, n	vocabulary sizes
k	dimension of word vectors
W, H	$m \times k$ and $n \times k$ latent matrices
C_{ij}	weight for the (i, j) entry
A_{ij}	value of the PPMI matrix
Q_{ij}	value of the co-occurrence matrix
$\boldsymbol{w}_i, \boldsymbol{h}_j$	i-th row of W and j-th row of H
$\boldsymbol{b}, \hat{\boldsymbol{b}}$	bias term
$\lambda_i, \overline{\lambda}_j$	regularization parameters
$\lvert \cdot \rvert$	the size of a set
Ω	Set of possible word-context pairs
Ω^+	Set of observed word-context pairs
Ω^-	Set of unobserved word-context pairs

Table 1: Notations.

design a new approach for training word vectors by leveraging the PU-Learning framework and existing word embedding techniques. To the best of our knowledge, this is the first work to train word embedding models using the PU-learning framework.

3 PU-Learning for Word Embedding

Similar to GloVe and other word embedding learning algorithms, the proposed approach consists of three steps. The first step is to construct a co-occurrence matrix. Follow the literature (Levy and Goldberg, 2014a), we use the PPMI metric to measure the co-occurrence between words. Then, in the second step, a PU-Learning approach is applied to factorize the co-occurrence matrix and generate word vectors and context vectors. Finally, a post-processing step generates the final embedding vector for each word by combining the word vector and the context vector.

We summarize the notations used in this paper in Table 1 and describe the details of each step in the remainder of this section.

3.1 Building the Co-Occurrence Matrix

Various metrics can be used for estimating the co-occurrence between words in a corpus. PPMI metric stems from *point-wise mutual information* (PMI) which has been widely used as a measure of word association in NLP for various tasks (Church and Hanks, 1990). In our case, each entry $PMI(w, c)$ represents the relevant measure between a word w and a context word c by calculating the ratio between their joint probability (the chance they appear together in a local context window) and their marginal probabilities (the chance they appear independently) (Levy and Goldberg, 2014b). More specifically, each entry of PMI matrix can be defined by

$$PMI(w, c) = \log \frac{\hat{P}(w, c)}{\hat{P}(w) \cdot \hat{P}(c)}, \qquad (1)$$

where $\hat{P}(w), \hat{P}(c)$ and $\hat{P}(w, c)$ are the the frequency of word w, word c, and word pairs (w, c), respectively. The PMI matrix can be computed based on the co-occurrence counts of word pairs, and it is an information-theoretic association measure which effectively eliminates the big differences in magnitude among entries in the co-occurrence matrix.

Extending from the PMI metric, the PPMI metric replaces all the negative entries in PMI matrix by 0:

$$PPMI(w, c) = \max(PMI(w, c), 0). \qquad (2)$$

The intuition behind this is that people usually perceive positive associations between words (e.g. "ice" and "snow"). In contrast, the negative association is hard to define (Levy and Goldberg, 2014b). Therefore, it is reasonable to replace the negative entries in the PMI matrix by 0, such that the negative association is treated as "uninformative". Empirically, several existing works (Levy et al., 2015; Bullinaria and Levy, 2007) showed that the PPMI metric achieves good performance on various semantic similarity tasks.

In practice, we follow the pipeline described in Levy et al. (2015) to build the PPMI matrix and apply several useful tricks to improve its quality. First, we apply a context distribution smoothing mechanism to enlarge the probability of sampling a rare context. In particular, all context counts are scaled to the power of α.[4]:

$$PPMI_\alpha(w, c) = \max\left(\log \frac{\hat{P}(w, c)}{\hat{P}(w)\hat{P}_\alpha(c)}, 0\right)$$

$$\hat{P}_\alpha(c) = \frac{\#(c)^\alpha}{\sum_{\bar{c}} \#(\bar{c})^\alpha},$$

where $\#(w)$ denotes the number of times word w appears. This smoothing mechanism effectively

[4]Empirically, $\alpha = 0.75$ works well (Mikolov et al., 2013b).

alleviates PPMI's bias towards rare words (Levy et al., 2015).

Next, previous studies show that words that occur too frequent often dominate the training objective (Levy et al., 2015) and degrade the performance of word embedding. To avoid this issue, we follow Levy et al. (2015) to sub-sample words with frequency more than a threshold t with a probability p defined as:

$$p = 1 - \sqrt{\frac{t}{\hat{P}(w)}}.$$

3.2 PU-Learning for Matrix Factorization

We proposed a matrix factorization based word embedding model which aims to minimize the reconstruction error on the PPMI matrix. The low-rank embeddings are obtained by solving the following optimization problem:

$$\min_{W,H} \sum_{i,j \in \Omega} C_{ij}(A_{ij} - \boldsymbol{w}_i^T \boldsymbol{h}_j - b_i - \hat{b}_j)^2 + \sum_i \lambda_i \|\boldsymbol{w}_i\|^2 + \sum_j \overline{\lambda}_j \|\boldsymbol{h}_j\|^2, \quad (3)$$

where W and H are $m \times k$ and $n \times k$ latent matrices, representing words and context words, respectively. The first term in Eq. (3) aims for minimizing reconstruction error, and the second and third terms are regularization terms. λ_i and $\overline{\lambda}_j$ are weights of regularization term. They are hyper-parameters that need to be tuned.

The zero entries in co-occurrence matrix denote that two words never appear together in the current corpus, which also refers to unobserved terms. The unobserved term can be either real zero (two words shouldn't be co-occurred even when we use very large corpus) or just missing in the small corpus. In contrast to SGNS sub-sampling a small set of zero entries as negative samples, our model will try to use the information from all zeros.

The set Ω includes all the $|\mathcal{W}| \times |\mathcal{C}|$ entries—both positive and zero entries:

$$\Omega = \Omega^+ \cup \Omega^-. \quad (4)$$

Note that we define the positive samples Ω^+ to be all the (w, c) pairs that appear at least one time in the corpus, and negative samples Ω^- are word pairs that never appear in the corpus.

Weighting function. Eq (3) is very similar to the one used in previous matrix factorization approaches such as GloVe, but we propose a new way to set the weights C_{ij}. If we set equal weights for all the entries, then C_{ij} = constant, and the model is very similar to conducting SVD for the PPMI matrix. Previous work has shown that this approach often suffers from poor performance (Pennington et al., 2014). More advanced methods, such as GloVe, set non-uniform weights for observed entries to reflect their confidence. However, the time complexity of their algorithm is proportional to number of nonzero weights ($|(i,j) \mid C_{ij} \neq 0|$), thus they have to set zero weights for all the unobserved entries ($C_{ij} = 0$ for Ω^-), or try to incorporate a small set of unobserved entries by negative sampling.

We propose to set the weights for Ω^+ and Ω^- differently using the following scheme:

$$C_{ij} = \begin{cases} (Q_{ij}/x_{max})^\alpha, & \text{if } Q_{ij} \leq x_{max}, \text{ and } (i,j) \in \Omega^+ \\ 1, & \text{if } Q_{ij} > x_{max}, \text{ and } (i,j) \in \Omega^+ \\ \rho, & (i,j) \in \Omega^- \end{cases} \quad (5)$$

Here x_{max} and α are re-weighting parameters, and ρ is the unified weight for unobserved terms. We will discuss them later.

For entries in Ω^+, we set the non-uniform weights as in GloVe (Pennington et al., 2014), which assigns larger weights to context word that appears more often with the given word, but also avoids overwhelming the other terms. For entries in Ω^-, instead of setting their weights to be 0, we assign a small constant weight ρ. The main idea is from the literature of PU-learning (Hu et al., 2008; Hsieh et al., 2015): although missing entries are highly uncertain, they are still likely to be true 0, so we should incorporate them in the learning process but multiplying with a smaller weight according to the uncertainty. Therefore, ρ in (5) reflects how confident we are to the zero entries.

In our experiments, we set $x_{max} = 10, \alpha = 3/4$ according to (Pennington et al., 2014), and let ρ be a parameter to tune. Experiments show that adding weighting function obviously improves the performance especially on analogy tasks.

Bias term. Unlike previous work on PU matrix completion (Yu et al., 2017b; Hsieh et al., 2015), we add the bias terms for word and context word

vectors. Instead of directly using $\boldsymbol{w}_i^\top \boldsymbol{h}_j$ to approximate A_{ij}, we use

$$A_{ij} \approx \boldsymbol{w}_i^\top \boldsymbol{h}_j + b_i + \hat{b}_j.$$

Yu et al. (2017b) design an efficient column-wise coordinate descent algorithm for solving the PU matrix factorization problem; however, they do not consider the bias term in their implementations. To incorporate the bias term in (3), we propose the following training algorithm based on the coordinate descent approach. Our algorithm does not introduce much overhead compared to that in (Yu et al., 2017b).

We augment each $\boldsymbol{w}_i, \boldsymbol{h}_j \in R^k$ into the following $(k+2)$ dimensional vectors:

$$\boldsymbol{w}'_i = \begin{bmatrix} w_{i1} \\ \vdots \\ w_{ik} \\ 1 \\ b_i \end{bmatrix} \quad \boldsymbol{h}'_j = \begin{bmatrix} h_{j1} \\ \vdots \\ h_{jk} \\ \hat{b}_j \\ 1 \end{bmatrix}$$

Therefore, for each word and context vector, we have the following equality

$$\langle \boldsymbol{w}'_i, \boldsymbol{h}'_j \rangle = \langle \boldsymbol{w}_i, \boldsymbol{h}_j \rangle + b_i + \hat{b}_j,$$

which means the loss function in (3) can be written as

$$\sum_{i,j \in \Omega} C_{ij} (A_{ij} - \boldsymbol{w}_i'^\top \boldsymbol{h}'_j)^2.$$

Also, we denote $W' = [\boldsymbol{w}'_1, \boldsymbol{w}'_2, \ldots, \boldsymbol{w}'_n]^\top$ and $H' = [\boldsymbol{h}'_1, \boldsymbol{h}'_2, \ldots, \boldsymbol{h}'_n]^\top$. In the column-wise coordinate descent method, at each iteration we pick a $t \in \{1, \ldots, (k+2)\}$, and update the t-th column of W' and H'. The updates can be derived for the following two cases:

a. When $t \leq k$, the elements in the t-th column is $w_{1t}, \ldots, w_{nt}$ and we can directly use the update rule derived in Yu et al. (2017b) to update them.

b. When $t = k+1$, we do not update the corresponding column of W' since the elements are all 1, and we use the similar coordinate descent update to update the $k+1$-th column of H' (corresponding to $\hat{b}_1, \ldots, \hat{b}_n$). When $t = k+2$, we do not update the corresponding column of H' (they are all 1) and we update the $k+2$-th column of W' (corresponding to $b_1, \ldots, b_n$) using coordinate descent.

With some further derivations, we can show that the algorithm only requires $O(\text{nnz}(A) + nk)$ time to update each column,[5] so the overall complexity is $O(\text{nnz}(A)k + nk^2)$ time per epoch, which is only proportional to number of nonzero terms in A. Therefore, with the same time complexity as GloVe, we can utilize the information from all the zero entries in A instead of only sub-sampling a small set of zero entries.

3.3 Interpretation of Parameters

In the PU-Learning formulation, ρ represents the unified weight that assigned to the unobserved terms. Intuitively, ρ reflects the confidence on unobserved entries—larger ρ means that we are quite certain about the zeroes, while small ρ indicates the many of unobserved pairs are not truly zero. When $\rho = 0$, the PU-Learning approach reduces to a model similar to GloVe, which discards all the unobserved terms. In practice, ρ is an important parameter to tune, and we find that $\rho = 0.0625$ achieves the best results in general. Regarding the other parameter, λ is the regularization term for preventing the embedding model from overfitting. In practice, we found the performance is not very sensitive to λ as long as it is resonably small. More discussion about the parameter setting can be found in Section 5.

Post-processing of Word/Context Vectors The PU-Learning framework factorizes the PPMI matrix and generates two vectors for each word i, $\boldsymbol{w}_i \in R^k$ and $\boldsymbol{h}_i \in R^k$. The former represents the word when it is the central word and the latter represents the word when it is in context. Levy et al. (2015) shows that averaging these two vectors ($\boldsymbol{u}_i^{\text{avg}} = \boldsymbol{w}_i + \boldsymbol{h}_i$) leads to consistently better performance. The same trick of constructing word vectors is also used in GloVe. Therefore, in the experiments, we evaluate all models with $\boldsymbol{u}^{\text{avg}}$.

4 Experimental Setup

Our goal in this paper is to train word embedding models for low-resource languages. In this section, we describe the experimental designs to evaluate the proposed PU-learning approach. We first describe the data sets and the evaluation metrics. Then, we provide details of parameter tuning.

[5] Here we assume $m = n$ for the sake of simplicity. And, nnz(A) denotes the number of nonzero terms in the matrix A.

	Similarity task					Analogy task	
Word embedding	WS353	Similarity	Relatedness	M. Turk	MEN	3CosAdd	3CosMul
GloVe	48.7	50.9	53.7	54.1	17.6	32.1	28.5
SGNS	67.2	70.3	67.9	**59.9***	**25.1***	30.4	27.8
PU-learning	**68.3***	**71.8***	**68.2***	57.0	22.7	**32.6***	**32.3***

Table 2: Performance of the best SGNS, GloVe, PU-Learning models, trained on the text8 corpus. Results show that our proposed model is better than SGNS and GloVe. Star indicates it is significantly better than the second best algorithm in the same column according to Wilcoxon signed-rank test. ($p < 0.05$)

	Similarity task					Analogy task
Language	WS353	Similarity	Relatedness	M. Turk	MEN	Google
English (en)	353	203	252	287	3,000	19,544
Czech (cs)	337	193	241	268	2,810	18,650
Danish (da)	346	198	247	283	2,951	18,340
Dutch (nl)	346	200	247	279	2,852	17,684

Table 3: The size of the test sets. The data sets in English are the original test sets. To evaluate other languages, we translate the data sets from English.

4.1 Evaluation tasks

We consider two widely used tasks for evaluating word embeddings, the word similarity task and the word analogy task. In the word similarity task, each question contains a word pairs and an annotated similarity score. The goal is to predict the similarity score between two words based on the inner product between the corresponding word vectors. The performance is then measured by the Spearmans rank correlation coefficient, which estimates the correlation between the model predictions and human annotations. Following the settings in literature, the experiments are conducted on five data sets, *WordSim353* (Finkelstein et al., 2001), *WordSim Similarity* (Zesch et al., 2008), *WordSim Relatedness* (Agirre et al., 2009), *Mechanical Turk* (Radinsky et al., 2011) and *MEN* (Bruni et al., 2012).

In the word analogy task, we aim at solving analogy puzzles like "man is to woman as king is to ?", where the expected answer is "queen." We consider two approaches for generating answers to the puzzles, namely 3CosAdd and 3CosMul (see (Levy and Goldberg, 2014a) for details). We evaluate the performances on *Google analogy* dataset (Mikolov et al., 2013a) which contains 8,860 semantic and 10,675 syntactic questions. For the analogy task, only the answer that exactly matches the annotated answer is counted as correct. As a result, the analogy task is more difficult than the similarity task because the evaluation metric is stricter and it requires algorithms to differentiate words with similar meaning and find the right answer.

To evaluate the performances of models in the low-resource setting, we train word embedding models on Dutch, Danish, Czech and, English data sets collected from Wikipedia. The original Wikipedia corpora in Dutch, Danish, Czech and English contain 216 million, 47 million, 92 million, and 1.8 billion tokens, respectively. To simulate the low-resource setting, we sub-sample the Wikipedia corpora and create a subset of 64 million tokens for Dutch and Czech and a subset of 32 million tokens for English. We will demonstrate how the size of the corpus affects the performance of embedding models in the experiments.

To evaluate the performance of word embeddings in Czech, Danish, and Dutch, we translate the English similarity and analogy test sets to the other languages by using Google Cloud Translation API[6]. However, an English word may be translated to multiple words in another language (e.g., compound nouns). We discard questions containing such words (see Table 3 for details). Because all approaches are compared on the same test set for each language, the comparisons are fair.

4.2 Implementation and Parameter Setting

We compare the proposed approach with two baseline methods, GloVe and SGNS. The imple-

[6]https://cloud.google.com/translate

Dutch (nl)	Similarity task					Analogy task	
Word embedding	WS353	Similarity	Relatedness	M. Turk	MEN	3CosAdd	3CosMul
GloVe	35.4	35.0	41.7	44.3	11	21.2	20.2
SGNS	51.9	52.9	53.5	**49.8***	15.4	22.1	23.6
PU-learning	**53.7***	**53.4***	**55.1***	46.7	**16.4***	**23.5***	**24.7***
Danish (da)	Similarity task					Analogy task	
Word embedding	WS353	Similarity	Relatedness	M. Turk	MEN	3CosAdd	3CosMul
GloVe	25.7	18.4	40.3	49.0	16.4	**25.8***	**24.3***
SGNS	49.7	47.1	52.1	51.5	22.4	22.0	21.2
PU-learning	**53.5***	**49.5***	**59.3***	**51.7***	**22.7***	22.6	22.8
Czech (cs)	Similarity task					Analogy task	
Word embedding	WS353	Similarity	Relatedness	M. Turk	MEN	3CosAdd	3CosMul
GloVe	34.3	23.2	48.9	36.5	16.2	8.9	8.6
SGNS	51.4	42.7	61.1	44.2	21.3	**10.4***	9.8
PU-learning	**54.0***	**45.4***	**65.3***	**46.2***	**21.7***	9.9	**10.1***
English (en)	Similarity task					Analogy task	
Word embedding	WS353	Similarity	Relatedness	M. Turk	MEN	3CosAdd	3CosMul
GloVe	47.9	52.1	49.5	58.8	19.1	34.3	32.6
SGNS	65.7	**67.1***	66.5	**62.8***	**26.1***	31.2	27.4
PU-learning	**67.0***	66.7	**69.6***	59.4	22.4	**39.2***	**38.8***

Table 4: Performance of SGNS, GloVe, and the proposed PU-Learning model in four different languages. Results show that the proposed PU-Learning model outperforms SGNS and GloVe in most cases when the size of corpus is relatively small (around 50 million tokens). Star indicates it is significant better than the second best algorithm in the same column according to Wilcoxon signed-rank test. ($p < 0.05$).

mentations of Glove[7] and SGNS[8] and provided by the original authors, and we apply the default settings when appropriate. The proposed PU-Learning framework is implemented based on Yu et al. (2017a). With the implementation of efficient update rules, our model requires less than 500 seconds to perform one iteration over the entire text8 corpus, which consists of 17 million tokens [9]. All the models are implemented in C++.

We follow Levy et al. (2015)[10] to set windows size as 15, minimal count as 5, and dimension of word vectors as 300 in the experiments. Training word embedding models involves selecting several hyper-parameters. However, as the word embeddings are usually evaluated in an unsupervised setting (i.e., the evaluation data sets are not seen during the training), the parameters should not be tuned on each dataset. To conduct a fair comparison, we tune hyper-parameters on the text8 dataset. For GloVe model, we tune the discount parameters x_{max} and find that $x_{max} = 10$ performs the best. SGNS has a natural parameter k which denotes the number of negative samples. Same as Levy et al. (2015), we found that setting k to 5 leads to the best performance. For the PU-learning model, ρ and λ are two important parameters that denote the unified weight of zero entries and the weight of regularization terms, respectively. We tune ρ in a range from 2^{-1} to 2^{-14} and λ in a range from 2^0 to 2^{-10}. We analyze the sensitivity of the model to these hyper-parameters in the experimental result section. The best performance of each model on the text8 dataset is shown in the Table 2. It shows that PU-learning model outperforms two baseline models.

[7] https://nlp.stanford.edu/projects/glove
[8] https://code.google.com/archive/p/word2vec/
[9] http://mattmahoney.net/dc/text8.zip
[10] https://bitbucket.org/omerlevy/hyperwords

5 Experimental Results

We compared the proposed PU-Learning framework with two popular word embedding models – SGNS (Mikolov et al., 2013b) and Glove (Pennington et al., 2014) on English and three other languages. The experimental results are reported in Table 4. The results show that the proposed PU-Learning framework outperforms the two baseline approaches significantly in most datasets. This re-

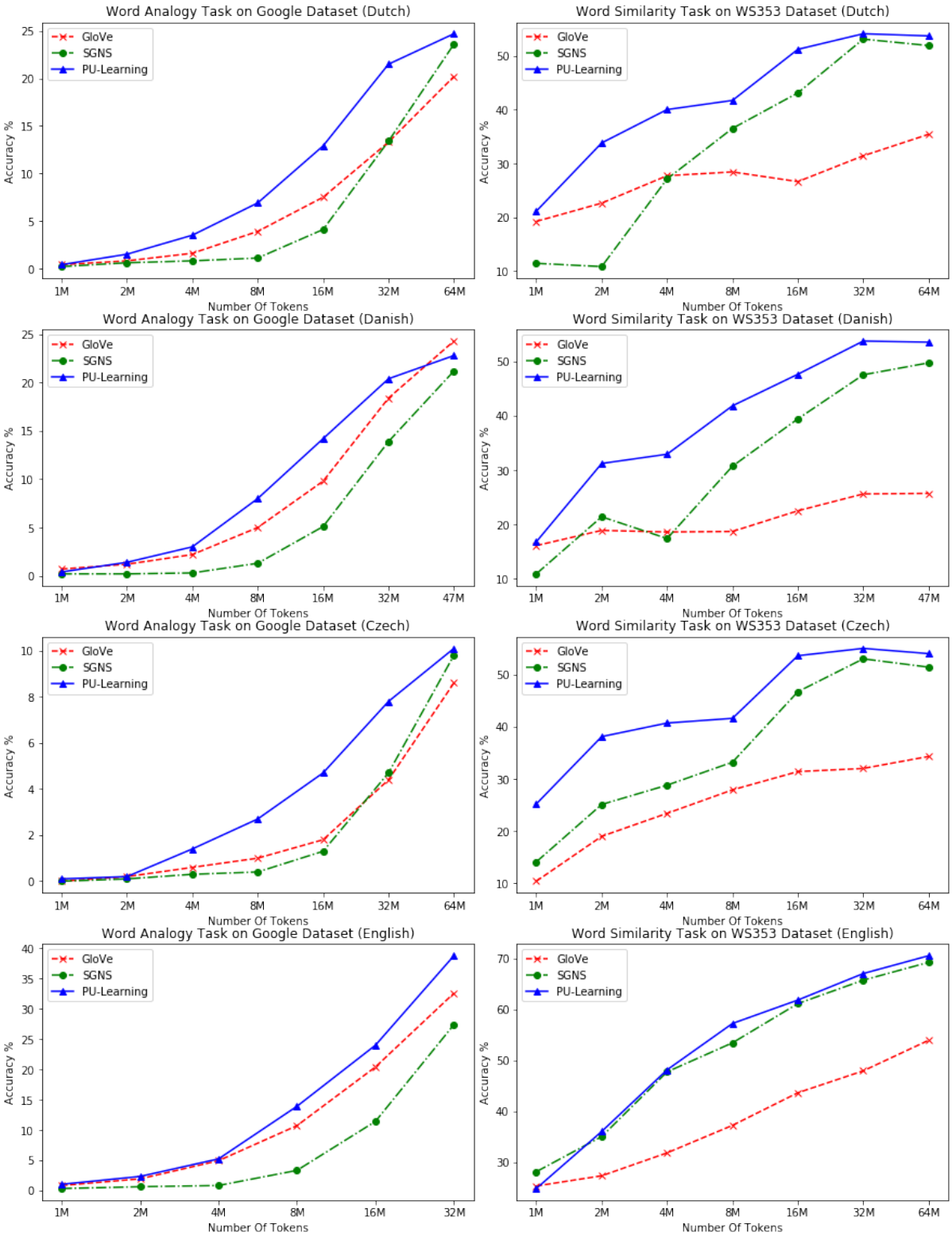


Figure 1: Performance change as the corpus size growing (a) on the Google word analogy task (on the left-hand side) and (b) on the WS353 word similarity task (on the right-hand side). We demonstrate the performance on four languages, Dutch, Danish, Czech and English datasets. Results show that PU-Learning model consistently outperforms SGNS and GloVe when the size of corpus is small.

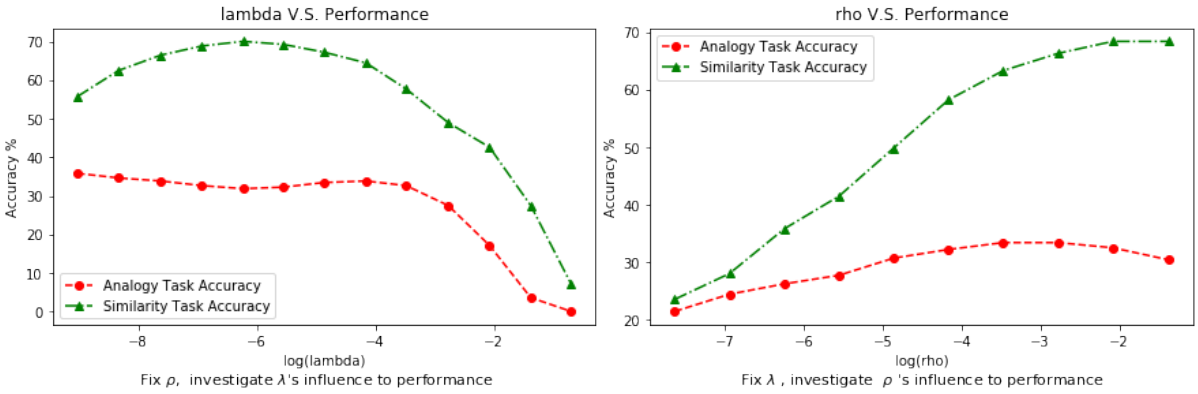


Figure 2: Impact of ρ and λ in the PU-Learning framework.

sults confirm that the unobserved word pairs carry important information and the PU-Learning model leverages such information and achieves better performance. To better understand the model, we conduct detailed analysis as follows.

Performance v.s. Corpus size We investigate the performance of our algorithm with respect to different corpus size, and plot the results in Figure 1. The results in analogy task are obtained by 3CosMul method (Levy and Goldberg, 2014a). As the corpus size grows, the performance of all models improves, and the PU-learning model consistently outperforms other methods in all the tasks. However, with the size of the corpus increases, the difference becomes smaller. This is reasonable as when the corpus size increases the number of non-zero terms becomes smaller and the PU-learning approach is resemblance to Glove.

Impacts of ρ and λ We investigate how sensitive the model is to the hyper-parameters, ρ and λ. Figure 2 shows the performance along with various values of λ and ρ when training on the text8 corpus, respectively. Note that the x-axis is in log scale. When ρ is fixed, a big λ degrades the performance of the model significantly. This is because when λ is too big the model suffers from underfitting. The model is less sensitive when λ is small and in general, $\lambda = 2^{-11}$ achieves consistently good performance.

When λ is fixed, we observe that large ρ (e.g., $\rho \approx 2^{-4}$) leads to better performance. As ρ represents the weight assigned to the unobserved term, this result confirms that the model benefits from using the zero terms in the co-occurrences matrix.

6 Conclusion

In this paper, we presented a PU-Learning framework for learning word embeddings of low-resource languages. We evaluated the proposed approach on English and other three languages and showed that the proposed approach outperforms other baselines by effectively leveraging the information from unobserved word pairs.

In the future, we would like to conduct experiments on other languages where available text corpora are relatively hard to obtain. We are also interested in applying the proposed approach to domains, such as legal documents and clinical notes, where the amount of accessible data is small. Besides, we plan to study how to leverage other information to facilitate the training of word embeddings under the low-resource setting.

Acknowledge

This work was supported in part by National Science Foundation Grant IIS-1760523, IIS-1719097 and an NVIDIA Hardware Grant.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and WordNet-based approaches. In *The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. pages 19–27.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual Question Answering. In *Proceedings of the IEEE International Conference on Computer Vision*. pages 2425–2433.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics, pages 136–145.

John A Bullinaria and Joseph P Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior research methods* 39(3):510–526.

Kai-Wei Chang, Wen tau Yih, Bishan Yang, and Chris Meek. 2014. Typed Tensor Decomposition of Knowledge Bases for Relation Extraction. In *EMNLP*.

Kai-Wei Chang, Wen-tau Yih, and Christopher Meek. 2013. Multi-relational latent semantic analysis. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. pages 1602–1612.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics* 16(1):22–29.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science* 41(6):391.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*. ACM, pages 406–414.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Revisiting embedding features for simple semi-supervised learning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 110–120.

Cho-Jui Hsieh, Nagarajan Natarajan, and Inderjit Dhillon. 2015. Pu learning for matrix completion. In *International Conference on Machine Learning*. pages 2445–2453.

Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*. pages 263–272.

Ozan Irsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *Advances in neural information processing systems*. pages 2096–2104.

Shihao Ji, Hyokun Yun, Pinar Yanardag, Shin Matsushima, and SVN Vishwanathan. 2015. Wordrank: Learning word embeddings via robust ranking. *arXiv preprint arXiv:1506.02761* .

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 3128–3137.

Omer Levy and Yoav Goldberg. 2014a. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the eighteenth conference on computational natural language learning*. pages 171–180.

Omer Levy and Yoav Goldberg. 2014b. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*. pages 2177–2185.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics* 3:211–225.

Xiao-Li Li and Bing Liu. 2005. Learning from positive and unlabeled examples with different data distributions. In *European Conference on Machine Learning*. Springer, pages 218–229.

Kevin Lund and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers* 28(2):203–208.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Eric Nalisnick, Bhaskar Mitra, Nick Craswell, and Rich Caruana. 2016. Improving document ranking with dual word embeddings. In *Proceedings of the 25th International Conference Companion on World Wide Web*. International World Wide Web Conferences Steering Committee, pages 83–84.

Rong Pan and Martin Scholz. 2009. Mind the gaps: Weighting the unknown in large-scale one-class collaborative filtering. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. pages 667–676.

Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE, pages 502–511.

Ulrich Paquet and Noam Koenigstein. 2013. One-class collaborative filtering with random graphs. In *Proceedings of the 22nd international conference on World Wide Web*. ACM, pages 999–1008.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pages 1532–1543.

Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. 2010. LETOR: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval* 13(4):346–374.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th international conference on World wide web*. ACM, pages 337–346.

Karl Stratos, Michael Collins, and Daniel Hsu. 2015. Model-based word embeddings from decompositions of count matrices. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. volume 1, pages 1282–1291.

Wen-tau Yih, Geoffrey Zweig, and John C Platt. 2012. Polarity inducing latent semantic analysis. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, pages 1212–1222.

Hsiang-Fu Yu, Mikhail Bilenko, and Chih-Jen Lin. 2017a. Selection of negative samples for one-class matrix factorization. In *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, pages 363–371.

Hsiang-Fu Yu, Hsin-Yuan Huang, Inderjit S Dhillon, and Chih-Jen Lin. 2017b. A unified algorithm for one-class structured matrix factorization with side information. In *AAAI*. pages 2845–2851.

Torsten Zesch, Christof Müller, and Iryna Gurevych. 2008. Using wiktionary for computing semantic relatedness. In *AAAI*. volume 8, pages 861–866.

Exploring the Role of Prior Beliefs for Argument Persuasion

Esin Durmus
Cornell University
ed459@cornell.edu

Claire Cardie
Cornell University
cardie@cs.cornell.edu

Abstract

Public debate forums provide a common platform for exchanging opinions on a topic of interest. While recent studies in natural language processing (NLP) have provided empirical evidence that the language of the debaters and their patterns of interaction play a key role in changing the mind of a reader, research in psychology has shown that prior beliefs can affect our interpretation of an argument and could therefore constitute a competing alternative explanation for resistance to changing one's stance. To study the actual effect of language use vs. prior beliefs on persuasion, we provide a new dataset and propose a controlled setting that takes into consideration two reader-level factors: political and religious ideology. We find that prior beliefs affected by these reader-level factors play a more important role than language use effects and argue that it is important to account for them in NLP studies of persuasion.

1 Introduction

Public debate forums provide to participants a common platform for expressing their point of view on a topic; they also present to participants the different sides of an argument. The latter can be particularly important: awareness of divergent points of view allows one, in theory, to make a fair and informed decision about an issue; and exposure to new points of view can furthermore possibly persuade a reader to change his overall stance on a topic.

Research in natural language processing (NLP) has begun to study persuasive writing and the role of language in persuasion. Tan et al. (2016) and Zhang et al. (2016), for example, have shown that the language of opinion holders or debaters and their patterns of interaction play a key role in changing the mind of a reader. At the same time, research in psychology has shown that prior beliefs can affect our interpretation of an argument even when the argument consists of numbers and empirical studies that would seemingly belie misinterpretation (Lord et al., 1979; Vallone et al., 1985; Chambliss and Garner, 1996).

We hypothesize that studying the actual effect of language on persuasion will require a more controlled experimental setting — one that takes into account any potentially confounding user-level (i.e., reader-level) factors[1] that could cause a person to change, or keep a person from changing, his opinion. In this paper we study one such type of factor: the prior beliefs of the reader as impacted by their political or religious ideology. We adopt this focus since it has been shown that ideologies play an important role for an individual when they form beliefs about controversial topics, and potentially affect how open the individual is to being persuaded (Stout and Buddenbaum, 1996; Goren, 2005; Croucher and Harris, 2012).

We first present a dataset of online debates that enables us to construct the setting described above in which we can study the effect of language on persuasion while taking into account selected user-level factors. In addition to the text of the debates, the dataset contains a multitude of background information on the users of the debate platform. To the best of our knowledge, it is the first publicly available dataset of debates that simultaneously provides such comprehensive information about the debates, the debaters and those voting on the debates.

With the dataset in hand, we then propose the novel task of studying persuasion (1) at the level of individual users, and (2) in a setting that can control for selected user-level factors, in our case, the prior beliefs associated with the political or

[1] Variables that affect both the dependent and independent variables causing misleading associations.

Proceedings of NAACL-HLT 2018, pages 1035–1045
New Orleans, Louisiana, June 1 - 6, 2018.

religious ideology of the debaters and voters. In particular, previous studies focus on predicting the winner of a debate based on the cumulative change in pre-debate vs. post-debate votes for the opposing sides (Zhang et al., 2016; Potash and Rumshisky, 2017). In contrast, we aim to predict which debater an individual user (i.e., reader of the debate) perceives as more successful, given their stated political and religious ideology.

Finally, we identify which features appear to be most important for persuasion, considering the selected user-level factors as well as the more traditional linguistic features associated with the language of the debate itself. We hypothesize that the effect of political and religious ideology will be stronger when the debate topic is *Politics* and *Religion*, respectively. To test this hypothesis, we experiment with debates on only *Politics* or only *Religion* vs. debates from all topics including *Music*, *Health*, *Arts*, etc.

Our main finding is that prior beliefs associated with the selected user-level factors play a larger role than linguistic features when predicting the successful debater in a debate. In addition, the effect of these factors varies according to the topic of the debate topic. The best performance, however, is achieved when we rely on features extracted from user-level factors in conjunction with linguistic features derived from the debate text. Finally, we find that the set of linguistic features that emerges as the most predictive changes when we control for user-level factors (political and religious ideology) vs. when we do not, showing the importance of accounting for these factors when studying the effect of language on persuasion.

In the remainder of the paper, we describe the debate dataset (Section 2) and the prediction task (Section 3) followed by the experimental results and analysis (Section 4), related work (Section 5) and conclusions (Section 6).

2 Dataset

For this study, we collected 67,315 debates from debate.org[2] from 23 different topic categories including *Politics*, *Religion*, *Health*, *Science* and *Music*.[3] In addition to text of the debates, we collected 198,759 votes from the readers of these debates. Votes evaluate different dimensions of the debate.

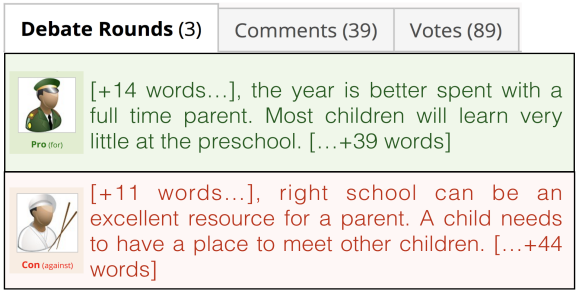


Figure 1: ROUND 1 for the debate claim "PRESCHOOL IS A WASTE OF TIME".

To study the effect of user characteristics, we collected user information for 36,294 different users. Aspects of the dataset most relevant to our task are explained in the following section in more detail.

2.1 Debates

Debate rounds. Each debate consists of a sequence of ROUNDS in which two debaters from opposing sides (one is supportive of the claim (i.e., PRO) and the other is against the claim (i.e., CON)) provide their arguments. Each debater has a single chance in a ROUND to make his points. Figure 1 shows an example ROUND 1 for the debate claim "PRESCHOOL IS A WASTE OF TIME". The number of ROUNDS in debates ranges from 1 to 5 and the majority of debates (61,474 out of 67,315) contain 3 or more ROUNDS.

Votes. All users in the *debate.org* community can vote on debates. As shown in Figure 2, voters share their stances on the debate topic before and after the debate and evaluate the debaters' conduct, their spelling and grammar, the convincingness of their arguments and the reliability of the sources they refer to. For each such dimension, voters have the option to choose one of the debaters as better or indicate a tie. This fine-grained voting system gives a glimpse into the reasoning behind the voters' decisions.

2.1.1 Determining the successful debater

There are two alternate criteria for determining the successful debater in a debate. Our experiments consider both.

Criterion 1: Argument quality. As shown in Figure 2, debaters get points for each dimension of the debate. The most important dimension — in

[2] www.debate.org

[3] The dataset will be made publicly available at http://www.cs.cornell.edu/ esindurmus/.

	Debater 1	Debater 2	Tied	
Agreed with before the debate:	✔	-	-	*0 points*
Agreed with after the debate:	-	✔	-	*0 points*
Who had better conduct:	-	✔	-	*1 point*
Had better spelling and grammar:	-	✔	-	*1 point*
Made more convincing arguments:	-	✔	-	*3 points*
Used the most reliable sources:	-	✔	-	*2 points*
Total points awarded:	**0**	**7**		

Figure 2: An example post-debate vote. Convincingness of arguments contributes to the total points the most.

that it contributes most to the point total — is making convincing arguments. *debate.org* uses Criterion 1 to determine the winner of a debate.

Criterion 2: Convinced voters. Since voters share their stances before and after the debate, the debater who convinces more voters to change their stance is declared as the winner.

2.2 User information

On *debate.org*, each user has the option to share demographic and private state information such as their age, gender, ethnicity, political ideology, religious ideology, income level, education level, the president and the political party they support. Beyond that, we have access to information about their activities on the website such as their overall success rate of winning debates, the debates they participated in as a debater or voter, and their votes. An example of a user profile is shown in Figure 3.

Opinions on the *big issues*. *debate.org* maintains a list of the most controversial debate topics as determined by the editors of the website. These are referred to as *big issues*.[4] Each user shares his stance on each *big issue* on his profile (see Figure 3): either PRO (in favor), CON (against), N/O (no opinion), N/S (not saying) or UND (undecided).

3 Prediction task: which debater will be declared as more successful by an individual voter?

In this section, we first analyze which dimensions of argument quality are the most important for determining the successful debater. Then, we analyze whether there is any connection between selected user-level factors and users' opinions on the

[4] http://www.debate.org/big-issues/

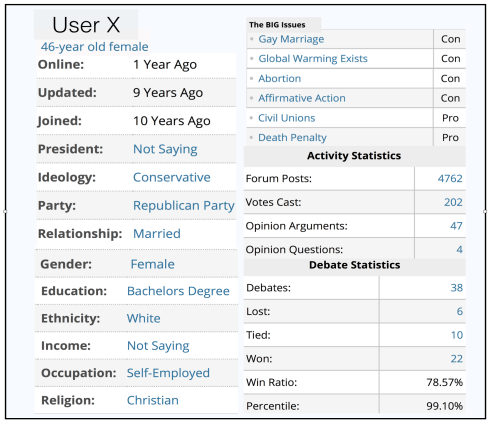


Figure 3: An example of a (partial) user profile. Top right: Some of the *big issues* on which the user shares his opinion are included. The user is against (CON) abortion and gay marriage and in favor of (PRO) the death penalty.

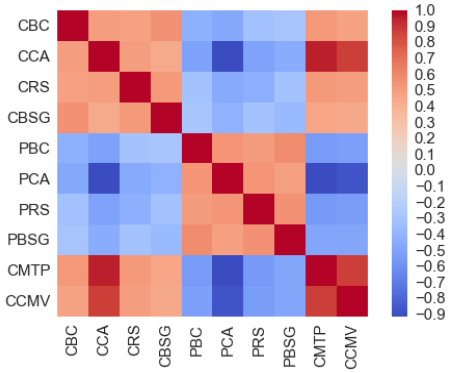


Figure 4: The correlations among argument quality dimensions.

big issues to see if we can infer their opinions from these factors. Finally, using our findings from these analyses, we perform the task of predicting which debater will be perceived as more successful by an individual voter.

3.1 Relationships between argument quality dimensions

Figure 4 shows the correlation between pairs of voting dimensions (in the first 8 rows and columns) and the correlation of each dimension with (1) getting more points (row or column 9) and (2) convincing more people as a debater (final row or column). Abbreviations stand for (on the CON side): has better conduct (CBC), makes more convincing arguments (CCA), uses more reliable sources (CRS), has better spelling and grammar (CBSG), gets more total points (CMTP) and convinces more voters (CCMV). For the PRO side we

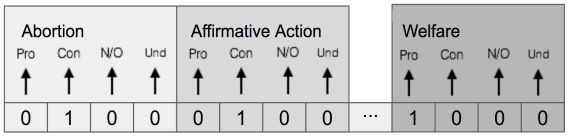


Figure 5: The representation of the BIGISSUES vector derived by this user's decisions on *big issues*. Here, the user is CON for ABORTION and AFFIRMATIVE ACTION issues and PRO for the WELFARE issue.

use PBC, PCA, and so on.

From Figure 4, we can see that making more convincing arguments (CCA) correlates the most with total points (CMTP) and convincing more voters (CCMV). This analysis motivates us to identify the linguistic features that are indicators of more convincing arguments.

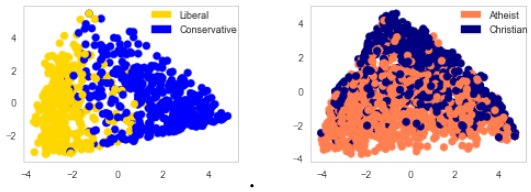
3.2 The relationship between a user's opinions on the *big issues* and their prior beliefs

We disentangle different aspects of a person's prior beliefs to understand how well each correlates with their opinions on the *big issues*. As noted earlier, we focus here only on prior beliefs in the form of self-identified political and religious ideology.

Representing the *big issues*. To represent the opinions of a user on a *big issue*, we use a four-dimensional one-hot encoding where the indices of the vector correspond to PRO, CON, N/O (no opinion), and UND (undecided), consecutively (1 if the user chooses that value for the issue, 0 otherwise). Note that we do not have a representation for N/S since we eliminate users having N/S for at least one *big issue* for this study. We then concatenate the vector for each *big issue* to get a representation for a user's stance on all the *big issues* as shown in Figure 5. We denote this vector by BIGISSUES.

We test the correlation between the individual's opinions on *big issues* and the selected user-level factors in this study using two different approaches: clustering and classification.

Clustering the users' decisions on *big issues*. We apply PCA on the BIGISSUES vectors of users who identified themselves as CONSERVATIVE vs. LIBERAL (740 users). We do the same for the users who identified themselves as ATHEIST vs. CHRISTIAN (1501 users). In Figure 6, we see that there are distinctive clusters of CONSERVATIVE vs. LIBERAL users in the two-dimensional representation



(a) LIBERAL VS. CONSERVATIVE (b) ATHEIST VS. CHRISTIAN.

Figure 6: PCA representation of decisions on *big issues* color-coded with political and religious ideology. We see more distinctive clusters for CONSERVATIVE vs. LIBERAL users suggesting that people's opinions are more correlated with their political ideology.

Prior belief type	Majority	BIGISSUES
Political ideology	57.70%	92.43%
Religious Ideology	52.70%	82.81%

Table 1: Accuracy using majority baseline vs. BIGISSUES vectors as features.

while for ATHEIST VS. CHRISTIAN, the separation is not as distinct. This suggests that people's opinions on the *big issues* identified by *debate.org* correlate more with their political ideology than their religious ideology.

Classification approach. We also treat this as a classification task[5] using the BIGISSUES vectors for each user as features and the user's religious and political ideology as the labels to be predicted. So the classification task is: Given the user's BIGISSUES vector, predict his political and religious ideology. Table 1 shows the accuracy for each case. We see that using the BIGISSUES vectors as features performs significantly better[6] than majority baseline[7].

This analysis shows that there is a clear relationship between people's opinions on the *big issues* and the selected user-level factors. It raises the question of whether it is even possible to persuade someone with prior beliefs relevant to a debate claim to change their stance on the issue. It may be the case that people prefer to agree with the individuals having the same (or similar) beliefs regardless of the quality of the arguments and the

[5]For all the classification tasks described in this paper, we experiment with logistic regression, optimizing the regularizer ($\ell 1$ or $\ell 2$) and the regularization parameter C (between 10^{-5} and 10^{5}).

[6]We performed the McNemar significance test.

[7]The majority class baseline predicts CONSERVATIVE for political and CHRISTIAN for religious ideology for each example, respectively.

particular language used. Therefore, it is important to understand the relative effect of prior beliefs vs. argument strength on persuasion.

3.3 Task descriptions

Some of the previous work in NLP on persuasion focuses on predicting the winner of a debate as determined by the change in the number of people supporting each stance before and after the debate (Zhang et al., 2016; Potash and Rumshisky, 2017). However, we believe that studies of the effect of language on persuasion should take into account other, extra-linguistic, factors that can affect opinion change: in particular, we propose an experimental framework for studying the effect of language on persuasion that aims to control for the prior beliefs of the reader as denoted through their self-identified political and religious ideologies. As a result, we study a more fine-grained prediction task: for an individual voter, predict which side/debater/argument the voter will declare as the winner.

Task 1 : Controlling for religious ideology. In the first task, we control for religious ideology by selecting debates for which each of the two debaters is from a different religious ideology (e.g., debater 1 is ATHEIST, debater 2 is CHRISTIAN). In addition, we consider only voters that (a) self-identify with one of these religious ideologies (e.g., the voter is either ATHEIST or CHRISTIAN) and (b) changed their stance on the debate claim post-debate vs. pre-debate. For each such voter, we want to predict which of the PRO-side debater or the CON-side debater did the convincing. Thus, in this task, we use *Criterion 2* to determine the winner of the debate from the point of view of the voter. Our hypothesis is that the voter will be convinced by the debater that espouses the religious ideology of the voter.

In this setting, we can study the factors that are important for a particular voter to be convinced by a debater. This setting also provides an opportunity to understand how the voters who change their minds perceive arguments from a debater who is expressing the same vs. the opposing prior belief.

To study the effect of the debate topic, we perform this study for two cases — debates belonging to the *Religion* category and then all the categories. The *Religion* category contains debates like "IS THE BIBLE AGAINST WOMEN'S RIGHTS?" and "RELIGIOUS THEORIES SHOULD NOT BE TAUGHT IN SCHOOL". We want to see how strongly a user's religious ideology affects the persuasive effect of language in such a topic as compared to the all topics. We expect to see stronger effects of prior beliefs for debates on *Religion*.

Task 2: Controlling for political ideology. Similar to the setting described above, Task 2 controls for political ideology. In particular, we only use debates where the two debaters are from different political ideologies (CONSERVATIVE vs. LIBERAL). In contrast to Task 1, we consider all voters that self-identify with one of the two debater ideologies (regardless of whether the voter's stance changed post-debate vs. pre-debate). This time, we predict whether the voter gives more total points to the PRO side or the CON side argument. Thus, Task 2 uses *Criterion 1* to determine the winner of the debate from the point of view of the voter. Our hypothesis is that the voter will assign more points to the debater that has the same political ideology as the voter.

For this task too, we perform the study for two cases — debates from the *Politics* category only and debates from all categories. And we expect to see stronger effects of prior beliefs for debates on *Politics*.

3.4 Features

The features we use in our model are shown in Table 2. They can be divided into two groups — features that describe the prior beliefs of the users and linguistic features of the arguments themselves.

User features

We use the cosine similarities between the voter and each of the debaters' *big issue* vectors. These features give a good approximation of the overall similarity of two user's opinions. Second, we use indicator features to encode whether the religious and political beliefs of the voter match those of each of the debaters.

Linguistic features

We extract linguistic features separately for both the PRO and CON side of the debate (combining all the utterances of PRO across different turns and doing the same for CON). Table 2 contains a list of these features. It includes features that carry information about the style of the language (e.g., usage of modal verbs, length, punctuation), represent different semantic aspects of the argu-

User-based features	Description
Opinion similarity.	For $userA$ and $userB$, the cosine similarity of BIGISSUES$_{userA}$ and BIGISSUES$_{userB}$.
Matching features.	For $userA$ and $userB$, 1 if $userA_f$==$userB_f$, 0 otherwise where $f \in$ {political ideology, religious ideology}. We denote these features as *matching political ideology* and *matching religious ideology*.
Linguistic features	**Description**
Length.	Number of tokens.
Tf-idf.	Unigram, bigram and trigram features.
Referring to the opponent.	Whether the debater refers to their opponent using words or phrases like "opponent, my opponent".
Politeness cues.	Whether the text includes any signs of politeness such as "thank" and "welcome".
Showing evidence.	Whether the text has any signs of citing any other sources (e.g., phrases like "according to"), or quotation.
Sentiment.	Average sentiment polarity.
Subjectivity (Wilson et al., 2005).	Number of words with negative strong, negative weak, positive strong, and positive weak subjectivity.
Swear words.	# of swear words.
Connotation score (Feng and Hirst, 2011).	Average # of words with positive, negative and neutral connotation.
Personal pronouns.	Usage of first, second, and third person pronouns.
Modal verbs.	Usage of modal verbs.
Argument lexicon features. (Somasundaran et al., 2007).	# of phrases corresponding to different argumentation styles.
Spelling.	# of spelling errors.
Links.	# of links.
Numbers.	# of numbers.
Exclamation marks.	# of exclamation marks.
Questions.	# of questions.

Table 2: Feature descriptions

ment (e.g., showing evidence, connotation (Feng and Hirst, 2011), subjectivity (Wilson et al., 2005), sentiment, swear word features) as well as features that convey different argumentation styles (argument lexicon features (Somasundaran and Wiebe, 2010). Argument lexicon features include the counts for the phrases that match with the regular expressions of argumentation styles such as assessment, authority, conditioning, contrasting, emphasizing, generalizing, empathy, inconsistency, necessity, possibility, priority, rhetorical questions, desire, and difficulty. We then concatenate these features to get a single feature representation for the entire debate.

4 Results and Analysis

For each of the tasks, prediction accuracy is evaluated using 5-fold cross validation. We pick the model parameters for each split with 3-fold cross validation on the training set. We do ablation for each of user-based and linguistic features. We report the results for the feature sets that perform better than the baseline.

We perform analysis by training logistic regression models using only user-based features, only linguistic features and finally combining user-based and linguistic features for both the tasks.

	Accuracy
Baseline	
Majority	56.10%
User-based Features	
Matching religious ideology	65.37 %
Linguistic features	
Personal pronouns	57.00 %
Connotation	61.26 %
All two features above	65.37 %
User-based+linguistic features	
USER*+ Personal pronouns	65.37%
USER*+ Connotation	66.42%
USER*+ LANGUAGE*	64.37%

Table 3: Results for Task 1 for debates in category *Religion*. USER* represents the best performing combination of user-based features. LANGUAGE* represents the best performing combination of linguistic features. Since using linguistic features only would give the same prediction for all voters in a debate, the maximum accuracy that can be achieved using language features only is 92.86%.

	Accuracy
Baseline	
Majority	57.31%
User-based Features	
Matching religious ideology	62.79 %
Matching religious ideology+ Opinion similarity	62.97%
Linguistic features	
Length [8]	61.01 %
User-based+linguistic features	
USER* + Length	64.56 %
USER*+ Length + Exclamation marks	65.74%

Table 4: Results for Task 1 for debates in all categories. The maximum accuracy that can be achieved using language features only is 95.77%.

Task 1 for debates in category *Religion*. As shown in Table 3, the majority baseline (predicting the winner side of the majority of training examples out of PRO or CON) gets 56.10% accuracy. User features alone perform significantly better than the majority baseline. The most important user-based feature is *matching religious ideology*. This means it is very likely that people change their views in favor of a debater with the same religious ideology. In a linguistic-only features analysis, combination of the *personal pronouns* and *connotation* features emerge as most important and also perform significantly better than the majority baseline at 65.37% accuracy. When we use both user-based and linguistic features to predict, the accuracy improves to 66.42% with *connotation* features. An interesting observation is that including the user-based features along with the linguistic features changes the set of important linguistic features for persuasion removing the *personal pronouns* from the important linguistic features set. This shows the importance of studying potentially confounding user-level factors.

Task 1 for debates in all categories. As shown in Table 4, for the experiments with user-based features only, *matching religious ideology* and *opinion similarity* features are the most important. For this task, *length* is the most predictive linguistic feature and can achieve significant improvement over the baseline (61.01%). When we combine the language features with user-based features, we see that with *exclamation mark* the accuracy improves to (65.74%).

Task 2 for debates in category *Politics*. As shown in Table 5, using user-based features only, the *matching political ideology* feature performs the best (80.40%). Linguistic features (refer to Table 5 for the full list) alone, however, can still obtain significantly better accuracy than the baseline (59.60%). The most important linguistic features include *approval*, *politeness*, *modal verbs*, *punctuation* and *argument lexicon features* such as *rhetorical questions* and *emphasizing*. When combining this linguistic feature set with the *matching political ideology* feature, we see that with the accuracy improves to (81.81%). *Length* feature does not give any improvement when it is combined with the user features.

Task 2 for debates in all categories. As shown in Table 6, when we include all categories, we see that the best performing user-based feature is the *opinion similarity* feature (73.96%). When using language features only, *length* feature (56.88%) is the most important. For this setting, the best accuracy is achieved when we combine user features with *length* and *Tf-idf* features. We see that the set of language features that improve the performance of user-based features do not include some of that perform significantly better than the baseline when used alone (*modal verbs* and *politeness* features).

	Accuracy
Baseline	
Majority	50.91%
User-based Features	
Opinion similarity	80.00 %
Matching political ideology	80.40 %
Linguistic features	
Length	57.37 %
linguistic feature set	59.60 %
User-based+linguistic features	
USER*+ *linguistic feature set*	81.81%

Table 5: Results for Task 2 for debates in category *Politics*. The maximum accuracy that can be achieved using linguistic features only is 75.35%. The *linguistic feature set* includes *rhetorical questions, emphasizing, approval, exclamation mark, questions, politeness, referring to opponent, showing evidence, modals, links,* and *numbers* as features.

	Accuracy
Baseline	
Majority	51.75%
User-based Features	
Opinion similarity	73.96%
Linguistic features	
Length	56.88%
Politeness	55.00%
Modal verbs	52.32%
Tf-idf features	52.89 %
User-based+linguistic features	
USER*+ Length	74.53%
USER*+ Tf-idf	74.13%
USER*+ Length + Tf-idf	75.20%

Table 6: Results for Task 2 for debates in all categories. The maximum accuracy that can be achieved using linguistic features only is 74.53%.

5 Related Work

Below we provide an overview of related work from the multiple disciplines that study persuasion.

Argumentation mining. Although most recent work on argumentation has focused on identifying the structure of arguments and extracting argument components (Persing and Ng, 2015; Palau and Moens, 2009; Biran and Rambow, 2011; Mochales and Moens, 2011; Feng and Hirst, 2011; Stab and Gurevych, 2014; Lippi and Torroni, 2015; Park and Cardie, 2014; Nguyen and Litman, 2015; Peldszus and Stede, 2015; Niculae et al., 2017; Rosenthal and McKeown, 2015), more relevant is research on identifying the characteristics of persuasive text, e.g., what distinguishes persuasive from non-persuasive text (Tan et al., 2016; Zhang et al., 2016; **?**; Habernal and Gurevych, 2016a,b; Fang et al., 2016; Hidey et al., 2017). Similar to these, our work aims to understand the characteristics of persuasive text but also considers the effect of people's prior beliefs.

Persuasion. There has been a tremendous amount of research effort in the social sciences (including computational social science) to understand the characteristics of persuasive text (Kelman, 1961; Burgoon et al., 1975; Chaiken, 1987; Tykocinskl et al., 1994; Chambliss and Garner, 1996; Dillard and Pfau, 2002; Cialdini, 2007; Durik et al., 2008; Tan et al., 2014; Marquart and Naderer, 2016). Most relevant among these is the research of Tan et al. (2016), Habernal and Gurevych (2016a) and Hidey et al. (2017). Tan et al. (2016) focused on the effect of user interaction dynamics and language features looking at the ChangeMyView[9] (an internet forum) community on Reddit and found that user interaction patterns as well as linguistic features are connected to the success of persuasion. In contrast, Habernal and Gurevych (2016a) created a crowd-sourced corpus consisting of argument pairs and, given a pair of arguments, asked annotators which is more convincing. This allowed them to experiment with different features and machine learning techniques for persuasion prediction. Taking motivation from Aristotle's definition for modes of persuasion, Hidey et al. (2017) annotated claims and premises extracted from the ChangeMyView community with their semantic types to study if certain semantic types or different combinations of semantic types appear in persuasive but not in non-persuasive essays. In contrast to the above, our work focuses on persuasion in debates than monologues and forum datasets and accounts for the user-based features.

Persuasion in debates. Debates are another resource for studying the different aspects of persuasive arguments. Different from monologues where the audience is exposed to only one side of the opinions about an issue, debates allow the audience to see both sides of a particular issue via a

[9] https://www.reddit.com/r/changemyview/

controlled discussion. There has been some work on argumentation and persuasion on online debates. Sridhar et al. (2015), Somasundaran and Wiebe (2010) and Hasan and Ng (2014), for example, studied detecting and modeling stance on online debates. Zhang et al. (2016) found that the side that can adapt to their opponents' discussion points over the course of the debate is more likely to be the winner. None of these studies investigated the role of prior beliefs in stance detection or persuasion.

User effects in persuasion. Persuasion is not independent from the characteristics of the people to be persuaded. Research in psychology has shown that people have biases in the ways they interpret the arguments they are exposed to because of their prior beliefs (Lord et al., 1979; Vallone et al., 1985; Chambliss and Garner, 1996). Understanding the effect of persuasion strategies on people, the biases people have and the effect of prior beliefs of people on their opinion change has been an active area of research interest (Correll et al., 2004; Hullett, 2005; Petty et al., 1981). Eagly and Chaiken (1975), for instance, found that the attractiveness of the communicator plays an important role in persuasion. Work in this area could be relevant for the future work on modeling shared characteristics between the user and the debaters. To the best of our knowledge, Lukin et al. (2017) is the most relevant work to ours since they consider features of the audience on persuasion. In particular, they studied the effect of an individual's personality features (open, agreeable, extrovert, neurotic, etc.) on the type of argument (factual vs. emotional) they find more persuasive. Our work differs from this work since we study debates and in our setting the voters can see the debaters' profiles as well as all the interactions between the two sides of the debate rather than only being exposed to a monologue. Finally, we look at different types of user profile information such as a user's religious and ideological beliefs and their opinions on various topics.

6 Conclusion

In this work we provide a new dataset of debates and a more controlled setting to study the effects of prior belief on persuasion. The dataset we provide and the framework we propose open several avenues for future research. One could explore the effect different aspects of people's background (e.g., gender, education level, ethnicity) on persuasion. Furthermore, it would be interesting to study how people's prior beliefs affect their other activities on the website and the language they use while interacting with people with the same and different prior beliefs. Finally, one could also try to understand in what aspects and how the language people with different prior beliefs/backgrounds use is different. These different directions would help people better understand characteristics of persuasive arguments and the effects of prior beliefs in language.

7 Acknowledgements

This work was supported in part by NSF grant SES-1741441 and DARPA DEFT Grant FA8750-13-2-0015. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF, DARPA or the U.S. Government. We thank Yoav Artzi, Faisal Ladhak, Amr Sharaf, Tianze Shi, Ashudeep Singh and the anonymous reviewers for their helpful feedback. We also thank the Cornell NLP group for their insightful comments.

References

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

Michael Burgoon, Stephen B Jones, and Diane Stewart. 1975. Toward a message-centered theory of persuasion: Three empirical investigations of language intensity1. *Human Communication Research* 1(3):240–256.

Shelly Chaiken. 1987. The heuristic model of persuasion. In *Social influence: the ontario symposium*. Hillsdale, NJ: Lawrence Erlbaum, volume 5, pages 3–39.

Marilyn J. Chambliss and Ruth Garner. 1996. Do adults change their minds after reading persuasive text? *Written Communication* 13(3):291–313. https://doi.org/10.1177/0741088396013003001.

Robert B. Cialdini. 2007. Influence: The psychology of persuasion.

Joshua Correll, Steven J Spencer, and Mark P Zanna. 2004. An affirmed self and an open mind: Self-affirmation and sensitivity to argument

strength. *Journal of Experimental Social Psychology* 40(3):350–356.

S.M. Croucher and T.M. Harris. 2012. *Religion and Communication: An Anthology of Extensions in Theory, Research, and Method*. Peter Lang. https://books.google.com/books?id=CTfpugAACAAJ.

James Price Dillard and Michael Pfau. 2002. *The persuasion handbook: Developments in theory and practice*. Sage Publications.

Amanda M Durik, M Anne Britt, Rebecca Reynolds, and Jennifer Storey. 2008. The effects of hedges in persuasive arguments: A nuanced analysis of language. *Journal of Language and Social Psychology* 27(3):217–234.

Alice H Eagly and Shelly Chaiken. 1975. An attribution analysis of the effect of communicator characteristics on opinion change: The case of communicator attractiveness. *Journal of personality and social psychology* 32(1):136.

Hao Fang, Hao Cheng, and Mari Ostendorf. 2016. Learning latent local conversation modes for predicting community endorsement in online discussions. *arXiv preprint arXiv:1608.04808* .

Vanessa Wei Feng and Graeme Hirst. 2011. Classifying arguments by scheme. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pages 987–996.

Paul Goren. 2005. Party identification and core political values. *American Journal of Political Science* 49(4):881–896. https://doi.org/10.1111/j.1540-5907.2005.00161.x.

Ivan Habernal and Iryna Gurevych. 2016a. What makes a convincing argument? empirical analysis and detecting attributes of convincingness in web argumentation. In *EMNLP*. pages 1214–1223.

Ivan Habernal and Iryna Gurevych. 2016b. Which argument is more convincing? analyzing and predicting convincingness of web arguments using bidirectional lstm. In *ACL (1)*.

Kazi Saidul Hasan and Vincent Ng. 2014. Why are you taking this stance? identifying and classifying reasons in ideological debates. In *EMNLP*. volume 14, pages 751–762.

Christopher Hidey, Elena Musi, Alyssa Hwang, Smaranda Muresan, and Kathy McKeown. 2017. Analyzing the semantic types of claims and premises in an online persuasive forum. In *Proceedings of the 4th Workshop on Argument Mining*. Association for Computational Linguistics, Copenhagen, Denmark, pages 11–21. http://www.aclweb.org/anthology/W17-5102.

Craig R Hullett. 2005. The impact of mood on persuasion: A meta-analysis. *Communication Research* 32(4):423–442.

Herbert C Kelman. 1961. Processes of opinion change. *Public opinion quarterly* 25(1):57–78.

Marco Lippi and Paolo Torroni. 2015. Context-independent claim detection for argument mining. http://www.aaai.org/ocs/index.php/IJCAI/IJCAI15/paper/view/10942.

Charles G Lord, Lee Ross, and Mark R Lepper. 1979. Biased assimilation and attitude polarization: The effects of prior theories on subsequently considered evidence. *Journal of personality and social psychology* 37(11):2098.

Stephanie M Lukin, Pranav Anand, Marilyn Walker, and Steve Whittaker. 2017. Argument strength is in the eye of the beholder : Audience effects in persuasion. *arXiv preprint arXiv:1708.09085* .

Franziska Marquart and Brigitte Naderer. 2016. *Communication and Persuasion: Central and Peripheral Routes to Attitude Change*, Springer Fachmedien Wiesbaden, Wiesbaden, pages 231–242. https://doi.org/10.1007/978-3-658-09923-7_20.

Raquel Mochales and Marie-Francine Moens. 2011. Argumentation mining. *Artificial Intelligence and Law* 19(1):1–22.

Huy Nguyen and Diane Litman. 2015. Extracting argument and domain words for identifying argument components in texts. In *Proceedings of the 2nd Workshop on Argumentation Mining*. Association for Computational Linguistics, Denver, CO, pages 22–28. http://www.aclweb.org/anthology/W15-0503.

Vlad Niculae, Joonsuk Park, and Claire Cardie. 2017. Argument mining with structured svms and rnns. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 985–995. https://doi.org/10.18653/v1/P17-1091.

Raquel Mochales Palau and Marie-Francine Moens. 2009. Argumentation mining: the detection, classification and structure of arguments in text. In *Proceedings of the 12th international conference on artificial intelligence and law*. ACM, pages 98–107.

Joonsuk Park and Claire Cardie. 2014. Identifying appropriate support for propositions in online user comments. In *Proceedings of the First Workshop on Argumentation Mining*. Association for Computational Linguistics, Baltimore, Maryland, pages 29–38. http://www.aclweb.org/anthology/W/W14/W14-2105.

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

Isaac Persing and Vincent Ng. 2015. Modeling argument strength in student essays. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*. pages 543–552. http://aclweb.org/anthology/P/P15/P15-1053.pdf.

Richard E Petty, John T Cacioppo, and Rachel Goldman. 1981. Personal involvement as a determinant of argument-based persuasion. *Journal of personality and social psychology* 41(5):847.

Peter Potash and Anna Rumshisky. 2017. Towards debate automation: a recurrent model for predicting debate winners. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 2455–2465.

Sara Rosenthal and Kathy McKeown. 2015. I couldn't agree more: The role of conversational structure in agreement and disagreement detection in online discussions. In *SIGDIAL Conference*. pages 168–177.

Swapna Somasundaran, Josef Ruppenhofer, and Janyce Wiebe. 2007. Detecting arguing and sentiment in meetings. In *Proceedings of the SIGdial Workshop on Discourse and Dialogue*. volume 6.

Swapna Somasundaran and Janyce Wiebe. 2010. Recognizing stances in ideological on-line debates. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*. Association for Computational Linguistics, pages 116–124.

Dhanya Sridhar, James Foulds, Bert Huang, Lise Getoor, and Marilyn Walker. 2015. Joint models of disagreement and stance in online debate. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. volume 1, pages 116–125.

Christian Stab and Iryna Gurevych. 2014. Annotating argument components and relations in persuasive essays. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin City University and Association for Computational Linguistics, Dublin, Ireland, pages 1501–1510. http://www.aclweb.org/anthology/C14-1142.

D.A. Stout and J.M. Buddenbaum. 1996. *Religion and mass media: audiences and adaptations*. Sage Publications. https://books.google.com/books?id=V4cKAQAAMAAJ.

Chenhao Tan, Lillian Lee, and Bo Pang. 2014. The effect of wording on message propagation: Topic- and author-controlled natural experiments on twitter. *arXiv preprint arXiv:1405.1438* .

Chenhao Tan, Vlad Niculae, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2016. Winning arguments: Interaction dynamics and persuasion strategies in good-faith online discussions. In *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, pages 613–624.

Orit Tykocinskl, E Tory Higgins, and Shelly Chaiken. 1994. Message framing, self-discrepancies, and yielding to persuasive messages: The motivational significance of psychological situations. *Personality and Social Psychology Bulletin* 20(1):107–115.

Robert P Vallone, Lee Ross, and Mark R Lepper. 1985. The hostile media phenomenon: biased perception and perceptions of media bias in coverage of the beirut massacre. *Journal of personality and social psychology* 49(3):577.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*. Association for Computational Linguistics, pages 347–354.

Justine Zhang, Ravi Kumar, Sujith Ravi, and Cristian Danescu-Niculescu-Mizil. 2016. Conversational flow in oxford-style debates. *arXiv preprint arXiv:1604.03114* .

Inducing a Lexicon of Abusive Words – A Feature-Based Approach

Michael Wiegand*, **Josef Ruppenhofer**†, **Anna Schmidt***, **Clayton Greenberg***
*Spoken Language Systems, Saarland University, D-66123 Saarbrücken, Germany
†Institute for German Language, D-68161 Mannheim, Germany
michael.wiegand@lsv.uni-saarland.de
ruppenhofer@ids-mannheim.de
anna.schmidt@lsv.uni-saarland.de
clayton.greenberg@lsv.uni-saarland.de

Abstract

We address the detection of abusive words. The task is to identify such words among a set of negative polar expressions. We propose novel features employing information from both corpora and lexical resources. These features are calibrated on a small manually annotated base lexicon which we use to produce a large lexicon. We show that the word-level information we learn cannot be equally derived from a large dataset of annotated microposts. We demonstrate the effectiveness of our (domain-independent) lexicon in the cross-domain detection of abusive microposts.

1 Introduction

Abusive or offensive language is commonly defined as hurtful, derogatory or obscene utterances made by one person to another person.[1] Examples are (1)-(3). In the literature, closely related terms include *hate speech* (Waseem and Hovy, 2016) or *cyber bullying* (Zhong et al., 2016). While there may be nuanced differences in meaning[2], they are all compatible with the general definition above for abusive language.[3]

(1) stop editing this, you **dumbass**.
(2) Just want to slap the **stupid** out of these **bimbos**!!!
(3) Go lick a pig you arab muslim piece of **scum**.

Due to the rise of user-generated web content, in particular on social media networks, the amount of abusive language is also steadily growing. NLP methods are required to focus human review efforts towards the most relevant microposts.

In this paper, we address the task of detecting abusive words (e.g. *dumbass*, *bimbo*, *scum*). Our main assumption is that abusive words form a subset of negative polar expressions. The classification task is to **filter the abusive words from a given set of negative polar expressions**. We proceed as follows. On a base lexicon that is a small subset of negative polar expressions where the abusive words among them have been marked via crowdsourcing (§3), we calibrate a supervised classifier by examining various novel features (§4). A classifier trained on that base lexicon, which contains 551 abusive words, is then applied to a very large list of unlabeled negative polar expressions (from Wiktionary) to extract an expanded lexicon of 2989 abusive words (§5).

We extrinsically evaluate our new lexicon in the novel task of **cross-domain classification** of abusive documents (§6) where we use it as a *high-level* feature. In this work, we consider microposts as documents. While for in-domain classification, supervised classifiers trained on generic features, such as bag of words or word embeddings, usually score very well, on cross-domain classification they perform poorly since they latch on to domain-specific information. In subjectivity, polarity and emotion classification, high-level features based on predictive domain-independent word lists have been proposed to bridge the domain mismatch (Dias et al., 2009; Mohammad, 2012; Wiegand et al., 2013).

New abusive words constantly enter natural language. For example, according to Wiktionary[4] the word *gimboid*, which refers to an incompetent person, was coined in the British television series *Red Dwarf*, possibly from the word *gimp* and the suffix *-oid*. According to Urban Dictionary[5], the word *twunt*, which is a portmanteau of the swearwords *twat* and *cunt*, has been invented

[1] http://thelawdictionary.org/
[2] For example, several research efforts just focus on utterances addressed towards minorities.
[3] The examples in this work are included to illustrate the severity of abusive language. They are taken from actual web data and in no way reflect the opinion of the authors.
[4] https://en.wiktionary.org
[5] www.urbandictionary.com

Proceedings of NAACL-HLT 2018, pages 1046–1056
New Orleans, Louisiana, June 1 - 6, 2018.

by humourist Chris Morris for the Channel 4 series 'Jam' in 2000. One of the most recent abusive words is *remoaner* which describes someone who complains about or rejects the outcome of the 2016 EU referendum on the UK's membership of the European Union. It is a blend of *moan* and *remainer*. Wiktionary states that this word has a pejorative connotation.

These examples show that the task of creating a lexicon of abusive words cannot be reduced to a one-time manual annotation effort. Recent web corpora and crowdsourced dictionaries (e.g. Wiktionary) should be ideal resources to find evidence of such words.

Our **contribution** is that we present the first work that systematically describes the *automatic* construction of a lexicon of abusive words. We examine novel features derived from various textual resources. We show that the information we learn cannot be equally derived from a large dataset with labeled microposts. The effectiveness of our expanded lexicon is demonstrated on cross-domain detection of abusive microposts. This is also the first work to address this task in general. The supplementary material to this paper[6] includes all resources newly created for our research.

We frame our task as a **binary classification problem**. Each given expression is to be classified as either abusive or not. We study this problem on English. However, many of our features should also be applicable to other languages.

2 Related Work

Lexical knowledge for the detection of abusive language has only received little attention in previous work. Most approaches consider it as one feature among many. Very often existing word lists from the web are employed (Xiang et al., 2012; Burnap and Williams, 2015; Nobata et al., 2016). Their limited effectiveness may be due to the fact that they were not built for the task of abusive language detection. Only the manually-compiled lexicon from Razavi et al. (2010) and the lexicon of *hate verbs* from Gitari et al. (2015) have been compiled for this specific task. Since the latter lexicon is not publicly available we can only consider the former in our evaluation. In both publications, very little is said on the creation of these resources.

Previous work focused on in-domain classification, a setting where generic features (e.g. bag of words) work well and word lists are less important. There have been investigations examining features on various datasets (Nobata et al., 2016; Samghabadi et al., 2017), however, these studies always trained and tested on the *same* domain. We show that a lexicon-based approach is effective in cross-domain classification.

For a more detailed overview on previous work on the detection of abusive language in general, we refer the reader to Schmidt and Wiegand (2017).

3 Data

Base Lexicon. Our base lexicon exclusively comprises negative polar expressions. It is a small set which we have **annotated via crowdsourcing**. We consider abusive words to be a proper subset of negative polar expressions. By just focusing on these types of words, we are more likely to obtain a significant amount of abusive words than just considering a sample of arbitrary words. This lexicon will be used as a gold standard for calibrating features of a classifier. That classifier will be run on a large set of unlabeled negative polar expressions to produce our expanded lexicon (§5).

We sampled 500 negative nouns, verbs and adjectives each from the Subjectivity Lexicon (Wilson et al., 2005). We chose that lexicon since we have extra information available for its entries that we want to examine, namely polar intensity (§4.1.1) and sentiment views (§4.1.2). However, since we noted that the Subjectivity Lexicon misses some prototypical abusive words (e.g. *nigger*, *slut*, *cunt*) we added another 10% (i.e. 150 words) which are abusive words frequently occurring in the word lists mentioned in Schmidt and Wiegand (2017).

Each of the negative polar expressions was judged by 5 annotators from the crowdsourcing platform *ProlificAcademic*.[7] Each annotator had to be a native speaker of English and possess a task approval rate of at least 90%. For our base lexicon (Table 1), we considered a binary word categorization: *abusive* or *non-abusive*. A word was only classified *abusive* if at least 4 out of the 5 raters judged the word to be abusive. This threshold should prevent many ambiguous words from being classified as abusive, a general problem of existing resources (Davidson et al., 2017).

Corpora. In our experiments we employ three

[6] https://github.com/miwieg/naacl2018

[7] The supplementary material contains more information regarding our annotation set-up (including guidelines).

	adj		noun		verb		all	
class	freq	%	freq	%	freq	%	freq	%
abusive	170	33.8	291	45.3	90	17.8	551	33.4
not abusive	332	66.2	352	54.7	415	82.2	1099	66.6

Table 1: The base lexicon: 1650 entries in total of which 551 are abusive.

		intensity (§4.1.1)		views (§4.1.2)	
class	**all**	weak	strong	actor	speaker
abusive	26.7	14.1	32.0	9.7	32.8
not abusive	73.3	85.9	68.0	90.3	67.2

all numbers only refer to the subset of the base lexicon (Table 1) taken from the Subjectivity Lexicon (i.e. 1500 entries)

Table 3: Percentage of abusive/not abusive instances among (binary) intensity and views.

unlabeled corpora (Table 2). The two larger corpora, the *Amazon Review Corpus – AMZ* (Jindal and Liu, 2008) and the *Web As Corpus – WAC* (Baroni et al., 2009), are used for inducing word embeddings (§4.2). AMZ and the smallest corpus, *rateitall.com – RIA*[8], are used for computing polar word intensity (§4.1.1) from star ratings.

4 Feature Calibration

In the following, we describe the two types of features of our feature-based approach: novel linguistic features and generic word embeddings. They will be examined against some baselines on our base lexicon. As a classifier we use an SVM as implemented in SVM^{light} (Joachims, 1999). We chose that classifier since it is most commonly used for the detection of abusive language (Schmidt and Wiegand, 2017). *For all classifiers in this paper, the supplementary material*[6] *contains information regarding (hyper)parameter settings.*

4.1 Linguistic Features

4.1.1 Polar Intensity (INT)

Intuitively, abusive language should coincide with high polar intensity. We inspect 3 different types.

Binary Intensity (INT$_{bin}$). Our first feature is a simple binary intensity feature we obtain from the Subjectivity Lexicon. In that resource, each entry is categorized as either a *weak* polar expression (e.g. *dirty*) or a *strong* polar expression (e.g. *filthy*). Table 3 (left half), which shows the distribution of intensity on the intersection of our base lexicon and the Subjectivity Lexicon, confirms that abusive words are rarely weak polar expressions and more frequently strong polar expressions.

Fine-grained Intensity (INT$_{fine}$). We also investigate a more fine-grained feature which assigns a real-valued intensity score to polar expressions. It is computed by leveraging the star-rating assigned to the reviews comprising the AMZ corpus (Table 2), a large publicly available review corpus. A review is awarded between 1 and 5 stars where 1 is the most negative score. We infer the polar intensity of a word by the distribution of star-ratings associated with the reviews in which it occurs. We assume negative polar expressions with a very high polar intensity to occur significantly more often in reviews assigned few stars (i.e. 1 or 2). Ruppenhofer et al. (2014) established that the most effective method to derive such polar intensity is by ranking words by their *weighted mean of star ratings* (Rill et al., 2012). All words of our base lexicon are ranked according to that score. As a feature we use the rank of a word.

Intensity Directed towards Persons (INT$_{person}$). Not all negative polar expressions with a high intensity are equally likely to be abusive. The high intensity expressions should also be words typically directed towards persons. Most polar statements in AMZ, however, are directed towards a movie, book or some electronic product. In order to extract negative polar intensity directed towards persons, we replace the AMZ corpus with the RIA corpus (Table 2). RIA contains reviews on arbitrary entities rather than just commercial products as in the case of AMZ. Each review has a category label (e.g. *computer*, *person*, *travel*) that very easily allows us to extract from RIA just those reviews that concern persons.

Table 4 compares a typical 1-star review from AMZ with one from RIA. We consider the RIA-review an abusive comment. It contains many words predictive of abusive language (e.g. *self-absorbed*, *loser*, *arrogant* or *loud-mouthed*).

4.1.2 Sentiment Views (VIEW)

Wiegand et al. (2016b) define sentiment views as the perspective of the opinion holder of polar expressions. They distinguish between expressions conveying the view of the implicit speaker of the utterance typically referred to as *speaker views* (e.g. *cheating* in (4); *ugly* and *stinks* in (5)), and expressions conveying the view of event participants typically referred to as *actor views* (e.g. *disappointed* and *horrified* in (6); *protested* in (7)).

[8]This is a crawl from the review website www.rateitall.com.

corpus	size	properties	purpose
RateItAll (**RIA**)	4.7M	review corpus focused on persons	computation of polar intensity (§4.1.1)
Amazon (**AMZ**)	1.2B	product review corpus	comp. of polar intensity (§4.1.1)/word embeddings (§4.2)
Web as Corpus (**WAC**)	2.3B	large general web corpus	computation of word embeddings (§4.2)

Table 2: Information about unlabeled corpora used (*by size we mean the number of tokens*).

AMZ	*on Halloween 5:* this movie is horrible with a bad plot a disappointment to the halloween series.
RIA	*on Bill Maher:* Self-absorbed loser who tries to pretend to be fair. He is rude, arrogant, loud-mouthed...

Table 4: 1-star reviews in different corpora.

WAC	liar (19), coward (7), name (6), idiot (6), hero (5), horse (5), saint (5), fool (5), snob (4), genius (4)
Twitter	bitch (1534), hoe (432), liar (317), cunt (274), whore (254), pussy (228), nigger (226), loser (217), faggot (217), slut (197)

Table 5: Comparison of the 10 most frequent pattern matches (*numbers in brackets indicate frequency*).

Wiegand et al. (2016b) provided sentiment-view annotations for the entries of the Subjectivity Lexicon.

(4) Peter is always **cheating**$_{speaker\ view}$. *(holder: speaker)*
(5) Mary is an **ugly**$_{speaker\ view}$ girl that **stinks**$_{speaker\ view}$. *(holder: speaker)*
(6) [Peter]$_{holder}$ was **disappointed**$_{actor\ view}$ and **horrified**$_{actor\ view}$ at the same time.
(7) [The public]$_{holder}$ **protested**$_{actor\ view}$ against that law.

Sentiment views have been used for improving the extraction of opinion holders and targets (Deng and Wiebe, 2016; Wiegand et al., 2016a). In this paper we show that they also have relevance for the detection of abusive words. Among actor-view words, there is a much lower proportion of abusive words than among speaker-view words (right half of Table 3). This can be explained by the fact that verbal abuse usually originates from the speaker of an utterance rather than some other discourse entity. We use sentiment-view information as a binary feature.

4.1.3 Emotion Categories (NRC)

We also examine whether knowledge of emotion categories associated with words is helpful. Potentially negative emotions, such as *disgust* or *anger*, should correlate with abusive words. We use the NRC lexicon (Mohammad and Turney, 2013) and employ the categories associated with the words contained in that resource as a feature.

4.1.4 Patterns (PAT)

Noun Pattern (PAT$_{noun}$). We found that the noun pattern (8) can be used to extract abusive nouns. Since this pattern is very sparse even on our largest corpus (i.e. WAC), we also run our pattern as a query on Twitter and extracted all matching tweets coming in a time period of 14 days. (We observed that by then we had reached a saturation point.)

(8) *pattern*: called {me|him|her} a(n) <noun>
(9) *pattern match example*: He called me a **bitch**.

Table 5 compares the most frequent matches for that pattern. Our pattern matches much more frequently on Twitter than on WAC. The quality of the matches on Twitter is also much better than on WAC, where we still find many false positives (e.g. *name* or *saint*). We assume that tweets, in general, are much more negative in tone than arbitrary web documents (as represented by WAC) which could explain the fewer false positives on Twitter. Note that the ranking from Twitter is not restricted to just prototypical abusive words (as Table 5 might suggest). The entire ranking also contains many less common words, such as *weaboo*, *dudebro* or *butterface*. The frequency ranks of the nouns extracted from Twitter are used as a feature.

Adjective Pattern (PAT$_{adj}$). Abusive adjectives often modify an abusive noun as in *brainless idiot*, *smarmy liar* or *gormless twat*. Therefore, we mined Twitter for adjectives modifying mentions of our extracted nouns (PAT$_{noun}$). (We were not able to find a construction identifying abusive verbs, so our output from PAT includes no verbs.)

4.1.5 WordNet (WN) and Wiktionary (WK)

We compare WordNet (Miller et al., 1990) and Wiktionary[4] as two general-purpose lexical resources. Unlike WordNet, Wiktionary is produced collaboratively by volunteers rather than linguistic experts. It contains more abusive words from our base lexicon, i.e. 97% (WK) vs. 87% (WN).

A common way to harness a general-purpose lexicon for induction tasks in sentiment analysis is by using its **glosses** (Choi and Wiebe, 2014; Kang et al., 2014). Assuming that the explanatory texts

of glosses are similar among abusive words, we treat glosses as a bag-of-words feature.

We also exploit information on **word usage**. Many abusive words are marked with tags such as *pejorative*, *derogatory* or *vulgar*. Both WordNet and Wiktionary contain such information. However, in Wiktionary more than 6 times as many of our entries include a tag compared to WordNet.

In order to incorporate a semantic representation more general than individual words, we employ **supersenses**. Supersenses are only contained in WordNet. They represent a set of 45 classes into which entries are categorized. They have been found effective for sentiment analysis (Flekova and Gurevych, 2016). Some categories correlate with abusive words. For example, 76% of the words of our base lexicon that belong to the supersense *person* (e.g. *loser*, *idiot*) are abusive words.

4.1.6 FrameNet (FN)

FrameNet (Baker et al., 1998) is a semantic resource which provides over 1200 semantic frames that comprise words with similar semantic behaviour. We use the frame-memberships of a word as features, expecting that abusive and non-abusive words occur in separate frames.

4.2 Generic Features: Word Embeddings

We induce word embeddings from the two largest corpora, i.e. AMZ and WAC (Table 2) using *Word2Vec* (Mikolov et al., 2013) in default configuration (i.e. 200 dimensions; cbow). The best performance was obtained by concatenating for each word the vectors induced from the two corpora.[9]

4.3 Baselines to Feature-based Approach

In addition to a majority-class classifier we consider the following baselines:

Weak Supervision (WSUP). With this baseline we want to build a lightweight classifier that does not require proper labeled training data. It is inspired by previous induction approaches for sentiment lexicons, such as Hatzivassiloglou and McKeown (1997) or Velikovich et al. (2010) which heuristically label some seed instances and then apply graph-based propagation to label the remaining words of a dataset. On the basis of word embeddings (§4.2), we build a word-similarity graph, where the nodes represent our negative polar expressions and each edge denotes the semantic similarity between two arbitrary words. We compute it by the cosine of their word-embedding vectors. The output of PAT from Twitter (§4.1.4) is considered as positive class seed instances. We chose PAT since it is an effective feature that does not depend on a lexical resource. As negative class seeds, we use the most frequent words in the WAC corpus (Table 2). Our rationale is that high-frequency words are unlikely to be abusive. We chose WAC instead of Twitter since the evidence of PAT (Table 5) suggested less abusive language in that corpus. This word-similarity graph is illustrated in Figure 1. In order to propagate the labels to the unlabeled words from the seeds, we use the Adsorption algorithm (Talukdar et al., 2008).

Using Labeled Microposts (MICR). With our last baseline we examine in how far we can detect abusive words by only using information from labeled microposts rather than labeled words. These experiments are driven by the fact that labeled microposts already exist. We consider two methods using the largest dataset comprising manually labeled microposts, *Wulczyn* (Table 8). The class labels of the microposts and our base lexicon (§3) are the same. Our aim is to produce a ranking of words where the high ranks represent words more likely to be abusive. Since we want to produce a strong baseline, we consider the best possible cut-off rank (*see supplementary material*[6]). Every word higher than this rank is considered abusive and all other words not abusive.

The first method **MICR:pmi** ranks the words of our base lexicon by their Pointwise Mutual Information with the class label *abusive* that is assigned to microposts. To be even more competitive, we introduce a second method **MICR:proj** that learns a projection of embeddings. MICR:proj has the advantage over MICR:pmi that it does not only rank words observed in the labeled microposts but all words represented by embeddings. Since our embeddings (§4.2) are induced on the combination of AMZ and WAC corpora, which together are about 360 times the size of the *Wulczyn* dataset, MICR:proj is likely to cover more abusive words. Let $\mathbf{M} = [\mathbf{w_1}, \ldots, \mathbf{w_n}]$ denote a labeled micropost of n words. Each column $\mathbf{w} \in \{0,1\}^v$ of $\mathbf{M}$ represents a word in a one-hot form. Our aim is learning a one-dimensional projection $\mathbf{S} \cdot \mathbf{E}$ where $\mathbf{E} \in \mathbb{R}^{e \times v}$ represents our unsupervised embeddings of dimensionality e over the vocabulary size v (§4.2) and $\mathbf{S} \in \mathbb{R}^{1 \times e}$ represents the learnt

[9] We also ran experiments with pretrained embeddings from *GoogleNews* but they did not improve classification.

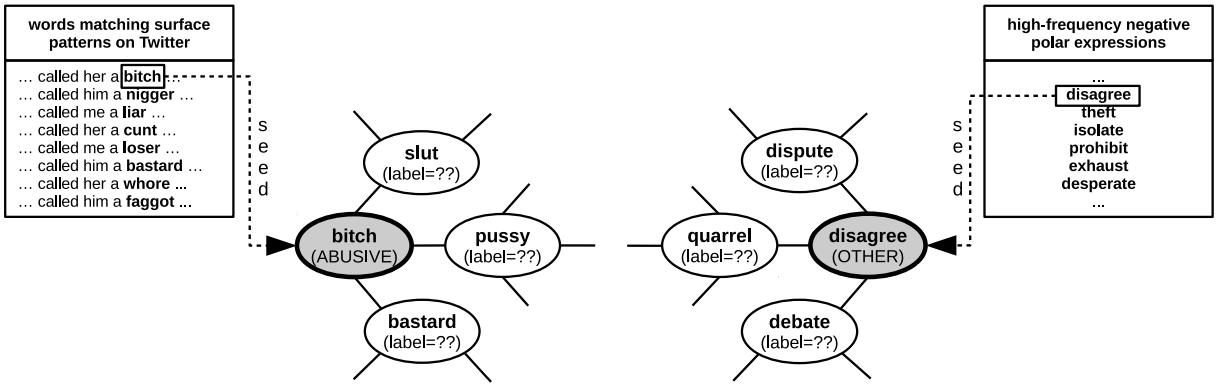


Figure 1: Illustration of word-similarity graph as used for weakly-supervised baseline (WSUP); seeds for abusive words (e.g. *bitch*) are obtained by the output of feature PAT (§4.1.4); seeds for non-abusive words (e.g. *disagree*) are high-frequency negative polar expressions.

classifier	Prec	Rec	F1
MAJORITY	33.3	50.0	40.0
MICR:pmi	65.3	59.5	62.2†
MICR:proj	67.1	64.6	65.8*†
WSUP	77.3	71.0	74.0*†
SVM:embeddings	77.6	73.9	75.7*
SVM:linguistic	81.6	73.8	77.5*
SVM:linguistic+WSUP	82.5	76.5	79.4*†
SVM:linguistic+embeddings	81.6	79.7	80.7*
SVM:linguistic+embed.+WSUP	**82.9**	**80.4**	**81.6**†

statistical significance testing (paired t-test at $p < 0.05$): *: better than previous line but 1; †: better than previous line

Table 6: Different classifiers on base lexicon (Table 1).

projection matrix. We compute a projected micropost $\mathbf{h} = \mathbf{S} \cdot \mathbf{E} \cdot \mathbf{M}$ which is an n-dimensional vector. Each component represents a word from the micropost. The value represents the predictability of the word towards being abusive. We then apply a bag-of-words assumption to use that projected micropost to predict the binary class label y: $p(y|\mathbf{M}) \propto exp(\mathbf{h} \cdot \mathbf{1})$ where $\mathbf{1} \in \{1\}^n$. This model is a feed-forward network trained using Stochastic Gradient Descent (Rumelhart et al., 1986). On the basis of the projected embeddings we rank our negative polar expressions.

4.4 Evaluation of Features on Base Lexicon

We conduct experiments on our base lexicon (Table 1) and report macro-average precision, recall and f-score. SVMs are evaluated on a 10-fold crossvalidation. Table 6 displays the performance of the different classifiers. The least effective information source are labeled microposts (MICR), though, as expected, the projected embeddings (MICR:proj) outperform PMI. The performance of weak supervision (WSUP) outperforms MICR.

Among the SVM configurations, embeddings are already effective. The linguistic features outperform all other methods. The best classifier is an SVM trained on embeddings, linguistic features and the output of WSUP as a further feature.[10]

Table 7 shows the performance of SVMs using different linguistic features (§4.1). Among the three intensity types, the most effective one is the person-based intensity (INT$_{person}$). However, it can be effectively combined with the remaining types. Among the lexical sentiment resources used (i.e. NRC, INT$_{bin}$ and VIEW), VIEW is most effective. Their combination also results in an improvement. The surface patterns (PAT) are surprisingly predictive. Of the general-purpose lexical resources (i.e. WN, WK and FN), WN and WK are both very effective resources. Glosses from WN are the strongest individual feature. Combining WK, WN and FN results in significant improvement. The best feature set combines all features.

Our results also suggest that for languages other than English, there are some very strong features, such as PAT, WK or embeddings, that could be easily adopted since they do not depend on a resource which is only available in English.

5 Expanding the Lexicon

We produce a large **feature-based lexicon** of abusive words by classifying all (unlabeled) negative polar expressions from Wiktionary. We chose Wiktionary since our previous experiments indicated a high coverage of abusive words on that resource (§4.1.5). The negative polar expressions

[10] We did not include MICR among the further features, as they are trained on the labeled microposts that we also use as test data in the extrinsic evaluation (§6).

features used in SVM	Prec	Rec	F1
MAJORITY	33.3	50.0	40.0
INT_{fine}	62.0	57.0	59.4†
INT_{bin}	61.7	60.4	61.0*
INT_{person}	70.8	55.4	62.1*
INT_{fine}+INT_{bin}+INT_{person}	70.8	60.7	65.3*†
NRC	60.2	60.1	60.2
VIEW	65.6	62.8	64.2†
INT_{bin}+NRC+VIEW	66.9	68.8	67.9*†
PAT_{noun}	79.9	58.4	67.4
PAT_{noun}+PAT_{adj}	76.4	63.2	69.1
WN_{usage}	82.6	52.6	64.3
FN	66.3	66.4	66.4
WK_{usage}	76.7	61.0	67.9*†
WK_{gloss}	74.8	64.9	69.5*†
WN_{super}	78.7	64.9	71.1*†
WN_{gloss}	75.9	67.4	71.4*
WN_{usage}+WN_{super}+WN_{gloss}	76.7	68.0	72.0*
WK_{usage}+WK_{gloss}	79.5	67.0	72.7*
all WN + *all* WK	80.0	68.7	73.9*
all WN + *all* WK + FN	80.3	69.5	74.5*
all from above	**81.6**	**73.8**	**77.5***†

statistical significance testing (paired t-test at $p < 0.05$): *: better than previous line but 1; †: better than previous line

Table 7: Performance of the different linguistic features on base lexicon (Table 1).

are identified by applying to the vocabulary of Wiktionary an SVM trained on the words from the Subjectivity Lexicon with their respective polarities. As features, we use word embeddings (§4.2). In order to produce the feature-based lexicon of abusive words another SVM is trained on our base lexicon (Table 1) using the best feature set from Table 6. With 2989 abusive words, our expanded lexicon is 5 times as large as the base lexicon.

In order to measure the impact of our proposed features on the quality of the resulting lexicon, we devised an alternative expansion which just employs word embeddings. For this, we used **SentProp**, the most effective induction method from the *SocialSent* package (Hamilton et al., 2016).[11]

6 Cross-domain Classification

6.1 Motivation and Set Up

We now apply our expanded lexicon (§5) to the classification of abusive microposts, i.e. we classify entire comments rather than words out of context. Table 8 shows the datasets of labeled microposts that we use. The difference between these datasets is the source from which they originate. Consequently, different topics are represented in the different datasets. Still, we find similar types

[11] Since SentProp produces a ranking rather than a classification, we consider 2989 as a cut-off value to separate the instances into 2 classes. This corresponds to the size of abusive words predicted by our feature-based lexicon (Table 9).

dataset	size†	abusive	source
(Warner and Hirschberg, 2012)	3438	14.3%	*diverse*
(Waseem and Hovy, 2016)	16165	35.3%	Twitter
(Razavi et al., 2010)	1525	31.9%	UseNet
(Wulczyn et al., 2017)	115643	11.6%	Wikipedia

†: total number of microposts in the dataset

Table 8: Datasets comprising labeled microposts.

of abusive language (e.g. *racism*, *sexism*). For example, both (10)-(11) from *Waseem* and (12) from *Wulczyn* are sexist comments[12] but (10)-(11) discuss the role of women in sports while (12) addresses women's hygiene in Slavic countries.

(10) *from Waseem dataset:* maybe that's where they should focus? Less **cunts** on *football*.

(11) *from Waseem dataset:* I would rather brush my teeth with sandpaper then watch *football* with a girl!!

(12) *from Wulczyn dataset:* slavic women don't like to wash ... Their **pussy** stinks.

Since our aim is to produce the best possible cross-domain classifier, **all classifiers are trained on one dataset and tested on another**. This is a real-life scenario. Often when a classifier for abusive microposts is needed, sufficient labeled data is only available for other text domains.

Having different topics in training and test data makes cross-domain classification difficult. For example, since a large proportion of sexist comments in *Waseem* relate to sports, traditional supervised classifiers (using bag of words or word embeddings) will learn correlations between words of that domain with the class labels. For instance, the domain-specific word *football* occurs frequently in *Waseem* (i.e. 90 occurrences) with a strong correlation towards abusive language (precision: 95%). Other words, such as *sports* and *commentator*, display a similar behaviour. A supervised classifier will assign a high weight to such words. While such domain-specific words may aid in-domain classification and enable a correct classification of microposts, such as (11), we will show that it has a detrimental effect on cross-domain classification. We claim that the predictive words that abusive comments share across different domains are abusive words, just of the sort that our expanded lexicon contains, e.g. *cunts* in (10) and *pussy* in (12).

Our **proposed classifier** for labeling *microposts* is an SVM trained on features derived from our expanded lexicon (§5). We do not use a binary feature encoding the presence of abusive words. Instead, we rank all abusive words of our lexicon

[12] (12) is also a racist comment.

baseline lexicons		newly created lexicons	
lexicon	**entries**	**lexicon**	**entries**
Hatebase	430	base (Table 1)	551
Derogatory	1609	expanded:SentProp (§5)	2989
Ottawa	1746	expanded:feature-based (§5)	2989

Table 9: Lexicons used in cross-domain classification of microposts (*figures denote the amount of unigrams*).

classifier	Razavi	Warner	Waseem	Wulczyn
expand.:feature-b. (SVM)	75.7	64.8	63.8	78.4
FastText	**83.4**	71.8	76.3	85.6
RNN	74.8	70.5	78.0	86.9
Yahoo (SVM)	82.4	**78.2**	**84.1**	**90.0**

Table 10: In-domain classification of microposts (*eval.: F1-score*).

according to the confidence score of the classifier it produced and use their ranks as features.

As **baseline classifiers** we consider publicly available word lists (Table 9). We include the resource from Razavi et al. (2010), henceforth referred to as ***Ottawa***, the entries of ***Hatebase***[13], which has been used in Nobata et al. (2016) and Davidson et al. (2017), and the derogatory words from Wiktionary (***Derogatory***)[14].[15] Finally, we also include our **base** lexicon (Table 1) in order to evaluate the expansion process of our two expanded lexicons (§5). For all lists, we train on a single feature indicating the frequency of abusive words in a micropost to be classified. *Ottawa* also contains weights assigned to abusive words. We weight the observed frequency with these weights.

We further evaluate 3 classifiers representing the state of the art of in-domain evaluations: ***FastText*** (Joulin et al., 2017), Gated Recurrent Units Recurrent Neural Networks ***RNN***, which have been reported to work best on English microposts (Pavlopoulos et al., 2017), and ***Yahoo***, an SVM trained on the sophisticated feature set proposed by Nobata et al. (2016). Next to character and token n-grams, *Yahoo* includes word and comment embeddings, syntactic features and some linguistic diagnostics.

[13] www.hatebase.org

[14] https://en.wiktionary.org/wiki/Category:English_derogatory_terms

[15] There are also similar but smaller lists in Wiktionary, e.g. *offensive terms*. They produced no better results.

		Yahoo		feature-b. lex.	
test	**train**	**all**	**explicit**	**all**	**explicit**
Warner	Razavi	55.4	65.2	65.0	80.6
	Waseem	58.1	55.9	64.6	79.0
	Wulczyn	60.2	72.8	63.4	80.7
	Average	57.9	64.6	64.3	80.1
Waseem	Warner	58.5	61.2	63.3	62.0
	Razavi	61.1	63.1	58.7	78.8
	Wulczyn	51.2	68.2	62.9	78.5
	Average	56.9	64.2	61.6	73.1

Table 12: Cross-domain classification of microposts: *all* test data vs. *explicit* subset (*eval.: F1-score*).

6.2 Results

In Table 10, we list the performance of the 3 state-of-the-art classifiers along with our proposed classifier using our expanded lexicon on in-domain 10-fold crossvalidation. Due to space limitations, we cannot list the other classifiers. We *only* provide this list to demonstrate the strength of the state-of-the-art classifiers on in-domain evaluation. On this setting, a lexicon-based approach is not competitive since domain-specific information is not included. However, as we show in Table 11, for cross-domain classification, it is exactly that property that ensures that our feature-based lexicon provides best performance. Compared to the in-domain setting, *FastText*, *RNN* and *Yahoo* display a huge drop in performance. They all suffer from overfitting to domain-specific knowledge.

Of all lexicons, our proposed feature-based lexicon performs best. We were surprised by the poor performance of *Hatebase* but attribute this to its small size and the high amount of ambiguous (and debatable) entries, such as *Charlie*, *pancake*, *Pepsi*. Although our feature-based lexicon is the largest of all tested (i.e. 2989 words), our experiments do not support the general rule that larger lexicons always outperform smaller ones. For instance, already our base lexicon with 551 abusive words is much better than the lexicons *Derogatory* or *Ottawa* which are about 3 times larger (Table 9). Each word in our base lexicon was only included if 4 out of 5 raters judged it to be abusive. This ensured a fairly reliable annotation. In contrast, *Derogatory* and *Ottawa* suffer from many ambiguous entries (e.g. *bag*, *Tim*, *yellow*). The high precision of our base lexicon is what ensures that our expanded lexicon does not include much noise.

Another shortcoming of most of the other existing lexicons is that they overwhelmingly focus on nouns. While nouns undoubtedly represent the most frequent abusive terms, there is, however, a substantial number of abusive words that belong to other parts of speech, particularly adjectives (e.g. *vile*, *sneaky*, *slimy*, *moronic*). In our base lexicon, more than 30% of the abusive words are of that part of speech. Our expanded lexicon,

datasets					SVM						
						baseline lexicons			newly created lexicons		
test	training	majority	FastText	RNN	Yahoo	Hatebase	Derogat.	Ottawa	base	SentProp	feature-b.
Razavi	Warner	40.50	50.59	53.76	53.40	40.50	40.50	60.95	61.08	64.20	**66.13**
	Waseem	40.50	51.64	53.39	51.66	44.29	51.35	63.13	69.69	63.12	**74.15**
	Wulczyn	40.50	71.74	71.59	**75.10**	40.50	40.50	40.50	40.50	68.50	74.83
	Average	40.50	57.99	59.58	60.05	41.76	44.12	54.86	57.09	66.27	**71.70**
Warner	Razavi	46.14	57.73	48.99	55.42	46.14	57.49	59.81	63.57	**67.57**	64.98
	Waseem	46.14	61.45	57.63	56.54	63.52	57.49	**64.67**	63.57	62.75	64.64
	Wulczyn	46.14	58.35	57.36	60.19	46.14	46.14	46.14	46.14	**65.34**	63.35
	Average	46.14	59.18	54.66	57.38	51.93	53.71	56.87	57.76	**65.22**	64.32
Waseem	Razavi	40.62	60.91	54.67	57.83	40.62	52.66	52.95	57.33	**64.56**	63.32
	Warner	40.62	58.28	58.85	**60.65**	40.62	40.62	40.62	54.93	51.98	58.66
	Wulczyn	40.62	56.33	54.13	51.76	40.62	40.62	40.62	40.62	50.27	**62.90**
	Average	40.62	58.51	55.88	56.75	40.62	44.63	44.73	50.96	55.60	**61.63**
Wulczyn	Razavi	46.88	64.65	64.43	70.70	46.88	50.97	57.70	69.56	67.69	**73.71**
	Warner	46.88	56.21	56.13	52.73	46.88	46.88	55.93	59.55	66.38	**70.06**
	Waseem	46.88	52.66	57.33	51.23	43.51	50.97	60.08	69.56	66.38	**72.39**
	Average	46.88	57.84	59.30	58.22	45.76	49.61	57.90	66.22	63.52	**72.05**

Table 11: Different classifiers on **cross-domain** classification of microposts; best result in **bold**; (*eval.: F1-score*).

which roughly preserves that ratio, includes about 800 adjectives in total. Since abusive adjectives often co-occur with abusive nouns (§4.1.4), they may compensate for abusive nouns that are missing from the lexicon. Such unknown nouns often occur when authors of microposts try to obfuscate their abusive language, e.g. *sneaky assh0le*, *slimy b*st*rd*. Interestingly, the modifying adjectives are not obfuscated, probably because they are considered slightly less offensive in tone.

Given that among the newly created lexicons our feature-based expanded lexicon performs best, we conclude that the expansion is effective (since we improve over the base lexicon), and the features are more effective than a generic induction approach (i.e. *SentProp*).

6.3 Explicitly vs. Implicitly Abusive Microposts

The results in Table 11 also show that the cross-domain performance of our proposed feature-based lexicon is lower on the two datasets *Warner* and *Waseem*. We observed that while on the other two datasets almost all abusive microposts can be considered *explicitly abusive* posts, i.e. they contain abusive words, a large proportion of microposts labeled abusive in *Warner* and *Waseem* are *implicitly abusive* (Waseem et al., 2017), i.e. the abuse is conveyed by other means, such as sarcasm or metaphorical language (11). We asked raters from Prolific Academic to identify explicitly abusive microposts by marking abusive words in those posts. The annotators were not given access to any lexicon of abusive words. We then conducted cross-domain classification on those subsets where the abusive instances were only those rated as explicit. The results are displayed in Table 12. The table shows that our feature-based lexicon is much better on this subset, while the most sophisticated supervised classifier (*Yahoo*) still performs worse. From that we conclude that only *explicitly* abusive microposts can be reliably detected in cross-domain classification.

7 Conclusion

We examined the task of inducing a lexicon of abusive words. We presented novel features including surface patterns, sentiment views, polar intensity and general purpose lexical resources, particularly Wiktionary. The information we thus acquire cannot be learnt all that effectively from labeled microposts, not even with a projection-based classifier. While a lexicon of abusive words can only aid the detection of explicit abuse, its effectiveness was demonstrated on the novel task of cross-domain detection of abusive microposts, where our domain-independent lexicon outperforms previous supervised classifiers which suffer from overfitting to domain-specific features.

Acknowledgements

The authors would like to thank Thomas Kleinbauer, Katja Markert and Ines Rehbein for feedback on earlier drafts of this paper. We are also grateful to William Warner and Diana Inkpen for granting us access to their data on abusive language detection. Special thanks go to Stefan Kazalski for crawling the *rateitall*-website. We also give thanks to John Pavlopoulos for helping us reconstructing the configurations of his RNN. The authors were partially supported by the German Research Foundation (DFG) under grants RU 1873/2-1 and WI 4204/2-1.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the International Conference on Computational Linguistics and Annual Meeting of the Association for Computational Linguistics (COLING/ACL)*. Montréal, Quebec, Canada, pages 86–90.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetti. 2009. The WaCky Wide Web: A Collection of Very Large Linguistically Processed Web-Crawled Corpora. *Language Resources and Evaluation* 43(3):209–226.

Pete Burnap and Matthew L. Williams. 2015. Cyber Hate Speech on Twitter: An Application of Machine Classification and Statistical Modeling for Policy and Decition Making. *Policy & Internet* 7(2):223–242.

Yoonjung Choi and Janyce Wiebe. 2014. +/-EffectWordNet: Sense-level Lexicon Acquisition for Opinion Inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar, pages 1181–1191.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of the International AAAI Conference on Weblogs and Social Media (ICWSM)*. Montréal, Canada.

Lingjia Deng and Janyce Wiebe. 2016. Recognizing Opinion Sources Based On A New Categorization Of Opinion Types. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. New York City, NY, USA, pages 2775–2781.

Gaël Dias, Dinko Lambov, and Veska Noncheva. 2009. High-level Features for Learning Subjective Language across Domains. In *Proceedings of the International AAAI Conference on Weblogs and Social Media (ICWSM)*. San Jose, CA, USA.

Lucie Flekova and Iryna Gurevych. 2016. Supersense Embeddings: A Unified Model for Supersense Interpretation, Prediction, Utilization. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. Berlin, Germany, pages 2029–2041.

Njagi Dennis Gitari, Zhang Zuping, Hanyurwimfura Damien, and Jun Long. 2015. A Lexicon-based Approach for Hate Speech Detection. *International Journal of Multimedia and Ubiquitous Engineering* 10(4):2015–230.

William L. Hamilton, Kevin Clark, Jure Leskovec, and Dan Jurafsky. 2016. Inducing Domain-Specific Sentiment Lexicons from Unlabeled Corpora. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Austin, TX, USA, pages 595–605.

Vasileios Hatzivassiloglou and Kathleen R. McKeown. 1997. Predicting the Semantic Orientation of Adjectives. In *Proceedings of the Conference on European Chapter of the Association for Computational Linguistics (EACL)*. Madrid, Spain, pages 174–181.

Nitin Jindal and Bing Liu. 2008. Opinion Spam and Analysis. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*. Palo Alto, CA, USA, pages 219–230.

Thorsten Joachims. 1999. Making Large-Scale SVM Learning Practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, MIT Press, pages 169–184.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of Tricks for Efficient Text Classification. In *Proceedings of the Conference on European Chapter of the Association for Computational Linguistics (EACL)*. Valencia, Spain, pages 427–431.

Jun Seok Kang, Song Feng, Leman Akoglu, and Yejin Choi. 2014. ConnotationWordNet: Learning Connotation over the Word+Sense Network. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. Baltimore, MD, USA, pages 1544–1554.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of Workshop at the International Conference on Learning Representations (ICLR)*. Scottsdale, AZ, USA.

George Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. 1990. Introduction to WordNet: An On-line Lexical Database. *International Journal of Lexicography* 3:235–244.

Saif Mohammad. 2012. Portable Features for Classifying Emotional Text. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL (HLT/NAACL)*. Montréal, Canada, pages 587–591.

Saif Mohammad and Peter Turney. 2013. Crowdsourcing a Word-Emotion Association Lexicon. *Computational Intelligence* 39(3):555–590.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive Language Detection in Online User Content. In *Proceedings of the International Conference on World Wide Web (WWW)*. Republic and Canton of Geneva, Switzerland, pages 145–153.

John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2017. Deeper Attention to Abusive User Content Moderation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Copenhagen, Denmark.

Amir Hossein Razavi, Diana Inkpen, Sasha Uritsky, and Stan Matwin. 2010. Offensive Language Detection Using Multi-level Classification. In *Proceedings of the Canadian Conference on Artificial Intelligence*. Ottawa, Canada, pages 16–27.

Sven Rill, Johannes Drescher, Dirk Reinel, Joerg Scheidt, Oliver Schuetz, Florian Wogenstein, and Daniel Simon. 2012. A Generic Approach to Generate Opinion Lists of Phrases for Opinion Mining Applications. In *Proceedings of the KDD-Workshop on Issues of Sentiment Discovery and Opinion Mining (WISDOM)*. Beijing, China.

David. E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Parallel distributed processing: explorations in the microstructure of cognition. In *Learning internal representations by error propagation*, MIT Press Cambridge, pages 318–362.

Josef Ruppenhofer, Michael Wiegand, and Jasper Brandes. 2014. Comparing methods for deriving intensity scores for adjectives. In *Proceedings of the Conference on European Chapter of the Association for Computational Linguistics (EACL)*. Gothenburg, Sweden, pages 117–122.

Niloofar Safi Samghabadi, Suraj Maharjan, Alan Sprague, Raquel Diaz-Sprague, and Thamar Solorio. 2017. Detecting Nastiness in Social Media. In *Proceedings of the ACL-Workshop on Abusive Language Online*. Vancouver, Canada.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection using Natural Language Processing. In *Proceedings of the EACL-Workshop on Natural Language Processing for Social Media (SocialNLP)*. Valencia, Spain, pages 1–10.

Partha Pratim Talukdar, Joseph Reisinger, Marius Pasca, Deepak Ravichandran, Rahul Bhagat, and Fernando Pereira. 2008. Weakly-Supervised Acquisition of Labeled Class Instances using Graph Random Walks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Honolulu, HI, USA, pages 582–590.

Leonid Velikovich, Sasha Blair-Goldensohn, Kerry Hannan, and Ryan McDonald. 2010. The Viability of Web-derived Polarity Lexicons. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL (HLT/NAACL)*. Los Angeles, CA, USA, pages 777–785.

William Warner and Julia Hirschberg. 2012. Detecting Hate Speech on the World Wide Web. In *Proceedings of the Workshop on Language in Social Media (LSM)*. Montréal, Canada, pages 19–26.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of the ACL-Workshop on Abusive Language Online*. Vancouver, BC, Canada, pages 78–84.

Zeerak Waseem and Dirk Hovy. 2016. Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL – Student Research Workshop)*. San Diego, CA, USA, pages 88–93.

Michael Wiegand, Christine Bocionek, and Josef Ruppenhofer. 2016a. Opinion Holder and Target Extraction on Opinion Compounds – A Linguistic Approach. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL (HLT/NAACL)*. San Diego, CA, USA, pages 800–810.

Michael Wiegand, Manfred Klenner, and Dietrich Klakow. 2013. Bootstrapping polarity classifiers with rule-based classification. *Language Resources and Evaluation* 47(4):1049–1088.

Michael Wiegand, Marc Schulder, and Josef Ruppenhofer. 2016b. Separating Actor-View from Speaker-View Opinion Expressions using Linguistic Features. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL (HLT/NAACL)*. San Diego, CA, USA, pages 778–788.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing Contextual Polarity in Phrase-level Sentiment Analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT/EMNLP)*. Vancouver, BC, Canada, pages 347–354.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex Machina: Personal Attacks Seen at Scale. In *Proceedings of the International Conference on World Wide Web (WWW)*. Perth, Australia, pages 1391–1399.

Guang Xiang, Bin Fan, Ling Wang, Jason Hong, and Carolyn Rose. 2012. Detecting Offensive Tweets via Topical Discovery over a Large Scale Twitter Corpus. In *Proceedings of the Conference on Information and Knowledge Management (CIKM)*. Maui, HI, USA, pages 1980–1984.

Haoti Zhong, Hao Li, Anna Cinzia Squicciarini, Sarah Michele Rajtmajer, Christopher Griffin, David J. Miller, and Cornelia Caragea. 2016. Content-Driven Detection of Cyberbullying on the Instagram Social Network. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. New York City, NY, USA, pages 3952–3958.

Author Commitment and Social Power: Automatic Belief Tagging to Infer the Social Context of Interactions

Vinodkumar Prabhakaran
Stanford University
vinod@cs.stanford.edu

Premkumar Ganeshkumar
Agolo, Inc.
prem@agolo.com

Owen Rambow
Elemental Cognition, Inc.
owenr@elementalcognition.com

Abstract

Understanding how social power structures affect the way we interact with one another is of great interest to social scientists who want to answer fundamental questions about human behavior, as well as to computer scientists who want to build automatic methods to infer the social contexts of interactions. In this paper, we employ advancements in extra-propositional semantics extraction within NLP to study how author commitment reflects the social context of an interactions. Specifically, we investigate whether the level of commitment expressed by individuals in an organizational interaction reflects the hierarchical power structures they are part of. We find that subordinates use significantly more instances of non-commitment than superiors. More importantly, we also find that subordinates attribute propositions to other agents more often than superiors do — an aspect that has not been studied before. Finally, we show that enriching lexical features with commitment labels captures important distinctions in social meanings.

1 Introduction

Social power is a difficult concept to define, but is often manifested in how we interact with one another. Understanding these manifestations is important not only to answer fundamental questions in social sciences about power and social interactions, but also to build computational models that can automatically infer social power structures from interactions. The availability and access to large digital repositories of naturally occurring social interactions and the advancements in natural language processing techniques in recent years have enabled researchers to perform large scale studies on linguistic correlates of power, such as words and phrases (Bramsen et al., 2011; Gilbert, 2012), linguistic coordination (Danescu-Niculescu-Mizil et al., 2012), agenda control (Taylor et al., 2012), and dialog structure (Prabhakaran and Rambow, 2014).

Another area of research that has recently garnered interest within the NLP community is the modeling of author commitment in text. Initial studies in this area were done in processing hedges, uncertainty and lack of commitment, specifically focused on scientific text (Mercer et al., 2004; Di Marco et al., 2006; Farkas et al., 2010). More recently, researchers have also looked into capturing author commitment in non-scientific text, e.g., levels of factuality in newswire (Saurí and Pustejovsky, 2009), types of commitment of beliefs in a variety of genres including conversational text (Diab et al., 2009; Prabhakaran et al., 2015). These approaches are motivated from an information extraction perspective, for instance in aiding tasks such as knowledge base population.[1] However, it has not been studied whether such sophisticated author commitment analysis can go beyond what is expressed in language and reveal the underlying social contexts in which language is exchanged.

In this paper, we bring together these two lines of research; we study how power relations correlate with the levels of commitment authors express in interactions. We use the power analysis framework built by Prabhakaran and Rambow (2014) to perform this study, and measure author commitment using the committed belief tagging framework introduced by (Diab et al., 2009) that distinguishes different types of beliefs expressed in text. Our contributions are two-fold — statistical analysis of author commitment in relation with power, and enrichment of lexical features with commitment labels to aid in computational prediction of power relations. In the first part, we find that au-

[1] The BeSt track of the 2017 TAC-KBP evaluation aimed at detecting the "belief and sentiment of an entity toward another entity, relation, or event" (http://www.cs.columbia.edu/~rambow/best-eval-2017/).

Proceedings of NAACL-HLT 2018, pages 1057–1068
New Orleans, Louisiana, June 1 - 6, 2018.

thor commitment is significantly correlated with the social power relations between their participants — subordinates use more instances of non-commitment, a finding that is in line with sociolinguistics studies in this area. We also find that subordinates use significantly more reported beliefs (i.e., attributing beliefs to other agents) than superiors. This is a new finding; to our knowledge, there has not been any sociolinguistics studies investigating this aspect of interaction in relation with power. In the second part, we present novel ways of incorporating the author commitment information into lexical features that can capture important distinctions in word meanings conveyed through the belief contexts in which they occur; distinctions that are lost in a model that conflates all occurrences of a word into one unit.

We first describe the related work in computational power analysis and computational modeling of cognitive states in Section 2. In Section 3, we describe the power analysis framework we use. Section 4 formally defines the research questions we are investigating, and describes how we obtain the belief information. In Section 5, we present the statistical analysis of author commitment and power. Section 6 presents the utility of enriching lexical features with belief labels in the context of automatic power prediction. Section 7 concludes the paper and summarizes the results.

2 Related Work

The notion of belief that we use in this paper (Diab et al., 2009; Prabhakaran et al., 2015) is closely related to the notion of factuality that is captured in FactBank (Saurí and Pustejovsky, 2009). They capture three levels of factuality, certain (CT), probable (PB), and possible (PS), as well as the underspecified factuality (Uu). They also record the corresponding polarity values, and the source of the factuality assertions to distinguish between factuality assertions by the author and those by the agents/sources introduced by the author. While FactBank offers a finer granularity, they are annotated on newswire text. Hence, we use the corpus of belief annotations (Prabhakaran et al., 2015) that is obtained on online discussion forums, which is closer to our genre.

Automatic hedge/uncertainty detection is a very closely related task to belief detection. The belief tagging framework we use aims to capture the cognitive states of authors, whereas hedges are linguistic expressions that convey one of those cognitive states — non-committed beliefs. Automatic hedge/uncertainty detection has generated active research in recent years within the NLP community. Early work in this area focused on detecting speculative language in scientific text (Mercer et al., 2004; Di Marco et al., 2006; Kilicoglu and Bergler, 2008). The open evaluation as part of the CoNLL shared task in 2010 to detect uncertainty and hedging in biomedical and Wikipedia text (Farkas et al., 2010) triggered further research on this problem in the general domain (Agarwal and Yu, 2010; Morante et al., 2010; Velldal et al., 2012; Choi et al., 2012). Most of this work was aimed at formal scientific text in English. More recent work has tried to extend this work to other genres (Wei et al., 2013; Sanchez and Vogel, 2015) and languages (Velupillai, 2012; Vincze, 2014), as well as building general purpose hedge lexicons (Prokofieva and Hirschberg, 2014). In our work, we use the lexicons from (Prokofieva and Hirschberg, 2014) to capture hedges in text.

Sociolinguists have long studied the association between level of commitment and social contexts (Lakoff, 1973; O'Barr and Atkins, 1980; Hyland, 1998). A majority of this work studies gender differences in the use of hedges, triggered by the influential work by Robin Lakoff (Lakoff, 1973). She argued that women use linguistic strategies such as hedging and hesitations in order to adopt an unassertive communication style, which she terms "women's language". While many studies have found evidence to support Lakoff's theory (e.g., (Crosby and Nyquist, 1977; Preisler, 1986; Carli, 1990)), there have also been contradictory findings (e.g., (O'Barr and Atkins, 1980)) that link the difference in the use of hedges to other social factors (e.g., power). O'Barr and Atkins (1980) argue that the use of hedges is linked more to the social positions rather than gender, suggesting to rename "women's language" to "powerless language". In later work, O'Barr (1982) formalized the notion of powerless language, which formed the basis of many sociolinguistics studies on social power and communication. O'Barr (1982) analyzed courtroom interactions and identified hedges and hesitations as some of the linguistic markers of "powerless" speech. However, there has not been any computational work which has looked into how power relations relate to the level of commitment expressed in text. In this paper, we use com-

putational power analysis to perform a large scale data-oriented study on how author commitment in text reveals the underlying power relations.

There is a large body of literature in the social sciences that studies power as a social construct (e.g., (French and Raven, 1959; Dahl, 1957; Emerson, 1962; Pfeffer, 1981; Wartenberg, 1990)) and how it relates to the ways people use language in social situations (e.g., (Bales et al., 1951; Bales, 1970; O'Barr, 1982; Van Dijk, 1989; Bourdieu and Thompson, 1991; Ng and Bradac, 1993; Fairclough, 2001; Locher, 2004)). Recent years have seen growing interest in computationally analyzing and detecting power and influence from interactions. Early work in computational power analysis used social network analysis based approaches (Diesner and Carley, 2005; Shetty and Adibi, 2005; Creamer et al., 2009) or email traffic patterns (Namata et al., 2007). Using NLP to deduce social relations from online communication is a relatively new area of active research.

Bramsen et al. (2011) and Gilbert (2012) first applied NLP based techniques to predict power relations in Enron emails, approaching this task as a text classification problem using bag of words or ngram features. More recently, our work has used dialog structure features derived from deeper dialog act analysis for the task of power prediction in Enron emails (Prabhakaran and Rambow, 2014; Prabhakaran et al., 2012; Prabhakaran and Rambow, 2013). In this paper, We use the framework of (Prabhakaran and Rambow, 2014), but we analyze a novel aspect of interaction that has not been studied before — what level of commitment do the authors express in language.

There has also been work on analyzing power in other genres of interactions. Strzalkowski et al. (2010) and Taylor et al. (2012) concentrate on lower-level constructs called *Language Uses* such as agenda control to predict power in Wikipedia talk pages. Danescu-Niculescu-Mizil et al. (2012) study how social power and linguistic coordination are correlated in Wikipedia interactions as well as Supreme Court hearings. Bracewell et al. (2012) and Swayamdipta and Rambow (2012) try to identify pursuit of power in discussion forums. Biran et al. (2012) and Rosenthal (2014) study the problem of predicting influence in Wikipedia talk pages, blogs, and other online forums. Prabhakaran et al. (2013) study manifestations of power of confidence in presidential debates.

3 Power in Workplace Email: Data and Analysis Framework

The focus of our study is to investigate whether the level of commitment participants express in their contributions in an interaction is related to the power relations they have with other participants, and how it can help in the problem of predicting social power. In this section, we introduce the power analysis framework as well as the data we use in this study.

3.1 Problem

In order to model manifestations of power relations in interactions, we use our interaction analysis framework from (Prabhakaran and Rambow, 2014), where we introduced the problem of predicting organizational power relations between pairs of participants based on single email threads. The problem is formally defined as follows: given an email thread t, and a related interacting participant pair (p_1, p_2) in the thread, predict whether p_1 is the *superior* or *subordinate* of p_2. In this formulation, a *related interacting participant pair (RIPP)* is a pair of participants of the thread such that there is at least one message exchanged within the thread between them (in either direction) and that they are hierarchically related with a superior/subordinate relation.

3.2 Data

We use the same dataset we used in (Prabhakaran and Rambow, 2014), which is a version of the Enron email corpus in which the thread structure of email messages is reconstructed (Yeh and Harnly, 2006), and enriched by Agarwal et al. (2012) with gold organizational power relations, manually determined using information from Enron organizational charts. The corpus captures dominance relations between 13,724 pairs of Enron employees. As in (Prabhakaran and Rambow, 2014), we use these dominance relation tuples to obtain gold labels for the *superior* or *subordinate* relationships between pairs of participants. We use the same train-test-dev split as in (Prabhakaran and Rambow, 2014). We summarize the number of threads and related interacting participant pairs in each subset of the data in Table 1.

4 Research Hypotheses

Our first objective in this paper is to perform a large scale computational analysis of author com-

Description	Train	Dev	Test
Email threads	18079	8973	9144
# of RIPPs	7510	3578	3920

Table 1: Data Statistics. Row 1: number of threads in subsets of the corpus. Row 2: number of related interacting participant pairs in those subsets. RIPP: Related interacting participant pairs

mitment and power relations. Specifically, we want to investigate whether the commitment authors express towards their contributions in organizational interactions is correlated with the power relations they have with other participants. Sociolinguistics studies have found some evidence to suggest that lack of commitment expressed through hedges and hesitations is associated with lower power status (O'Barr, 1982). However, in our study, we go beyond hedge word lists, and analyze different cognitive belief states expressed by authors using a belief tagging framework that takes into account the syntactic contexts within which propositions are expressed.

4.1 Obtaining Belief Labels

We use the committed belief analysis framework introduced by (Diab et al., 2009; Prabhakaran et al., 2015) to model different levels of beliefs expressed in text. Specifically, in this paper, we use the 4-way belief distinction — COMMITTEDBELIEF, NONCOMMITTEDBELIEF, REPORTEDBELIEF, and NONAPPLICABLE— introduced in (Prabhakaran et al., 2015).[2] (Prabhakaran et al., 2015) presented a corpus of online discussion forums with over 850K words, annotating each propositional head in text with one of the four belief labels. The paper also presented an automatic belief tagger trained on this data, which we use to obtain belief labels in our data. We describe each belief label and our associated hypotheses below.

Committed belief (CB): the writer strongly believes that the proposition is true, and wants the reader/hearer to believe that. E.g.:

(1) a. John will **submit** the report.
b. I know that John is **capable**.

[2]We also performed analysis and experiments using an earlier 3-way belief distinction proposed by (Diab et al., 2009), which also yielded similar findings. We do not report the details of those analyses in this paper.

As discussed earlier, lack of commitment in one's writing/speech is identified as markers of powerless language. We thus hypothesize:

H. 1. *Superiors use more instances of committed belief in their messages than subordinates.*

Non-committed belief (NCB): the writer explicitly identifies the proposition as something which he or she could believe, but he or she happens not to have a strong belief in, for example by using an epistemic modal auxiliary. E.g.:

(2) a. John may **submit** the report.
b. I guess John is **capable**.

This class captures a more semantic notion of non-commitment than hedges, since the belief annotation attempts to model the underlying meaning rather than language uses, and hence captures other linguistic means of expressing non-committedness. Following (O'Barr, 1982), we formulate the below hypothesis:

H. 2. *Subordinates use more instances of non committed belief in their messages than superiors.*

Reported belief (ROB): the writer attributes belief (either committed or non-committed) to another person or group. E.g.:

(3) a. Sara says John will **submit** the report.
b. Sara thinks John may be **capable**.

Note that this label is only applied when the writer's own belief in the proposition is unclear. For instance, if the first example above was *Sara knows John will submit the report on-time*, the writer is expressing commitment toward the proposition that John will submit the report and it will be labeled as committed belief rather than reported belief. Reported belief captures instances where the writer is in effect limiting his/her commitment towards what is stated by attributing the belief to someone else. So, in line with our hypotheses for non-committed beliefs, we formulate the following hypothesis:

H. 3. *Subordinates use more instances of reported beliefs in their messages than superiors.*

Non-belief propositions (NA): – the writer expresses some other cognitive attitude toward the proposition, such as desire or intention (4a), or expressly states that he/she has no belief about the proposition (e.g., asking a question (4b)). E.g.:

(4) a. I need John to **submit** the report.
b. Will John be **capable**?

As per the above definition, requests for information (i.e., questions) and requests for actions are cases where the author is not expressing a belief about the proposition, but rather expressing the desire that some action be done. In the study correlating power with dialog act tags (Prabhakaran and Rambow, 2014), we found that superiors issue significantly more requests than subordinates. Hence, we expect the superiors to have significantly more non belief expressions in their messages, and formulate the following hypothesis:

H. 4. *Superiors use more instances of non beliefs in their messages than subordinates.*

4.2 Testing Belief Tagger Bias

NLP tools are imperfect and may produce errors, which poses a problem when using any NLP tool for sociolinguistic analysis. More than the magnitude of error, we believe that whether the error is correlated with the social variable of interest (i.e., power) is more important; e.g., is the belief-tagger more likely to find ROB false-positives in subordinates text? To test whether this is the case, we performed manual belief annotation on around 500 propositional heads in our corpus. Logistic regression test revealed that the belief-tagger is equally likely to make errors (both false-positives and false-negatives, for all four belief-labels) in sentences written by subordinates as superiors (the null hypothesis accepted at $p > 0.05$ for all eight tests).

5 Statistical Analysis

Now that we have set up the analysis framework and research hypotheses, we present the statistical analysis of how superiors and subordinates differ in their relative use of expressions of commitment.

5.1 Features

For each participant of each pair of related interacting participants in our corpus, we aggregate each of the four belief tags:

- *CBCount*: number of propositional heads tagged as Committed Belief (CB)
- *NCBCount*: number of propositional heads tagged as Non Committed Belief (NCB)
- *ROBCount*: number of propositional heads tagged as Reported Belief (ROB)
- *NACount*: number of propositional heads tagged as Non Belief (NA)

5.2 Hypotheses Testing

Our general hypothesis is that power relations do correlate with the level of commitment people express in their messages; i.e., at least one of H.1 - H.4 is true. In this analysis, each participant of the pair (p_1, p_2) is a data instance. We exclude the instances for which a feature value is undefined.[3]

In order to test whether superiors and subordinates use different types of beliefs, we used a linear regression based analysis. For each feature, we built a linear regression model predicting the feature value using power (i.e., superior vs. subordinate) as the independent variable. Since verbosity of a participant can be highly correlated with each of these feature values (we found it to be highly correlated with subordinates (Prabhakaran and Rambow, 2014)), we added token count as a control variable to the linear regression.

Our linear regression test revealed significant differences in NCB (b=-.095, t(-8.09), p<.001), ROB (b=-.083, t(-7.162), p<.001) and NA (b=.125, t(4.351), p<.001), and no significant difference in CB (b=.007, t(0.227), p=0.821). Figure 1 pictorially demonstrates these results by plotting the difference between the mean values of each commitment feature (here normalized by token count) of superiors vs. subordinates, as a percentage of mean feature value of the corresponding commitment feature for superiors. Dark bars denote statistically significant differences.

5.3 Interpretation of Findings

The results from our statistical analysis validate our original hypothesis that power relations do correlate with the level of commitment people express in their messages. This finding remains statistically significant ($p < 0.001$) even after applying the Bonferroni correction for multiple testing.

The results on NCB confirm our hypothesis that subordinates use more non-committedness in their language. Subordinates' messages contain 48% more instances of non-committed belief than superiors' messages, even after normalizing for the length of messages. This is in line with prior sociolinguistics literature suggesting that people with

[3]These are instances corresponding to participants who did not send any messages in the thread (some of the pairs in the set of related interacting participant pairs only had one-way communication) or whose messages were empty (e.g., forwarding messages).

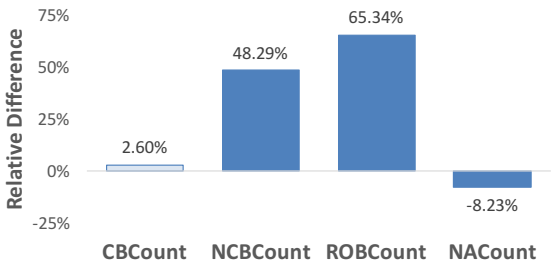


Figure 1: Relative difference (RD) between subordinates and superiors in their use of different types of commitment (counts normalized by word count of contributions). Dark bars: statistical significance at $p < 0.05$. (RD = $\frac{(Mean(Subordinates) - Mean(Superiors)) * 100}{Mean(Superiors)}$).

less power tend to use less commitment, previously measured in terms of hedges. However, in our work, we go beyond hedge dictionaries and use expressions of non-committedness that takes into account the syntactic configurations in which the words appear.

Another important finding is in terms of reported belief (ROB). Our results strongly verify the hypothesis H.3 that subordinates use significantly more reported beliefs than superiors. In fact, it obtained the largest magnitude of relative difference (65.3% more) of all features we analyzed. To our knowledge, ours is the first study that analyzed the manifestation of power in authors attributing beliefs to others. Our results are in line with the finding in (Agarwal et al., 2014) that "if many more people get mentioned to a person then that person is the boss", because as subordinates report other people's beliefs to superiors, they are also likely to mention them.

The finding that superiors use more NAs confirms our hypothesis H.4. As discussed earlier, this is expected since superiors issue more requests (as found by (Prabhakaran and Rambow, 2014)), the propositional heads of which would be tagged as NA by the belief tagger. However, our hypothesis H.1 is proven false. Being a superior or subordinate does not affect how often their messages contain CB, which suggests that power differences are manifested only in terms of lack of commitment.

6 Commitment in Power Prediction

Our next step is to explore whether we can utilize the hedge and belief labels to improve the performance of an automatic power prediction system. For this purpose, we use our POWERPREDICTOR system (Prabhakaran and Rambow, 2014) that predicts the direction of power between a pair of related interacting participants in an email thread. It uses a variety of linguistic and dialog structural features consisting of verbosity features (message count, message ratio, token count, token ratio, and tokens per message), positional features (initiator, first message position, last message position), thread structure features (number of all recipients and those in the *To* and *CC* fields of the email, reply rate, binary features denoting the adding and removing of other participants), dialog act features (request for action, request for information, providing information, and conventional), and overt displays of power, and lexical features (lemma ngrams, part-of-speech ngrams, and mixed ngrams, a version of lemma ngrams with open class words replaced with their part-of-speech tags). The feature sets are summarized in Table 2 ((Prabhakaran and Rambow, 2014) has a detailed description of these features).

Set	Description
VRB	Verbosity (e.g., message count)
PST	Positional (e.g., thread initiator?)
THR	Thread structure (e.g., reply rate)
DIA	Dialog act tagging (e.g., request count)
ODP	Overt displays of power
LEX	Lexical ngrams (lemma, POS, mixed ngrams)

Table 2: POWERPREDICTOR system: Features used

None of the features used in POWERPREDICTOR use information from the parse trees of sentences in the text However, in order to accurately obtain the belief labels, deep dependency parse based features are critical (Prabhakaran et al., 2010). We use the ClearTk wrapper for the Stanford CoreNLP pipeline to obtain the dependency parses of sentences in the email text. To ensure an unified analysis framework, we also use the Stanford CoreNLP for tokenization, part-of-speech tagging, and lemmatization steps, instead of OpenNLP. This change affects our analysis in two ways. First, the source of part-of-speech tags and word lemmas is different from what was presented in the original system, which might affect the performance of the dialog act tagger and overt display of power tagger (DIA and ODP features). Second, we had to exclude 117 threads (0.3%) from the corpus for which the Stanford CoreNLP failed to parse some sentences, resulting in the removal of 11 data points (0.2%), only one of which

was in the test set. On randomly checking, we found that they contained non-parsable text such as dumps of large tables, system logs, or unedited dumps of large legal documents.

In order to better interpret how the commitment features help in power prediction, we use a linear kernel SVM in our experiments. Linear kernel SVMs are significantly faster than higher order SVMs, and our preliminary experiments revealed the performance gain by using a higher order SVM to be only marginal. We use the best performing feature set from (Prabhakaran and Rambow, 2014) as a strong baseline for our experiments. This baseline feature set is the combination of thread structure features (THR) and lexical features (LEX). This baseline system obtained an accuracy of 68.8% in the development set.

6.1 Belief Label Enriched Lexical Features

Adding the belief label counts into the SVM directly as features will not yield much performance improvements, as signal in the aggregate counts would be minimal given the effect sizes of differences we find in Section 5. In this section, we investigate a more sophisticated way of incorporating the belief tags into the power prediction framework. Lexical features are very useful for the task of power prediction. However, it is often hard to capture deeper syntactic/semantic contexts of words and phrases using ngram features. We hypothesize that incorporating belief tags into the ngrams will enrich the representation and will help disambiguate different usages of same words/phrases. For example, let us consider two sentences: *I need the report by tomorrow* vs. *If I need the report, I will let you know*. The former is likely coming from a person who has power, whereas the latter does not give any such indication. Applying the belief tagger to these two sentences will result in *I need(CB) the report ...* and *If I need(NA) the report ....* Capturing the difference between *need(CB)* vs. *need(NA)* will help the machine learning system to make the distinction between these two usages and in turn improve the power prediction performance.

In building the ngram features, whenever we encounter a token that is assigned a belief tag, we append the belief tag to the corresponding lemma or part-of-speech tag in the ngram. We call it the *Append* version of corresponding ngram feature. We summarize the different versions of each type of ngram features below:

- *LN*: the original word lemma ngram; e.g., *i_need_the*.
- LN^{CBApnd}: word lemma ngram with appended belief tags; e.g., *i_need(CB)_the*.
- *PN*: the original part-of-speech ngram; e.g., *PRP_VB_DT*.
- PN^{CBApnd}: part-of-speech ngram with appended belief tags; e.g., *PRP_VB(CB)_DT*.
- *MN*: the original mixed ngram; e.g., *i_VB_the*.
- MN^{CBApnd}: mixed ngram with appended belief tags; e.g., *i_VB(CB)_the*.

Feature Configuration in LEXICAL	Accuracy
LN +PN +MN (*BaseLine*)	68.8
LN^{CBApnd}*+PN +MN*	**69.3**
LN +PN^{CBApnd}*+MN*	68.6
LN +PN +MN^{CBApnd}	69.0
LN^{CBApnd}*+ PN +* MN^{CBApnd}	69.2

Table 3: Power prediction results using different configurations of LEX features. (The full feature set also includes THR.)

In Table 3, we show the results obtained by incorporating the belief tags in this manner to the LEXICAL features of the original baseline feature set. The first row indicates the baseline results and the following rows show the impact of incorporating belief tags using the *Append* method. While the *Append* version of both lemma ngrams and mixed ngrams improved the results, the *Append* version of part of speech ngrams reduced the results. The combination of best performing version of each type of ngram obtained slightly lower result than using the *Append* version of word ngram alone, which posted the overall best performance of 69.3%, a significant improvement ($p<0.05$) over not using any belief information. We use the approximate randomization test (Yeh, 2000) for testing statistical significance of the improvement.

Finally, we verified that our best performing feature sets obtain similar improvements in the unseen test set. The baseline system obtained 70.2% accuracy in the test set. The best performing configuration from Table 3 significantly improved this accuracy to 70.8%. The second best performing configuration of using the *Append* version of both word and mixed ngrams obtained only a small improvement upon the baseline in the test set.

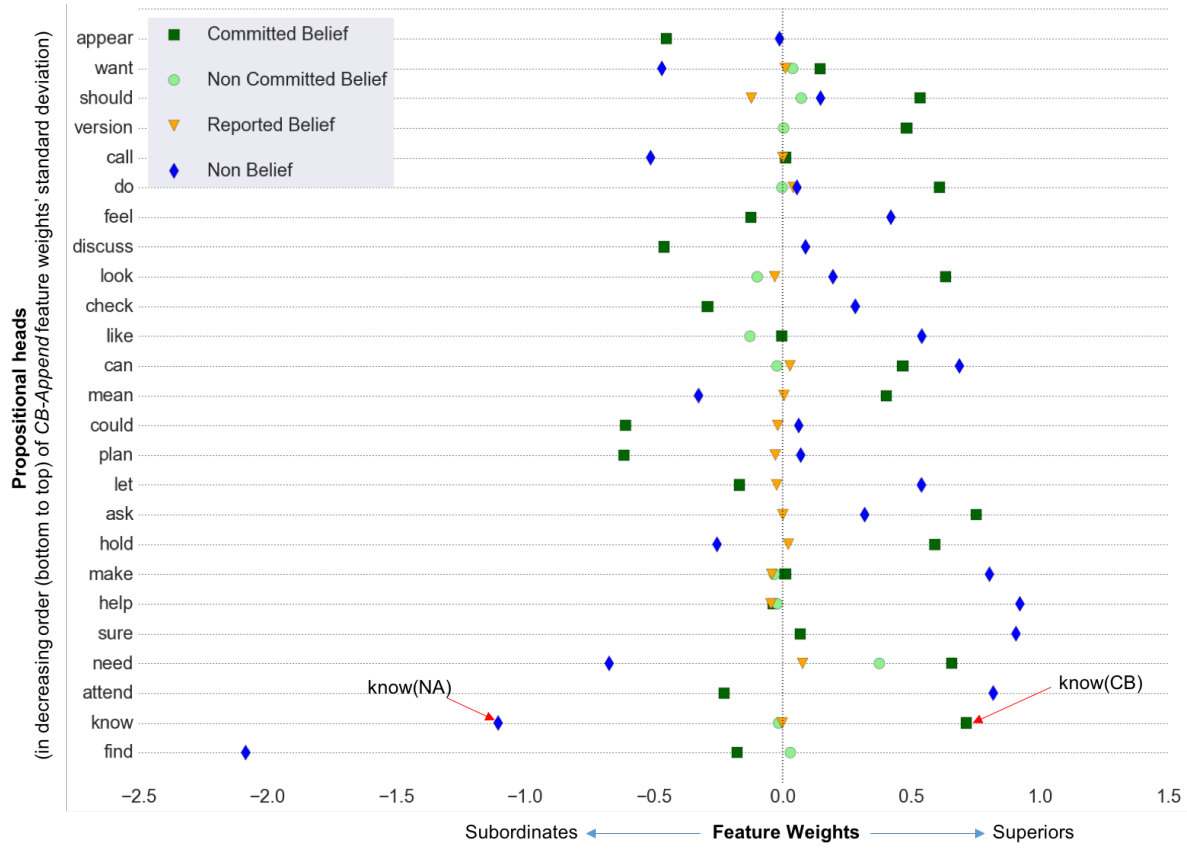


Figure 2: Feature weights of different belief appended versions of 25 propositional heads whose lemma unigrams had the highest standard deviation. Y-axis denotes the propositional heads in decreasing order of standard deviation from bottom to top. X-axis denotes the feature weights.

6.2 Word NGram Feature Analysis

We inspect the feature weights assigned to the LN^{CBApnd} version of lemma ngrams in our best performing model. Each lemma ngram that contains a propositional head (e.g., *need*) has four possible LN^{CBApnd} ngram versions: *need(CB)*, *need(NCB)*, *need(ROB)*, and *need(NA)*. For each lemma ngram, we calculate the standard deviation of weights assigned to different LN^{CBApnd} versions in the learned model as a measure of variation captured by incorporating belief tags into that ngram.[4]

Figure 2 shows the feature weights of different LN^{CBApnd} versions of twenty five propositional heads whose lemma unigrams had the highest standard deviation. The y-axis lists propositional heads arranged in the decreasing order of standard deviation from bottom to top, while the x-axis denotes the feature weights. The markers distinguish the different LN^{CBApnd} versions of each propositional head — square denotes COMMITTEDBELIEF, circle denotes NONCOMMITTEDBELIEF, triangle denotes REPORTEDBELIEF, and diamond denotes NONAPPLICABLE. The feature versions with negative weights are associated more with subordinates' messages, whereas those with positive weights are associated more with superiors' messages. Since NCB and ROB versions are rare, they rarely get high weights in the model.

We find that by incorporating belief labels into lexical features, we capture important distinctions in social meanings expressed through words that are lost in the regular lemma ngram formulation. For example, propositional heads such as *know*, *need*, *hold*, *mean* and *want* are indicators of power when they occur in CB contexts (e.g., *i need ...*), whereas their usages in NA contexts (e.g., *do you need?*, *if i need...*, etc.) are indicators of lack of power. In contrast, the CB version of *attend*, *let*, *plan*, *could*, *check*, *discuss*, and *feel* (e.g., *i will attend/check/plan ...*) are strongly associated with lack of power, while their NA versions (e.g., *can you attend/check/plan?*) are indicators of power.

[4]Not all lemma ngrams have all four versions; we calculated standard deviation using the versions present.

7 Conclusion

In this paper, we made two major contributions. First, we presented a large-scale data oriented analysis of how social power relations between participants of an interaction correlate with different types of author commitment in terms of their relative usage of hedges and different levels of beliefs — committed belief, non-committed belief, reported belief, and non-belief. We found evidence that subordinates use significantly more propositional hedges than superiors, and that superiors and subordinates use significantly different proportions of different types of beliefs in their messages. In particular, subordinates use significantly more non-committed beliefs than superiors. They also report others' beliefs more often than superiors. Second, we investigated different ways of incorporating the belief tag information into the machine learning system that automatically detects the direction of power between pairs of participants in an interaction. We devised a sophisticated way of incorporating this information into the machine learning framework by appending the heads of propositions in lexical features with corresponding belief tags, demonstrating its utility in distinguishing social meanings expressed through the different belief contexts.

This study is based on emails from a single corporation, at the beginning of the 21st century. Our findings on the correlation between author commitment and power may be reflective of the work culture that prevailed in that organization at the time when the emails were exchanged. It is important to replicate this study on emails from multiple organizations in order to assess whether these results generalize across board. It is likely that behavior patterns are affected by factors such as ethnic culture (Cox et al., 1991) of the organization, and the kinds of conversations interactants engage in (for instance, co-operative vs. competitive behavior (Hill et al., 1992)). We intend to explore this line of inquiry in future work.

Acknowledgments

This paper is partially based upon work supported by the DARPA DEFT program under a grant to Columbia University; all three co-authors were at Columbia University when portions of this work were performed. The views expressed here are those of the author(s) and do not reflect the official policy or position of the Department of Defense or the U.S. Government. We thank Dan Jurafsky and the anonymous reviewers for their helpful feedback.

References

Apoorv Agarwal, Adinoyi Omuya, Aaron Harnly, and Owen Rambow. 2012. A comprehensive gold standard for the Enron organizational hierarchy. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Jeju Island, Korea, pages 161–165. http://www.aclweb.org/anthology/P12-2032.

Apoorv Agarwal, Adinoyi Omuya, Jingwei Zhang, and Owen Rambow. 2014. Enron corporation: You're the boss if people get mentioned to you. In *Proceedings of the 2014 International Conference on Social Computing*. ACM, New York, NY, USA, SocialCom '14, pages 2:1–2:4. https://doi.org/10.1145/2639968.2640065.

Shashank Agarwal and Hong Yu. 2010. Detecting hedge cues and their scope in biomedical text with conditional random fields. *Journal of biomedical informatics* 43(6):953–961.

Robert F. Bales. 1970. *Personality and interpersonal behavior*. Holt, Rinehart, and Winston (New York).

Robert F. Bales, Fred L. Strodtbeck, Theodore M. Mills, and Mary E. Roseborough. 1951. Channels of communication in small groups. *American Sociological Review* pages 16(4), 461–468.

Or Biran, Sara Rosenthal, Jacob Andreas, Kathleen McKeown, and Owen Rambow. 2012. Detecting influencers in written online conversations. In *Proceedings of the Second Workshop on Language in Social Media*. Association for Computational Linguistics, Montréal, Canada, pages 37–45. http://www.aclweb.org/anthology/W12-2105.

Pierre Bourdieu and John B. Thompson. 1991. *Language and symbolic power*. Harvard University Press.

David B. Bracewell, Marc Tomlinson, and Hui Wang. 2012. A motif approach for identifying pursuits of power in social discourse. In *ICSC*. IEEE Computer Society, pages 1–8.

Philip Bramsen, Martha Escobar-Molano, Ami Patel, and Rafael Alonso. 2011. Extracting Social Power Relationships from Natural Language. In *ACL*. The Association for Computational Linguistics, pages 773–782.

Linda L. Carli. 1990. Gender, language, and influence. *Journal of Personality and Social Psychology* 59(5):941.

Eunsol Choi, Chenhao Tan, Lillian Lee, Cristian Danescu-Niculescu-Mizil, and Jennifer Spindel. 2012. Hedge detection as a lens on framing in the gmo debates: A position paper. In *Proceedings of the Workshop on Extra-Propositional Aspects of Meaning in Computational Linguistics*. Association for Computational Linguistics, pages 70–79.

Taylor H. Cox, Sharon A. Lobel, and Poppy Lauretta McLeod. 1991. Effects of ethnic group cultural differences on cooperative and competitive behavior on a group task. *Academy of management journal* 34(4):827–847.

Germán Creamer, Ryan Rowe, Shlomo Hershkop, and Salvatore J. Stolfo. 2009. Segmentation and automated social hierarchy detection through email network analysis. In Haizheng Zhang, Myra Spiliopoulou, Bamshad Mobasher, C. Lee Giles, Andrew Mccallum, Olfa Nasraoui, Jaideep Srivastava, and John Yen, editors, *Advances in Web Mining and Web Usage Analysis*, Springer-Verlag, Berlin, Heidelberg, pages 40–58.

Faye Crosby and Linda Nyquist. 1977. The female register: An empirical study of lakoff's hypotheses. *Language in Society* 6(03):313–322.

Robert A. Dahl. 1957. The concept of power. *Syst. Res.* 2(3):201–215. https://doi.org/10.1002/bs.3830020303.

Cristian Danescu-Niculescu-Mizil, Lillian Lee, Bo Pang, and Jon Kleinberg. 2012. Echoes of Power: Language Effects and Power Differences in Social Interaction. In *Proceedings of the 21st international conference on World Wide Web*. ACM, New York, NY, USA, WWW '12. https://doi.org/10.1145/2187836.2187931.

Chrysanne Di Marco, Frederick W. Kroon, and Robert E. Mercer. 2006. Using hedges to classify citations in scientific articles. In *Computing attitude and affect in text: theory and applications*, Springer, pages 247–263.

Mona Diab, Lori Levin, Teruko Mitamura, Owen Rambow, Vinodkumar Prabhakaran, and Weiwei Guo. 2009. Committed Belief Annotation and Tagging. In *Proceedings of the Third Linguistic Annotation Workshop*. Association for Computational Linguistics, Suntec, Singapore, pages 68–73. http://www.aclweb.org/anthology/W09-3012.

Jana Diesner and Kathleen M. Carley. 2005. Exploration of communication networks from the Enron email corpus. In *In Proc. of Workshop on Link Analysis, Counterterrorism and Security, SIAM International Conference on Data Mining 2005*. pages 21–23.

Richard M. Emerson. 1962. Power-Dependence Relations. *American Sociological Review* 27(1):31–41. https://doi.org/10.2307/2089716.

Norman Fairclough. 2001. *Language and power*. Pearson Education.

Richárd Farkas, Veronika Vincze, György Móra, János Csirik, and György Szarvas. 2010. The conll-2010 shared task: Learning to detect hedges and their scope in natural language text. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Uppsala, Sweden, pages 1–12. http://www.aclweb.org/anthology/W10-3001.

John R. French and Bertram Raven. 1959. The Bases of Social Power. In Dorwin Cartwright, editor, *Studies in Social Power*, University of Michigan Press, pages 150–167+.

Eric Gilbert. 2012. Phrases that Signal Workplace Hierarchy. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*. ACM, New York, NY, USA, CSCW '12, pages 1037–1046.

Charles W.L. Hill, Michael A. Hitt, and Robert E. Hoskisson. 1992. Cooperative versus competitive structures in related and unrelated diversified firms. *Organization Science* 3(4):501–521.

Ken Hyland. 1998. *Hedging in scientific research articles*, volume 54. John Benjamins Publishing.

Halil Kilicoglu and Sabine Bergler. 2008. Recognizing speculative language in biomedical research articles: a linguistically motivated perspective. *BMC bioinformatics* 9(Suppl 11):S10.

Robin Lakoff. 1973. Language and Woman's Place. *Language in society* 2(01):45–79.

Miriam A. Locher. 2004. *Power and politeness in action: disagreements in oral communication*. Language, power, and social process. M. de Gruyter. http://books.google.com/books?id=Aa32A4gWb8sC.

Robert E. Mercer, Chrysanne Di Marco, and Frederick W. Kroon. 2004. The frequency of hedging cues in citation contexts in scientific writing. In *Advances in artificial intelligence*, Springer, pages 75–88.

Roser Morante, Vincent Van Asch, and Walter Daelemans. 2010. Memory-based resolution of in-sentence scopes of hedge cues. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning—Shared Task*. Association for Computational Linguistics, pages 40–47.

Jr. Galileo Mark S. Namata, Lise Getoor, and Christopher P. Diehl. 2007. Inferring organizational titles in online communication. In *Proceedings of the 2006 conference on Statistical network analysis*. Springer-Verlag, Berlin, Heidelberg, ICML'06, pages 179–181. http://dl.acm.org/citation.cfm?id=1768341.1768359.

Sik Hung Ng and James J. Bradac. 1993. *Power in language: Verbal communication and social influence.*. Sage Publications, Inc.

William M. O'Barr. 1982. *Linguistic evidence: language, power, and strategy in the courtroom*. Studies on law and social control. Academic Press. http://books.google.com/books?id=bqO0PwAACAAJ.

William M. O'Barr and Bowman K. Atkins. 1980. "women's language" or "powerless language"? *Women and Language in Literature and Society* .

Jeffrey Pfeffer. 1981. *Power in organizations*. Pitman, Marshfield, MA.

Vinodkumar Prabhakaran, Tomas By, Julia Hirschberg, Owen Rambow, Samira Shaikh, Tomek Strzalkowski, Jennifer Tracey, Michael Arrigo, Rupayan Basu, Micah Clark, Adam Dalton, Mona Diab, Louise Guthrie, Anna Prokofieva, Stephanie Strassel, Gregory Werner, Janyce Wiebe, and Yorick Wilks. 2015. A New Dataset and Evaluation for Belief/Factuality. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics (*SEM 2015)*. Association for Computational Linguistics, Denver, USA.

Vinodkumar Prabhakaran, Ajita John, and Dorée D. Seligmann. 2013. Who Had the Upper Hand? Ranking Participants of Interactions Based on Their Relative Power. In *Proceedings of the IJCNLP*. Asian Federation of Natural Language Processing, Nagoya, Japan, pages 365–373. http://www.aclweb.org/anthology/I13-1042.

Vinodkumar Prabhakaran and Owen Rambow. 2013. Written Dialog and Social Power: Manifestations of Different Types of Power in Dialog Behavior. In *Proceedings of the IJCNLP*. Asian Federation of Natural Language Processing, Nagoya, Japan, pages 216–224. http://www.aclweb.org/anthology/I13-1025.

Vinodkumar Prabhakaran and Owen Rambow. 2014. Predicting power relations between participants in written dialog from a single thread. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 339–344. http://www.aclweb.org/anthology/P14-2056.

Vinodkumar Prabhakaran, Owen Rambow, and Mona Diab. 2010. Automatic Committed Belief Tagging. In *Coling 2010: Posters*. Coling 2010 Organizing Committee, Beijing, China, pages 1014–1022. http://www.aclweb.org/anthology/C10-2117.

Vinodkumar Prabhakaran, Owen Rambow, and Mona Diab. 2012. Who's (Really) the Boss? Perception of Situational Power in Written Interactions. In *24th International Conference on Computational Linguistics (COLING)*. Association for Computational Linguistics, Mumbai, India.

Bent Preisler. 1986. Linguistic sex roles in conversation. *Berlin: Mouton de Gruyter* .

Anna Prokofieva and Julia Hirschberg. 2014. Hedging and speaker commitment. In *5th International Workshop on Emotion, Social Signals, Sentiment and Linked Open Data. LREC*.

Sara Rosenthal. 2014. Detecting Influencers in Social Media Discussions. *XRDS: Crossroads, The ACM Magazine for Students* 21(1):40–45.

Liliana Mamani Sanchez and Carl Vogel. 2015. A hedging annotation scheme focused on epistemic phrases for informal language. In *Proceedings of the IWCS Workshop on Models for Modality Annotation*. Association for Computational Linguistics, London, UK.

Roser Saurí and James Pustejovsky. 2009. FactBank: a corpus annotated with event factuality. *Language Resources and Evaluation* 43:227–268. 10.1007/s10579-009-9089-9. http://dx.doi.org/10.1007/s10579-009-9089-9.

Jitesh Shetty and Jafar Adibi. 2005. Discovering important nodes through graph entropy the case of Enron email database. In *Proceedings of the 3rd international workshop on Link discovery*. ACM, New York, NY, USA, LinkKDD '05, pages 74–81. https://doi.org/http://doi.acm.org/10.1145/1134271.1134282.

Tomek Strzalkowski, George Aaron Broadwell, Jennifer Stromer-Galley, Samira Shaikh, Sarah Taylor, and Nick Webb. 2010. Modeling sociocultural phenomena in discourse. In *Proceedings of the 23rd International Conference on COLING 2010*. Coling 2010 Organizing Committee, Beijing, China. http://www.aclweb.org/anthology/C10-1117.

Swabha Swayamdipta and Owen Rambow. 2012. The Pursuit of Power and Its Manifestation in Written Dialog. *2012 IEEE Sixth International Conference on Semantic Computing* 0:22–29. https://doi.org/http://doi.ieeecomputersociety.org/10.1109/ICSC.2012.49.

Sarah M. Taylor, Ting Liu, Samira Shaikh, Tomek Strzalkowski, George Aaron Broadwell, Jennifer Stromer-Galley, Umit Boz, Xiaoai Ren, Jingsi Wu, and Feifei Zhang. 2012. Chinese and American Leadership Characteristics: Discovery and Comparison in Multi-party On-Line Dialogues. In *ICSC*. IEEE Computer Society, pages 17–21.

Teun A Van Dijk. 1989. Structures of discourse and structures of power. *Annals of the International Communication Association* 12(1):18–59.

Erik Velldal, Lilja Øvrelid, Jonathon Read, and Stephan Oepen. 2012. Speculation and negation: Rules, rankers, and the role of syntax. *Computational Linguistics* 38(2):369–410.

Sumithra Velupillai. 2012. *Shades of certainty: annotation and classification of swedish medical records*. Ph.D. thesis, Department of Computer and Systems Sciences, Stockholm University.

Veronika Vincze. 2014. Uncertainty Detection in Hungarian Texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin City University and Association for Computational Linguistics, Dublin, Ireland, pages 1844–1853. http://www.aclweb.org/anthology/C14-1174.

Thomas E. Wartenberg. 1990. *The forms of power: from domination to transformation*. Temple University Press. http://books.google.sh/books?id=yK52QgAACAAJ.

Zhongyu Wei, Junwen Chen, Wei Gao, Binyang Li, Lanjun Zhou, Yulan He, and Kam-Fai Wong. 2013. An empirical study on uncertainty identification in social media context. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 58–62. http://www.aclweb.org/anthology/P13-2011.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th conference on Computational linguistics - Volume 2*. Association for Computational Linguistics, Stroudsburg, PA, USA, COLING '00, pages 947–953. https://doi.org/http://dx.doi.org/10.3115/992730.992783.

Jen-Yuan Yeh and Aaron Harnly. 2006. Email Thread Reassembly Using Similarity Matching. In *CEAS 2006 - The Third Conference on Email and Anti-Spam, July 27-28, 2006, Mountain View, California, USA*. Mountain View, California, USA.

Comparing Automatic and Human Evaluation of Local Explanations for Text Classification

Dong Nguyen[♠◇]
[♠]The Alan Turing Institute, London
[◇]School of Informatics, University of Edinburgh, Edinburgh
dnguyen@turing.ac.uk

Abstract

Text classification models are becoming increasingly complex and opaque, however for many applications it is essential that the models are interpretable. Recently, a variety of approaches have been proposed for generating local explanations. While robust evaluations are needed to drive further progress, so far it is unclear which evaluation approaches are suitable. This paper is a first step towards more robust evaluations of local explanations. We evaluate a variety of local explanation approaches using automatic measures based on word deletion. Furthermore, we show that an evaluation using a crowdsourcing experiment correlates moderately with these automatic measures and that a variety of other factors also impact the human judgements.

1 Introduction

While the impact of machine learning is increasing rapidly in society, machine learning systems have also become increasingly complex and opaque. Classification models are usually evaluated based on prediction performance alone (e.g., by measuring the accuracy, recall, and precision) and the interpretability of these models has generally been undervalued. However, the importance of interpretable models is increasingly being recognized (Doshi-Velez and Kim, 2017; Freitas, 2014).

First, higher interpretability could lead to more effective models by revealing incompleteness in the problem formalization (Doshi-Velez and Kim, 2017), by revealing confounding factors that could lead to biased models, and by supporting error analyses or feature discovery (Aubakirova and Bansal, 2016). Second, with the increasing adoption of machine learning approaches for humanities and social science research, there is also an increasing need for systems that support exploratory analyses and theory development.

Various approaches have been explored to increase the interpretability of machine learning models (Lipton, 2016). This paper focuses on *local* explanation, which aims to explain the prediction for an individual instance (e.g., Ribeiro et al. (2016)). A study by Herlocker et al. (2000) found that providing local explanations could help improve the acceptance of movie recommendation systems. Local explanations can come in different forms. For example, Koh and Liang (2017) identify the most influential training documents for a particular prediction. The most common type of local explanation involves identifying the important parts of the input for a prediction, such as the most predictive words in a document for a text classification model.

In this paper we focus on local explanations for text classification. Below is a fragment of a movie review. The words identified by a local explanation method to explain a neural network prediction are in bold. The review is labeled with a negative sentiment, but the classifier incorrectly predicted a positive sentiment. The highlighted words help us understand why.

> steve martin is one of the **funniest** men alive. if you can take that as a **true** statement, then your disappointment at this film will equal mine. martin can be **hilarious**, **creating** some of the best laugh-out-loud **experiences** that have ever taken place in movie theaters. you won't find any of them here. [...]

Words such as *funniest* and *hilarious* were important for the prediction. Besides providing evidence *for* a predicted label, some local explanations can also provide evidence *against* a predicted label. For example, in the above example, the word *disappointment* was one of the highest ranked words against the predicted label.

Proceedings of NAACL-HLT 2018, pages 1069–1078
New Orleans, Louisiana, June 1 - 6, 2018.

Ineffective approaches could generate misleading explanations (Lipton, 2016), but evaluating local explanations is challenging. A variety of approaches has been used, including only visual inspection (Ding et al., 2017; Li et al., 2016a), intrinsic evaluation approaches such as measuring the impact of deleting the identified words on the classifier output (Arras et al., 2016), and user studies (Kulesza et al., 2015).

Contributions To further progress in this area, it is imperative to have a better understanding of how to evaluate local explanations. This paper makes the following contributions:

- *Comparison of local explanation methods for text classification.* We present an in-depth comparison between three local explanation approaches (and a random baseline) using two different automatic evaluation measures on two text classification tasks (Section 4).
- *Automatic versus human evaluation.* Automatic evaluations, such as those based on word deletions, are frequently used since they enable rapid iterations and are easy to reproduce. However, it is unclear to what extent they correspond with human-based evaluations. We show that the automatic measures correlate moderately with human judgements in a task setting and that other factors also impact human judgement. (Section 5).

2 Related Work

Research on interpretable machine learning models has so far mainly focused on computer vision systems (e.g., Simonyan et al. (2013)). Topic modeling is one of the exceptions within NLP where the interpretability of models has been important, since topic models are often valued for their interpretability and are integrated in various user interfaces (Paul, 2016). There has recently been an increasing interest in improving the interpretability of NLP models, perhaps driven by the increasing complexity of NLP models and the rise of deep learning (Manning, 2015).

Global approaches aim to provide a global view of the model. One line of work involves making the machine learning model itself more interpretable, e.g., by enforcing sparsity or imposing monotonicity constraints (Freitas, 2014). However, often there is a trade-off between accuracy and interpretability as adding constraints to the model could reduce the performance. An alternative involves extracting a more interpretable model, such as a decision tree, from a model that is less interpretable, such as a neural network (Craven, 1996). In this case, model performance is not sacrificed but it is essential that the proxy is faithful to the underlying model.

However, often a machine learning model is so complex that interpretable, trustworthy global explanations are difficult to attain. *Local* explanations aim to explain the output for an individual instance. For some models the local explanations are relatively easy to construct, e.g., displaying the word probabilities of a Naive Bayes model with respect to each label (Kulesza et al., 2015) or displaying the path of a decision tree (Lim et al., 2009). However, these models may not be easily interpretable if they make use of many features.

For many machine learning models, extracting local explanations is even less straight-forward. Proposed approaches so far include using the gradients to visualize neural networks (Aubakirova and Bansal, 2016; Li et al., 2016a; Simonyan et al., 2013), measuring the effect of removing individual words (or features) (Li et al., 2016b; Martens and Provost, 2014), decomposition approaches (Arras et al., 2016; Ding et al., 2017), and training an interpretable classifier (e.g., linear model) that approximates the neighborhood around a particular instance (Ribeiro et al., 2016).

Some approaches have only been evaluated using visual inspection (Ding et al., 2017; Li et al., 2016a). Goyal et al. (2016) identified important words for a visual question answering system and informally evaluated their approach by analyzing the distribution among PoS tags (e.g., assuming that nouns are important). However, quantitative evaluations are needed for more robust comparisons. Such evaluations have included measuring the impact of the deletion of words identified by the explanation approaches on the classification output (Arras et al., 2016, 2017), or testing whether the explanation was consistent with an underlying gold model (Ribeiro et al., 2016). These automatic evaluations are fast to carry out but act as a simplistic proxy for explanation quality. While a few user studies have been performed to evaluate explanations (e.g., Ribeiro et al. (2016)), we are not aware of work that analyzes how automatic evaluation measures compare to human-based evaluation.

3 Experimental Setup

This section describes the datasets, the classification models and the local explanation approaches used in our experiments.

3.1 Datasets

We experiment with two datasets (Table 1):

- **Twenty newsgroups (20news).** The Twenty Newsgroups dataset has been used in several studies on ML interpretability (Arras et al., 2016; Kapoor et al., 2010; Ribeiro et al., 2016). Similar to Ribeiro et al. (2016), we only distinguish between *Christianity* and *Atheism*. We use the `20news-bydate` version, and randomly reserve 20% of the training data for development.

- **Movie reviews.** Movie reviews with polarity labels (positive versus negative sentiment) from Pang and Lee (2004). We use the version from Zaidan et al. (2007). The dataset is randomly split into a train (60%), development (20%) and test (20%) set.

	Movie	20news
# training docs	1072	870
# development docs	358	209
# test docs	370	717
label distribution (pos. class)	50.00%	44.49%

Table 1: Dataset statistics

3.2 Text Classification Models

We experiment with two different models. Logistic Regression (**LR**) is implemented using Scikit-learn (Pedregosa et al., 2011) with Ridge regularisation, unigrams and a TF-IDF representation, resulting in a 0.797 accuracy on the movie dataset and a 0.921 accuracy on the 20news dataset. We experiment with a LR model, because the contributions of individual features in a LR model are known. We thus have a ground truth for feature importance to compare against for this model. We also use a feedforward neural network (**MLP**) implemented using Keras (Chollet et al., 2015), with 512 hidden units, ReLU activation, dropout (0.5, not optimized) and Adam optimization, resulting in a 0.832 accuracy on the movie dataset and a 0.939 accuracy on the 20news dataset.

3.3 Local Explanation Methods

In this paper, we focus on local explanation approaches that identify the most influential parts of the input for a particular prediction. In this paper we limit our focus to individual words for explaining the output of text classification models. Other representations, e.g., explanations using phrases or higher-level concepts are left for future work. We experiment with explanations for the *predicted* class, since in real-life settings usually no ground truth labels are available. We experiment with the following local explanation approaches:

- **Random.** A random selection of words in the document.

- **LIME** (Ribeiro et al., 2016) is a model-agnostic approach and involves training an interpretable model (in this paper, a linear model with Ridge regularisation) on samples created around the specific data point by perturbing the data. We experiment with 500–5000 samples and use the implementation provided by the authors.[1]

- **Word omission.** This approach aims to estimate the contribution of individual words by deleting them and measuring the effect, e.g., by the difference in probability (Robnik-Šikonja and Kononenko, 2008). Within NLP, variations have been proposed by Kádár et al. (2016), Li et al. (2016b) and Martens and Provost (2014). It is also similar to occlusion in the context of image classification, which involves occluding regions of the input image (Zeiler and Fergus, 2014). For **LR**, this approach corresponds to ranking words according to the regression weights (and considering the frequency in the text) and is therefore optimal. For **MLP**, we use the difference in probability for the predicted class ($\hat{y}$) when removing word w from input $\boldsymbol{x}$: $p(\hat{y}|\boldsymbol{x}) - p(\hat{y}|\boldsymbol{x}_{\setminus w})$. This approach supports explanations based on interpretable features (e.g., words) even when the underlying representation may be less interpretable. However note that in general, this omission approach might not be optimal, since it estimates the contribution of words independently. This approach is also computationally expensive, especially when many features are used.

[1] https://github.com/marcotcr/lime.

- **First derivative saliency.** This approach computes the gradient of the output with respect to the input (e.g., used in Aubakirova and Bansal (2016), Li et al. (2016a) and Simonyan et al. (2013)). The obtained estimates are often referred to as saliency values. Several variations exist, e.g., Li et al. (2016a) take the absolute value. In this paper, the raw value is taken to identify the words important for and against a certain prediction.

4 Automatic Evaluation

In this section we explore automatic evaluation of local explanations. Local explanations should exhibit high *local fidelity*, i.e. they should match the underlying model in the neighborhood of the instance (Ribeiro et al., 2016). An explanation with low local fidelity could be misleading. Because we generate explanations for the predicted class (rather than the ground truth), explanations with high local fidelity do not necessarily need to match human intuition, for example when the classifier is weak (Samek et al., 2017). Ideally, the evaluation metrics are model agnostic and do not require information that may not always be available such as probability outputs. This paper focuses on local fidelity, but other aspects might also be desired, such as sparsity (Samek et al., 2017; Ribeiro et al., 2016; Martens and Provost, 2014).

4.1 Evaluation Metrics

We measure local fidelity by deleting words in the order of their estimated importance for the prediction. Arras et al. (2016) generated explanations with the correct class as target. By deleting the identified words, accuracy increased for incorrect predictions and decreased for correct predictions. However, their approach assumes knowledge of the ground-truth labels.

We take an alternative, but similar, approach. Words are also deleted according to their estimated importance, e.g. $w_1 ... w_n$ with w_1 the word with the highest importance score, but for the *predicted class* instead. For each document, we measure the number of words that need to be deleted before the prediction switches to another class (the ***switching point***), normalized by the number of words in the document. For example, a value of 0.10 indicates that 10% of the words needed to be deleted before the prediction changed. An advantage of this approach is that ground-truth labels are not needed and that it can be applied to black-box classifiers, we only need to know the predicted class. Furthermore, the approach acts on the raw input. It requires no knowledge of the underlying feature representation (e.g., the actual features might be on the character level). We also experiment with the measure proposed by Samek et al. (2017), referred to as the area over the perturbation curve (***AOPC***):

$$AOPC = \frac{1}{K+1} \langle \sum_{k=1}^{K} f(\boldsymbol{x}) - f(\boldsymbol{x}_{\setminus \mathbf{1..k}}) \rangle_{p(x)}$$

where $f(\boldsymbol{x}_{\setminus \mathbf{1..k}})$ is the probability for the predicted class when words $1..k$ are removed and $\langle \cdot \rangle_{p(x)}$ denotes the average over the documents. This approach is also based on deleting words, but it is more fine-grained since it uses probability values rather than predicted labels. It also enables evaluating negative evidence. A drawback is that AOPC requires access to probability estimates of a classifier. In this paper, K is set to 10.

For **LR**, the exact contribution of individual features to a prediction is known and the words in the document that contributed most to the prediction can be computed directly. For this classifier, the optimal approach corresponds to the omission approach.

4.2 Results

Table 3 reports the results by measuring the effect of word deletions and reporting the average switching point. Lower values indicate that the method was better capable of identifying the words that contributed most towards the predicted class, because on average fewer words needed to be deleted to change a prediction. Table 2 shows the AOPC values with a cut-off at 10. We measure AOPC in two settings: removing positive evidence (higher values indicate a more effective explanation) and negative evidence (lower values indicate a more effective explanation).

Comparison local explanation methods As expected, LIME improves consistently when more samples are used. Furthermore, when comparing the scores of the omission approach for the **LR** model (which corresponds to the ground-truth) we observe that LIME with 5000 samples comes close to the optimal score. We use the two-tailed paired permutation test to test for significance between all methods with both evaluation measures. In al-

	20news (topic)				Movie (sentiment)			
	LR		MLP		LR		MLP	
	pos.	neg.	pos.	neg.	pos.	neg.	pos.	neg.
random	0.0116	0.0101	0.0110	0.0112	0.0073	0.0112	0.0083	0.0066
LIME-500	0.1855	-0.0301	0.1279	-0.0266	0.3168	-0.0786	0.2125	-0.0727
LIME-1000	0.2013	-0.0303	0.1350	-0.0268	0.3509	-0.0793	0.2330	-0.0738
LIME-1500	0.2067	-0.0302	0.1369	-0.0269	0.3586	-0.0794	0.2375	-0.0740
LIME-2000	0.2092	-0.0304	0.1378	-0.0269	0.3628	-0.0794	0.2394	-0.0740
LIME-5000	0.2128	-0.0303	0.1391	-0.0270	0.3693	-0.0794	0.2425	**-0.0741**
omission	**0.2342**	**-0.0307**	**0.1422**	-0.0272	**0.3724**	**-0.0795**	**0.2440**	**-0.0741**
saliency	-	-	0.1418	**-0.0273**	-	-	0.2439	**-0.0741**

Table 2: AOPC results. For each method, AOPC is used to evaluate the words identified to be supportive of the predicted class (positive evidence) and words identified to be supportive of the other class (negative evidence). For LIME, results are reported for different sample sizes.

	20news		Movie	
	LR	MLP	LR	MLP
random	0.8617	0.8880	0.6586	0.6843
LIME-500	0.4394	0.5330	0.1747	0.1973
LIME-1000	0.3098	0.4164	0.0811	0.1034
LIME-1500	0.2607	0.3566	0.0613	0.0800
LIME-2000	0.2336	0.3235	0.0547	0.0743
LIME-5000	0.1895	0.2589	0.0474	0.0664
omission	**0.1595**	0.2662	**0.0449**	0.0644
saliency	-	**0.2228**	-	**0.0639**

Table 3: The % of words that needs to be deleted to change the prediction (the switching point).

method	SP	AOPC
random	0.581	-0.168
LIME-500	0.932	-0.897
LIME-1000	0.884	-0.877
LIME-1500	0.863	-0.872
LIME-2000	0.850	-0.870
LIME-5000	0.826	-0.866
omission	0.814	-0.865
saliency	0.812	-0.865

Table 4: Spearman correlation between prediction confidence and AOPC and the switching point (SP) for the **MLP** classifier on the movie dataset.

most all cases, the differences are highly significant ($p < 0.001$), except the difference in average switching point between the omission and salience approach on the movies dataset with the **MLP** classifier (n.s.) and the difference in average switching point between the omission and LIME-5000 approach on 20news with the **MLP** classifier (n.s.). The difference in AOPC scores for evaluating negative evidence was not significant in many cases.

Metric sensitivity First, the results suggest that the values obtained depend strongly on the type of task and classifier. The explanation approaches score better on the sentiment detection task in both Tables 2 and 3. For example, fewer words need to be removed on average to change a prediction in the movie dataset (Table 3). A possible explanation is that for sentiment detection, a few words can provide strong cues for the sentiment (e.g., *terrific*), while for (fine-grained) topic detection (e.g., distinguishing between *Christianity* and *atheism*) the evidence tends to be distributed among more words. Better values are also obtained for the **LR** classifier (a linear model) than for **MLP**.

Second, as shown in Table 2, AOPC enables assessing negative evidence (i.e. the words that provide evidence for the opposite class). The obtained absolute values are much smaller compared to the values obtained for the words identified as positive evidence. This is expected, since the positive evidence in a document for the predicted class should be larger than the negative evidence.

Third, we analyze the relation between the word deletion evaluation measures and the prediction confidence of the classifiers, based on the probability of the output class. Table 4 reports the Spearman correlations for the **MLP** classifier on the movie dataset (similar trends were observed with the **LR** classifier). There is a strong correlation between the prediction confidence and the word deletion evaluation measures. The higher the prediction confidence of a classifier, the more words need to be deleted before a prediction changes (e.g., see the switching points). However, the strength of the correlations is lower for the more robust explanation methods (LIME-5000, omission and saliency).

5 Human-based Evaluation

In the previous section we evaluated the local explanation approaches using automatic measures. However, the explanations are meant to be presented to *humans*. We therefore turn to evaluating the explanations using crowdsourcing. We analyze the usefulness of the generated explanations in a task setting and analyze to what extent the automatic measures correspond to the human-based evaluations. The crowdsourcing experiments are run on CrowdFlower. Only crowdworkers from Australia, Canada, Ireland, United Kingdom and the United States and with quality levels two or three were accepted.

5.1 Forward Prediction Task

One way to evaluate an explanation is by asking humans to guess the output of a model based on the explanation and the input. Doshi-Velez and Kim (2017) refer to this as forward simulation/prediction. As mentioned by Doshi-Velez and Kim (2017), this is a simplified task. Evaluations using more specific application-oriented tasks or tailored towards specific user groups should be explored in future work. We have chosen the forward prediction task as a first step since it is a general setup that could be used to evaluate explanations for a variety of tasks and models.

In this study, crowdworkers are shown the texts (e.g., a movie review), in which the top words identified by the local explanation approaches are highlighted. Crowdworkers are then asked to guess the output of the system (e.g., a positive or negative sentiment). The crowdworkers are also asked to state their confidence on a five-point Likert scale ('*I am confident in my answer*': strongly disagree . . . strongly agree).

Note that the workers need to guess the output of the model regardless of the true label (i.e. the model may be wrong). The crowdworkers are therefore presented with documents with different prediction outcomes (true positive, true negative, false negative, and false positive). We sample up to 50 documents for each prediction outcome. A screenshot is shown in Figure 1. A quiz and test questions are used to ensure the quality of the crowdworkers. Instructions as well as the test questions included cases where the system made an incorrect prediction, so that workers understood that the task was different than standard labeling tasks. See Appendix A for more details.

We experiment with the following parameters: *methods* (random baseline, LIME with 500 and 5000 samples, word omission, saliency) and the *number of words* (10, 20). We experiment with both datasets. Due to space constraints, we only experiment with the **MLP** classifier. We collected the data in August and September 2017. Each HIT (Human Intelligence Task) was carried out by five crowdworkers. We paid $0.03 per judgement. On the 20news dataset, we collected 7,200 judgements from 406 workers (mean nr of. judgements per worker: 17.73, std.: 7.21) and on the movie dataset we collected 8,100 judgements from 445 workers (mean nr of. judgements per worker 18.20, std: 7.24).

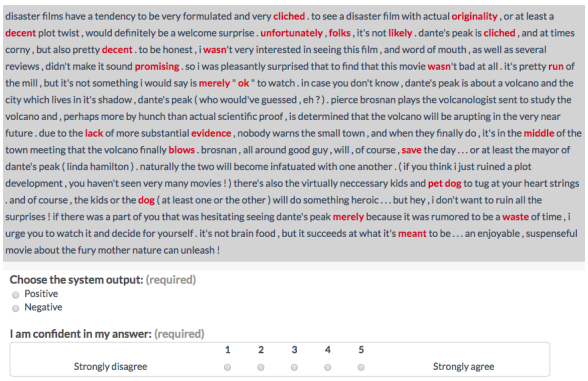


Figure 1: Screenshot of the task

Confidence Most workers chose confidence values of three or four. Table 6 reports the confidence scores by method. On the movie dataset, the trends match the intrinsic evaluations closely. The random method leads to the lowest confidence score, followed by LIME-500 and LIME-5000, and explanations from the omission and saliency approach both lead to the highest confidence scores. On the 20news dataset, the trends are less clear. We observe a small, significant negative correlation between confidence values and time spent (Spearman correlation: ρ=-0.08, p <0.0001 on the movie dataset, ρ=-0.06, p <0.0001 on 20news).

Accuracy Table 6 also reports the fraction of correct guesses per method. Random explanations lead to the lowest accuracies, followed by LIME with 500 samples. The differences between LIME-5000, omission and saliency are small and not consistent across datasets. The crowd had a higher accuracy on the movie data, except when the explanations were randomly generated.

Method	#w	TP Acc	TP Conf	TP n	TN Acc	TN Conf	TN n	FP Acc	FP Conf	FP n	FN Acc	FN Conf	FN n
Movies													
random	10	0.652	3.42	250	0.484	3.35	250	0.581	3.26	155	0.355	3.53	155
LIME-500	10	0.848	3.65	250	0.796	3.58	250	0.787	3.41	155	0.710	3.61	155
LIME-5000	10	0.900	3.73	250	0.896	3.70	250	0.852	3.43	155	0.748	3.63	155
omission	10	0.932	3.80	250	0.916	3.67	250	0.845	3.52	155	0.781	3.54	155
saliency	10	0.940	3.87	250	0.872	3.78	250	0.819	3.50	155	0.729	3.59	155
random	20	0.628	3.48	250	0.512	3.43	250	0.471	3.24	155	0.374	3.45	155
LIME-500	20	0.864	3.65	250	0.784	3.51	250	0.742	3.54	155	0.794	3.39	155
LIME-5000	20	0.880	3.76	250	0.864	3.63	250	0.787	3.77	155	0.800	3.67	155
omission	20	0.896	3.95	250	0.884	3.72	250	0.832	3.54	155	0.761	3.58	155
saliency	20	0.860	3.70	250	0.876	3.78	250	0.819	3.63	155	0.806	3.57	155
20news													
random	10	0.664	3.45	250	0.656	3.45	250	0.489	3.44	45	0.514	3.47	175
LIME-500	10	0.724	3.53	250	0.768	3.73	250	0.733	3.62	45	0.817	3.84	175
LIME-5000	10	0.740	3.52	250	0.832	3.87	250	0.556	3.29	45	0.697	3.75	175
omission	10	0.652	3.37	250	0.800	3.78	250	0.689	3.31	45	0.754	3.63	175
saliency	10	0.712	3.42	250	0.832	3.77	250	0.689	3.80	45	0.789	3.86	175
random	20	0.616	3.52	250	0.696	3.57	250	0.511	3.84	45	0.537	3.65	175
LIME-500	20	0.668	3.50	250	0.788	3.67	250	0.689	3.22	45	0.697	3.73	175
LIME-5000	20	0.720	3.52	250	0.888	3.86	250	0.667	3.36	45	0.709	3.60	175
omission	20	0.692	3.53	250	0.864	3.80	250	0.644	3.42	45	0.726	3.71	175
saliency	20	0.752	3.64	250	0.904	3.74	250	0.711	3.67	45	0.783	3.78	175

Table 5: Results forward prediction task, with the accuracy (acc), average confidence (conf) and the number of judgements (n). The results are separated according to TP (true positive), TN (true negative), FP (false positive) and FN (false negative) predictions, and the number of words shown (#w).

method	accuracy		confidence	
	20news	movie	20news	movie
random	0.616	0.522	3.520	3.402
LIME-500	0.740	0.798	3.640	3.555
LIME-5000	0.761	0.851	3.665	3.673
omission	0.744	0.868	3.615	3.694
saliency	0.790	0.851	3.691	3.701

Table 6: Confidence and accuracy results

Table 5 separates the results by the different prediction outcomes. The results suggest that false positive and false negative are the most revealing. In these cases, crowdworkers are not able to rely on their intuition and a strong explanation should convince them that the system makes a mistake. Otherwise, crowd workers might choose the label matching the document (and not necessarily the classifier output). This is especially salient in the 20news dataset, where the random approach performs better than expected on the true positives and true negatives. For example, compare the random approach with the omission approach on true positives with ten word explanations.

Our experiments also show that local explanations in the form of the most predictive words are sometimes not enough to simulate the output of a system. For example, the best accuracy on true positive instances in the 20news data is only 0.752. The movie dataset contains difficult instances as well. For example, the omission method identifies the following words in a movie review to explain a false positive prediction: '*believes*', '*become*', '*hair*', '*unhappy*', '*quentin*', '*directed*', '*runs*', '*filled*', '*fiction*', '*clint*'. Due to the composition of the training data, the system has associated words like '*quentin*' and '*clint*' with a positive sentiment. This may have confused the crowdworkers as most of them guessed incorrectly. Expanding the explanation with for example influential documents (Koh and Liang, 2017) or a visualization of the class distributions of the most influential words could make the explanations more informative.

Correlation with automatic evaluation For each explanation, we compute the fraction of workers who correctly predicted the classifier output (the 'crowd accuracy') and correlate these with the automatic measures. We expect a negative correlation with the switching points and a positive correlation with the AOPC. The correlations are moderate (Table 8). The correlations with AOPC on the movie data are the biggest on the false positives and false negatives, when workers are not able to rely on their intuition. The correlations

Noise	AOPC	TP Acc	TP Conf	TP n	TN Acc	TN Conf	TN n	FP Acc	FP Conf	FP n	FN Acc	FN Conf	FN n
0	0.2627	0.940	3.87	250	0.872	3.78	250	0.819	3.50	155	0.729	3.59	155
0.2	0.2044	0.896	3.60	250	0.780	3.67	250	0.735	3.39	155	0.735	3.58	155
0.4	0.1485	0.824	3.62	250	0.776	3.68	250	0.723	3.37	155	0.645	3.31	155
0.6	0.0851	0.800	3.40	250	0.756	3.40	250	0.710	3.63	155	0.639	3.34	155
0.8	0.0411	0.736	3.29	250	0.640	3.35	250	0.632	3.25	155	0.523	3.25	155

Table 7: Forward prediction task with noisy explanations on the movie dataset and the saliency method

	Movie SP	Movie AOPC	20news SP	20news AOPC
tp	-0.144^{**}	0.156^{***}	0.134^{**}	-0.161^{***}
fn	-0.283^{***}	0.367^{***}	-0.181^{***}	0.343^{***}
tn	-0.195^{***}	0.153^{***}	-0.203^{***}	-0.027
fp	-0.076	0.290^{***}	-0.076	0.172

Table 8: Spearman correlation between automatic measures and crowd accuracy. Significance: $^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$

Dependent variable: crowd accuracy	
Switching point	-0.365*** (0.023)
Classifier confidence	0.344*** (0.044)
Prediction outcome: fp	0.053** (0.021)
Prediction outcome: tn	0.093*** (0.020)
Prediction outcome: tp	0.132*** (0.019)
Constant	0.472*** (0.037)
R^2: 0.177 (Adj.: 0.174)	
*F Stat.: 69.255*** (df = 5; 1614)*	

Table 9: OLS results with switching points on the movie data (n = 1,620). *p<0.1; **p<0.05; ***p<0.01. Prediction outcome base level = fn.

measured on the true positives in 20news are opposite of what we expect. The 20news data is noisy and the classifier picks up on spurious features, possibly confusing the workers.

An example in the 20news data is an e-mail with the following words highlighted: '*thank*', '*mail*', '*discussions*', '*seminary*', '*before*', '*thanks*', '*question*', '*fill*', '*affected*', '*during*', '*proofs*'. The classifier was confident and the computed switchpoint was low. The e-mail comes from the atheism newsgroup, which becomes clear from reading the text. The highlighted words are all more likely to occur in the christianity newsgroup, but on their own they are not intuitive to lay people. Consequently, workers guessed incorrectly that the predicted label was atheism. Explanations that also show the negative evidence (in this case, words such as '*atheism*' and '*atheists*') and/or the word distributions across classes would likely have led to better crowd accuracy.

Dependent variable: crowd accuracy	
AOPC	0.543*** (0.042)
Classifier confidence	0.395*** (0.048)
Prediction outcome: fp	0.079*** (0.021)
Prediction outcome: tn	0.119*** (0.020)
Prediction outcome: tp	0.171*** (0.020)
Constant	0.222*** (0.046)
R^2: 0.133 (Adj.: 0.130)	
*F Stat.: 49.572*** (df = 5; 1614)*	

Table 10: OLS results with AOPC on the movie data (n = 1,620). *p<0.1; **p<0.05; ***p<0.01. Prediction outcome base level = fn.

As shown in section 4, the automatic measures correlate strongly with the prediction confidence of the classifier. More words need to be removed before a prediction changes (i.e. a higher switching point) when the classifier is more confident. However, we also find that higher classifier confidence leads to higher crowd accuracies (e.g., ρ = 0.236, $p < 0.001$ on the 20news dataset). We therefore fit an Ordinary Least Squares (OLS) model to control for these different factors (Table 9), with crowd accuracy as the dependent variable. A higher switching point significantly leads to a lower accuracy. However, classifier confidence and prediction outcome also significantly impact the accuracy. Similar trends are observed for the AOPC measure (Table 10). We also find that the automatic evaluation measures significantly impact crowd accuracy on the 20news dataset, but the patterns are less strong.

Noise In our final experiment we analyze the effect of noise. We focus on explanations based on saliency scores on the movie dataset. We experiment with introducing noise to the top ten words (Table 7) and we collect additional judgements. A noise level of 0.2 indicates that two out of the top ten words are randomly replaced by other words. The results show that with increasing the noise, as expected, both the performance and average AOPC score decrease.

6 Conclusion

There has been an increasing interest in improving the interpretability of machine learning systems, but evaluating the quality of explanations has been challenging. This paper focused on evaluating local explanations for text classification. Local explanations were generated by identifying important words in a document for a prediction. We compared automatic evaluation approaches, based on measuring the effect of word deletions, with human-based evaluations. Explanations generated using word omissions and first derivatives both performed well. LIME (Ribeiro et al., 2016) performed close to these methods when using enough samples. Our analyses furthermore showed that the evaluation numbers depend on the task/dataset and the confidence of the classifiers.

Next, crowd workers were asked to predict the output of the classifiers based on the generated explanations. We found moderate, but significant, correlations between the automatic measures and crowd accuracy. In addition, the human judgements were impacted by the confidence of the classifier and the type of prediction outcome (e.g., a false negative versus a true positive). Our results also suggest that only highlighting words is sometimes not enough. An explanation can highlight the most important parts of an input and score well on automatic measures, but if the explanation is not intuitive (for example due to biases in the data), humans are still not able to predict the output.

For the classification tasks in this paper (topic classification and sentiment detection) individual words are often predictive. As a result, local explanation approaches that select words independently worked well. However, we expect that for tasks where individual words are not predictive, the current evaluation methods and local explanation approaches may not be sufficient. Furthermore, in future work more fine-grained visualizations (e.g., Handler et al. (2016)) could be explored.

Acknowledgements

This work was supported by The Alan Turing Institute under the EPSRC grant EP/N510129/1. The author is supported with an Alan Turing Institute Fellowship (TU/A/000006). This work was supported with seed funding award SF023.

References

Leila Arras, Franziska Horn, Grégoire Montavon, Klaus-Robert Muller, and Wojciech Samek. 2016. Explaining predictions of non-linear classifiers in NLP. In *Proceedings of the 1st Workshop on Representation Learning for NLP*. pages 1–7.

Leila Arras, Franziska Horn, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2017. "What is relevant in a text document?": An interpretable machine learning approach. *PLOS ONE* 12(8):e0181142.

Malika Aubakirova and Mohit Bansal. 2016. Interpreting neural networks to improve politeness comprehension. In *Proceedings of EMNLP 2016*. pages 2035–2041.

Chollet et al. 2015. Keras. `https://github.com/fchollet/keras`.

Mark W. Craven. 1996. *Extracting comprehensible models from trained neural networks*. Ph.D. thesis, University of Wisconsin–Madison.

Yanzhuo Ding, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Visualizing and understanding neural machine translation. In *Proceedings of ACL 2017*. pages 1150–1159.

Finale Doshi-Velez and Been Kim. 2017. Towards a rigorous science of interpretable machine learning. In *arXiv preprint arXiv:1702.08608*.

Alex A. Freitas. 2014. Comprehensible classification models: A position paper. *SIGKDD Explorations Newsletter* 15(1):1–10.

Yash Goyal, Akrit Mohapatra, Devi Parikh, and Dhruv Batra. 2016. Towards transparent AI systems: Interpreting visual question answering models. In *International Conference on Machine Learning (ICML) Workshop on Visualization for Deep Learning*.

Abram Handler, Su Lin Blodgett, and Brendan O'Connor. 2016. Visualizing textual models with in-text and word-as-pixel highlighting. In *Proceedings of the 2016 Workshop on Human Interpretability in Machine Learning*.

Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. 2000. Explaining collaborative filtering recommendations. In *Proceedings of CSCW '00*. pages 241–250.

Ákos Kádár, Grzegorz Chrupala, and Afra Alishahi. 2016. Representation of linguistic form and function in recurrent neural networks. *CoRR* abs/1602.08952.

Ashish Kapoor, Bongshin Lee, Desney Tan, and Eric Horvitz. 2010. Interactive optimization for steering machine classification. In *Proceedings of CHI '10*. pages 1343–1352.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *Proceedings of ICML 2017*. pages 1885–1894.

Todd Kulesza, Margaret Burnett, Weng-Keen Wong, and Simone Stumpf. 2015. Principles of explanatory debugging to personalize interactive machine learning. In *Proceedings of IUI '15*. pages 126–137.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016a. Visualizing and understanding neural models in NLP. In *Proceedings of NAACL 2016*. pages 681–691.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016b. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220* .

Brian Y. Lim, Anind K. Dey, and Daniel Avrahami. 2009. Why and why not explanations improve the intelligibility of context-aware intelligent systems. In *Proceedings of CHI '09*. pages 2119–2128.

Zachary C. Lipton. 2016. The mythos of model interpretability. In *Proceedings of the 2016 ICML Workshop on Human Interpretability in Machine Learning (WHI 2016)*. pages 96–100.

Christopher D. Manning. 2015. Computational linguistics and deep learning. *Computational Linguistics* 41(4):701–707.

David Martens and Foster Provost. 2014. Explaining data-driven document classifications. *MIS Quarterly* 38(1):73–100.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity. In *Proceedings of ACL*. pages 271–278.

Paul. 2016. Interpretable machine learning: Lessons from topic modeling. In *Proceedings of the CHI Workshop on Human-Centered Machine Learning*.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of KDD '16*. pages 1135–1144.

Marko Robnik-Šikonja and Igor Kononenko. 2008. Explaining classifications for individual instances. *IEEE Transactions on Knowledge and Data Engineering* 20(5):589–600.

Wojciech Samek, Alexander Binder, Gregoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. 2017. Evaluating the visualization of what a deep neural network has learned. *IEEE Transactions on Neural Networks and Learning Systems* 28(11):2660 – 2673.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR* abs/1312.6034.

Omar Zaidan, Jason Eisner, and Christine Piatko. 2007. Using "annotator rationales" to improve machine learning for text categorization. In *Proceedings of NAACL 2007*. pages 260–267.

Matthew D. Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *ECCV 2014*. pages 818–833.

A Appendix: Crowdsourcing

Test questions were manually selected and were cases for which there should be no doubt about the correct answer (e.g., a simple movie review with only words such as '*brilliant*', '*terrific*', etc. highlighted). Thus, these are questions where workers would only fail if they did not pay attention or if they did not understand the task. Explanations were provided for most test questions and were shown after an answer was submitted. The test questions contained instances with different prediction outcomes (e.g. false positives and true positives) to make the task clear. To make sure that the test questions did not overlap with the actual HITs (which were generated to explain the predictions of the **`MLP`**), the test questions were explanations generated for the **`LR`** classifier.

A quiz with test questions was provided to the crowdworkers when starting the task. If the workers performed poorly on the quiz, they were not allowed to continue with the task. Throughout the task, test questions were entered in between the actual HITs (one out of five presented HITs was a test question), to monitor the quality and to flag crowdworkers who performed poorly. We closely monitored the responses to the test questions and in the pilot phase we did remove a few that turned out not be suitable. In the final task, workers performed overall very well on the test questions.

The task was consistently rated positive by the crowdworkers. The task was divided into several batches and the overall rating was consistently above 4.5 (out of 5). The payment rating was consistently above 4. The tasks explicitly mentioned that the results will be used for scientific research ('*By participating you agree that these results will be used for scientific research.*').

Deep Temporal-Recurrent-Replicated-Softmax for Topical Trends over Time

Pankaj Gupta[1,2], Subburam Rajaram[1], Hinrich Schütze[2], Bernt Andrassy[1]
[1]Corporate Technology, Machine-Intelligence (MIC-DE), Siemens AG Munich, Germany
[2]CIS, University of Munich (LMU) Munich, Germany
{pankaj.gupta, subburam.rajaram, bernt.andrassy}@siemens.com
pankaj.gupta@campus.lmu.de | inquiries@cislmu.org

Abstract

Dynamic topic modeling facilitates the identification of topical trends over time in temporal collections of unstructured documents. We introduce a novel unsupervised neural dynamic topic model named as Recurrent Neural Network-Replicated Softmax Model (RNN-RSM), where the discovered topics at each time influence the topic discovery in the subsequent time steps. We account for the temporal ordering of documents by explicitly modeling a joint distribution of latent topical dependencies over time, using distributional estimators with temporal recurrent connections. Applying RNN-RSM to 19 years of articles on NLP research, we demonstrate that compared to state-of-the art topic models, RNN-RSM shows better generalization, topic interpretation, evolution and trends. We also introduce a metric (named as SPAN) to quantify the capability of dynamic topic model to capture word evolution in topics over time.

1 Introduction

Topic Detection and Tracking (Allan et al., 1998) is an important area of natural language processing to find topically related ideas that evolve over time in a sequence of text collections and exhibit temporal relationships. The temporal aspects of these collections can present valuable insight into the topical structure of the collections and can be quantified by modeling the dynamics of the underlying topics discovered over time.

Problem Statement: We aim to generate temporal topical trends or automatic overview timelines of topics for a time sequence collection of documents. This involves the following three tasks in dynamic topic analysis: **(1)** *Topic Structure Detection* (TSD): Identifying main topics in the document collection. **(2)** *Topic Evolution Detection* (TED): Detecting the emergence of a new topic and recognizing how it grows or decays over time (Allan, 2002). **(3)** *Temporal Topic Characterization* (TTC): Identifying the characteristics for each of the main topics in order to track the words' usage (*keyword trends*) for a topic over time i.e. *topical trend analysis for word evolution* (Fig 1, Left).

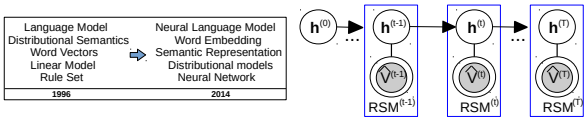


Figure 1: (Left): Word Usage over time for Topic (*Word Representation*) in scholarly articles. (Right): RSM-based dynamic topic model with explicit temporal topic dependence

Probabilistic static topic models, such as Latent Dirichlet Allocation (LDA) (Blei et al., 2003) and its variants (Wang and McCallum, 2006; Hall et al., 2008; Gollapalli and Li, 2015) have been investigated to examine the emergence of topics from historical documents. Another variant known as Replicated Softmax (RSM) (Hinton and Salakhutdinov, 2009) has demonstrated better generalization in log-probability and retrieval, compared to LDA. Prior works (Iwata et al., 2010; Pruteanu-Malinici et al., 2010; Saha and Sindhwani, 2012; Schein et al., 2016) have investigated Bayesian modeling of topics in time-stamped documents. Particularly, Blei and Lafferty (2006) developed a LDA based dynamic topic model (DTM) to capture the evolution of topics in a time sequence collection of documents; however they do not capture explicitly the topic popularity and usage of specific terms over time. We propose a family of probabilistic time series models with distributional estimators to explicitly model the dynamics of the underlying topics, introducing temporal latent topic dependencies (Fig 1, Right).

To model temporal dependencies in high dimen-

Proceedings of NAACL-HLT 2018, pages 1079–1089
New Orleans, Louisiana, June 1 - 6, 2018.

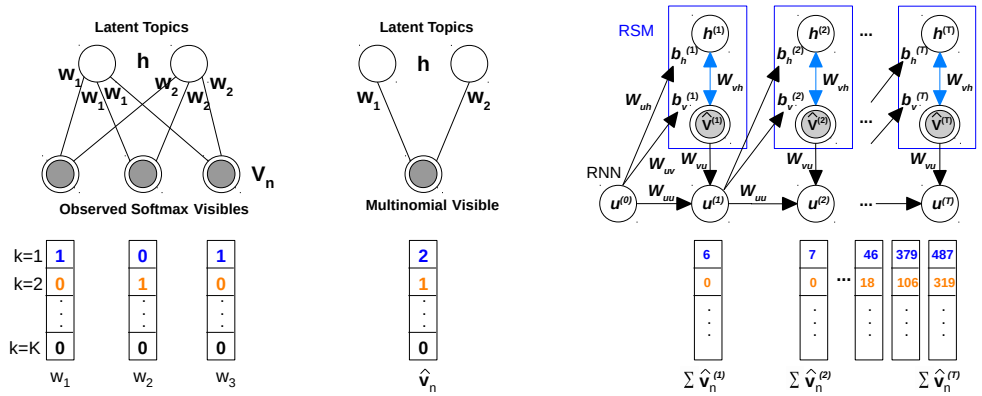


Figure 2: (Left): RSM for a document $\mathbf{V}_n$ of D_n=3 words (w). The bottom layer represents the softmax visible units, that share the same set of weights connected to binary hidden units $\mathbf{h}$. (Middle): Interpretation of RSM in which D_n softmax units with identical weights are replaced by a single multinomial unit, sampled D_n times. (Right): Graphical structure of 2-layered **RNN-RSM**, unfolded in time. Single and double headed arrows represent deterministic and stochastic-symmetric connections, respectively. $\widehat{\mathbf{V}}^{(t)}$ and $\mathbf{h}^{(t)}$ are binary visible and hidden layers of RSM for a document collection at time, t. $\mathbf{u}$ is RNN hidden layer. k: dictionary index for a word w

sional sequences, such as polyphonic music, the temporal stack of RBMs (Smolensky, 1986; Hinton, 2002) has been investigated to model complex distributions. The Temporal RBM (Taylor et al., 2007; Sutskever and Hinton, 2007), Recurrent Temporal RBM (RTRBM) (Sutskever et al., 2009) and RNN-RBM (Boulanger-Lewandowski et al., 2012) show success in modeling the temporal dependencies in such symbolic sequences. In addition, RNNs (Gupta et al., 2015a; Vu et al., 2016a,b; Gupta et al., 2016) have been recognized for sentence modeling in natural language tasks. We aspire to build neural dynamic topic model called RNN-RSM to model document collections over time and learn temporal topic correlations.

We consider RSM for TSD and introduce the explicit latent topical dependencies for TED and TTC tasks. Fig 1 illustrates our *motivation*, where temporal ordering in document collection $\widehat{\mathbf{V}}^{(t)}$ at each time step t, is modeled by conditioning the latent topic $\mathbf{h}^{(t)}$ on the sequence history of latent topics $\mathbf{h}^{(0)}$, ..., $\mathbf{h}^{(t-1)}$, accumulated with temporal lag. Each RSM discovers latent topics, where the introduction of a bias term in each RSM via the time-feedback latent topic dependencies enables to explicitly model topic evolution and specific topic term usage over time. The temporal connections and RSM biases allow to convey topical information and model relation among the words, in order to deeply analyze the dynamics of the underlying topics. We demonstrate the applicability of proposed **RNN-RSM** by analyzing 19 years of scientific articles from NLP research.

The *contributions* in this work are:
(1) Introduce an unsupervised neural dynamic topic model based on recurrent neural network and RSMs, named as RNN-RSM to explicitly model discovered latent topics (evolution) and word relations (topic characterization) over time.
(2) Demonstrate better generalization (log-probability and time stamp prediction), topic interpretation (coherence), evolution and characterization, compared to the state-of-the-art.
(3) It is the first work in dynamic topic modeling using undirected stochastic graphical models and deterministic recurrent neural network to model collections of different-sized documents over time, within the generative and neural network framework. The code and data are available at `https://github.com/pgcool/RNN-RSM`.

2 The RNN-RSM model

RSM (Fig 2, Left) models are a family of different-sized Restricted Boltzmann Machines (RBMs) (Gehler et al., 2006; Xing et al., 2005; Gupta et al., 2015b,c) that models *word counts* by sharing the same parameters with multinomial distribution over the observable i.e. it can be interpreted as a single multinomial unit (Fig 2, Middle) sampled as many times as the document size. This facilitates in dealing with the documents of different lengths.

The proposed RNN-RSM model (Fig 2, Right) is a sequence of conditional RSMs[1] such that at any time step t, the RSM's bias parameters $\mathbf{b_v}^{(t)}$

[1]Notations: $\widehat{\mathbf{U}}$=$\{\mathbf{U}_n\}_{n=1}^{N}$; **U**:2D-Matrix; **l**:vector; **U/l**:Upper/lower-case; Scalars in unbold

and $\mathbf{b_h}^{(t)}$ depend on the output of a deterministic RNN with hidden layer $\mathbf{u}^{(t-1)}$ in the previous time step, $t-1$. Similar to RNN-RBM (Boulanger-Lewandowski et al., 2012), we constrain RNN hidden units ($\mathbf{u}^{(t)}$) to convey temporal information, while RSM hidden units ($\mathbf{h}^{(t)}$) to model conditional distributions. Therefore, parameters ($\mathbf{b_v}^{(t)}$, $\mathbf{b_h}^{(t)}$) are time-dependent on the sequence history at time t (via a series of conditional RSMs) denoted by $\Theta^{(t)} \equiv \{\widehat{\mathbf{V}}^{(\tau)}, \mathbf{u}^{(\tau)} | \tau < t\}$, that captures temporal dependencies. The RNN-RSM is defined by its joint probability distribution:

$$P(\widehat{\mathfrak{V}}, \mathbf{H}) = P(\{\widehat{\mathbf{V}}^{(t)}, \mathbf{h}^{(t)}\}_{t=1}^{T}) = \prod_{t=1}^{T} P(\widehat{\mathbf{V}}^{(t)}, \mathbf{h}^{(t)} | \Theta^{(t)})$$

where $\widehat{\mathfrak{V}} = [\widehat{\mathbf{V}}^{(1)}, ...\widehat{\mathbf{V}}^{(T)}]$ and $\mathbf{H} = [\mathbf{h}^{(1)}, ...\mathbf{h}^{(T)}]$. Each $\mathbf{h}^{(t)} \in \{0,1\}^F$ be a binary stochastic hidden topic vector with size F and $\widehat{\mathbf{V}}^{(t)} = \{\mathbf{V}_n^{(t)}\}_{n=1}^{N^{(t)}}$ be a collection of N documents at time step t. Let $\mathbf{V}_n^{(t)}$ be a $K \times D_n^{(t)}$ observed binary matrix of the n^{th} document in the collection where, $D_n^{(t)}$ is the document size and K is the dictionary size over all the time steps. The conditional distribution (for each unit in hidden or visible) in each RSM at time step, is given by softmax and logistic functions:

$$P(v_{n,i}^{k,(t)} = 1 | \mathbf{h}_n^{(t)}) = \frac{\exp(b_{v,i}{}^{k,(t)} + \sum_{j=1}^{F} h_{n,j}^{(t)} W_{ij}^{k})}{\sum_{q=1}^{K} \exp(b_{v,i}{}^{q,(t)} + \sum_{j=1}^{F} h_{n,j}^{(t)} W_{ij}^{q})}$$

$$P(h_{n,j}^{(t)} = 1 | \mathbf{V}_n^{(t)}) = \sigma(b_{h,j}^{(t)} + \sum_{i=1}^{D_n^{(t)}} \sum_{k=1}^{K} v_{n,i}^{k,(t)} W_{ij}^{k})$$

where $P(v_{n,i}^{k,(t)} = 1 | \mathbf{h}_n^{(t)})$ and $P(h_{n,j}^{(t)} = 1 | \mathbf{V}_n^{(t)})$ are conditional distributions for i^{th} visible $v_{n,i}$ and j^{th} hidden unit $h_{n,j}$ for the n^{th} document at t. W_{ij}^k is a symmetric interaction term between i that takes on value k and j. $v_n^{k,(t)}$ is sampled $D_n^{(t)}$ times with identical weights connected to binary hidden units, resulting in multinomial visibles, therefore the name *Replicated Softmax*. The conditionals across layers are factorized as: $P(\mathbf{V}_n^{(t)} | \mathbf{h}_n^{(t)}) = \prod_{i=1}^{D_n^{(t)}} P(\mathbf{v}_{n,i}^{(t)} | \mathbf{h}_n^{(t)})$; $P(\mathbf{h}_n^{(t)} | \mathbf{V}_n^{(t)}) = \prod_j P(h_{n,j}^{(t)} | \mathbf{V}_n^{(t)})$.

Since biases of RSM depend on the output of RNN at previous time steps, that allows to propagate the estimated gradient at each RSM backward through time (BPTT). The *RSM biases* and RNN hidden state $\mathbf{u}^{(t)}$ at each time step t are given by-

$$\begin{aligned} \mathbf{b_v}^{(t)} &= \mathbf{b_v} + \mathbf{W_{uv}} \mathbf{u}^{(t-1)} \\ \mathbf{b_h}^{(t)} &= \mathbf{b_h} + \mathbf{W_{uh}} \mathbf{u}^{(t-1)} \end{aligned} \quad (1)$$

$$\mathbf{u}^{(t)} = \tanh(\mathbf{b_u} + \mathbf{W_{uu}} \mathbf{u}^{(t-1)} + \mathbf{W_{vu}} \sum_{n=1}^{N^{(t)}} \hat{\mathbf{v}}_n^{(t)}) \quad (2)$$

Algorithm 1 *Training RNN-RSM with BPTT*

Input: *Observed visibles*, $\widehat{\mathfrak{V}} = \{\widehat{\mathbf{V}}^{(0)}, \widehat{\mathbf{V}}^{(1)}, ..., \widehat{\mathbf{V}}^{(t)}, ..., \widehat{\mathbf{V}}^{(T)}\}$
RNN-RSM Parameters: $\theta = \{\mathbf{W_{uh}}, \mathbf{W_{vh}}, \mathbf{W_{uv}}, \mathbf{W_{vu}}, \mathbf{W_{uu}}, \mathbf{b_v}, \mathbf{b_u}, \mathbf{b_h}, \mathbf{b_v}^{(t)}, \mathbf{b_h}^{(t)}, \mathbf{u}^{(0)}\}$

1: Propagate $\mathbf{u}^{(t)}$ in RNN portion of the graph using eq 2.
2: Compute $\mathbf{b_v}^{(t)}$ and $\mathbf{b_h}^{(t)}$ using eq 1.
3: Generate negatives $\mathbf{V}^{(t)*}$ using k-step Gibbs sampling.
4: Estimate the gradient of the cost C w.r.t. parameters of RSM $\mathbf{W_{vh}}$, $\mathbf{b_v}^{(t)}$ and $\mathbf{b_h}^{(t)}$ using eq 5.
5: Compute gradients (eq 6) w.r.t. RNN connections ($\mathbf{W_{uh}}$, $\mathbf{W_{uv}}$, $\mathbf{W_{uu}}$, $\mathbf{W_{vu}}$, $\mathbf{u}^0$) and biases ($\mathbf{b_v}$, $\mathbf{b_h}$, $\mathbf{b_u}$).
6: **Goto** *step 1* until stopping_criteria (early stopping or maximum iterations reached)

where $\mathbf{W_{uv}}$, $\mathbf{W_{uh}}$ and $\mathbf{W_{vu}}$ are weights connecting RNN and RSM portions (Figure 2). $\mathbf{b_u}$ is the bias of $\mathbf{u}$ and $\mathbf{W_{uu}}$ is the weight between RNN hidden units. $\hat{\mathbf{v}}_n^{(t)}$ is a vector of $\hat{v}_n^k$ (denotes the count for the k^{th} word in n^{th} document). $\sum_{n=1}^{N^{(t)}} \hat{\mathbf{v}}_n^{(t)}$ refers to the sum of observed vectors across documents at time step t where each document is represented as-

$$\hat{\mathbf{v}}_n^{(t)} = [\{\hat{v}_n^{k,(t)}\}_{k=1}^{K}] \text{ and } \hat{v}_n^{k,(t)} = \sum_{i=1}^{D_n^{(t)}} v_{n,i}^{k,(t)} \quad (3)$$

where $v_{n,i}^{k,(t)}$=1 if visible unit i takes on k^{th} value.

In each RSM, a separate RBM is created for each document in the collection at time step t with $D_n^{(t)}$ softmax units, where $D_n^{(t)}$ is the count of words in the n^{th} document. Consider a document of $D_n^{(t)}$ words, the *energy* of the state $\{\mathbf{V}_n^{(t)}, \mathbf{h}_n^{(t)}\}$ at time step, t is given by-

$$\begin{aligned} E(\mathbf{V}_n^{(t)}, \mathbf{h}_n^{(t)}) = &- \sum_{j=1}^{F} \sum_{k=1}^{K} h_{n,j}^{(t)} W_j^k \hat{v}_n^{k,(t)} \\ &- \sum_{k=1}^{\mathrm{K}} \hat{v}_n^{k,(t)} b_v^k - D_n^{(t)} \sum_{j=1}^{F} b_{h,j} h_{n,j}^{(t)} \end{aligned}$$

Observe that the bias terms on hidden units are scaled up by document length to allow hidden units to stabilize when dealing with different-sized documents. The corresponding energy-probability relation in the energy-based model is-

$$P(\mathbf{V}_n^{(t)}) = \frac{1}{Z_n^{(t)}} \sum_{\mathbf{h}_n^{(t)}} \exp(-E(\mathbf{V}_n^{(t)}, \mathbf{h}_n^{(t)})) \quad (4)$$

where $Z_n^{(t)} = \sum_{\mathbf{V}_n^{(t)}} \sum_{\mathbf{h}_n^{(t)}} \exp(-E(\mathbf{V}_n^{(t)}, \mathbf{h}_n^{(t)}))$ is the normalization constant. The lower bound on the log likelihood of the data takes the form:

$$\begin{aligned} \ln P(\mathbf{V}_n^{(t)}) &\geq \sum_{\mathbf{h}^{(t)}} Q(\mathbf{h}_n^{(t)} | \mathbf{V}_n^{(t)}) \ln P(\mathbf{V}_n^{(t)}, \mathbf{h}_n^{(t)}) + H(Q) \\ &= \ln P(\mathbf{V}_n^{(t)}) - KL[Q(\mathbf{h}_n^{(t)} | \mathbf{V}_n^{(t)}) || P(\mathbf{h}_n^{(t)} | \mathbf{V}_n^{(t)})] \end{aligned}$$

Year	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	Total
ACL	58	73	250	83	79	70	177	112	134	134	307	204	214	243	270	349	227	398	331	3713
EMNLP	15	24	15	36	29	21	42	29	58	28	75	132	115	164	125	149	140	206	228	1756
ACL+EMNLP	73	97	265	119	108	91	219	141	192	162	382	336	329	407	395	498	367	604	559	**5469**

Table 1: Number of papers from ACL and EMNLP conferences over the years

where $H(\cdot)$ is the entropy and Q is the approximating posterior. Similar to Deep Belief Networks (Hinton et al., 2006), adding an extra layer improves lower bound on the log probability of data, we introduce the extra layer via RSM biases that propagates the prior via RNN connections. The dependence analogy follows-

$$E(\mathbf{V}_n^{(t)}, \mathbf{h}_n^{(t)}) \propto \tfrac{1}{\mathbf{b_v}^{(t)}} \text{ and } E(\mathbf{V}_n^{(t)}, \mathbf{h}_n^{(t)}) \propto \tfrac{1}{\mathbf{b_h}^{(t)}}$$

$$\ln P(\mathbf{V}_n^{(t)}) \propto \tfrac{1}{E(\mathbf{V}_n^{(t)}, \mathbf{h}_n^{(t)})}; \ln P(\widehat{\mathbf{V}}_n^{(t)}) \propto \ln P(\{\widehat{\mathbf{V}}_n^{\tau}\}_{\tau<t})$$

Observe that the prior is seen as the deterministic hidden representation of latent topics and injected into each hidden state of RSMs, that enables the likelihood of the data to model complex temporal densities i.e. heteroscedasticity in document collections ($\widehat{\mathfrak{V}}$) and temporal topics ($\mathbf{H}$).

Gradient Approximations: The *cost* in RNN-RSM is: $C = \sum_{t=1}^{T} C_t \equiv \sum_{t=1}^{T} -\ln P(\widehat{\mathbf{V}}^{(t)})$

Due to intractable Z, the gradient of cost at time step t w.r.t. (with respect to) RSM parameters are approximated by k-step Contrastive Divergence (CD) (Hinton, 2002). The gradient of the negative log-likelihood of a document collection $\{\mathbf{V}_n^{(t)}\}_{n=1}^{N^{(t)}}$ w.r.t. RSM parameter $\mathbf{W_{vh}}$,

$$\begin{aligned}
&\frac{1}{N^{(t)}} \sum_{n=1}^{N^{(t)}} \frac{\partial(-\ln P(\mathbf{V}_n^{(t)}))}{\partial \mathbf{W_{vh}}} \\
&= \frac{1}{N^{(t)}} \sum_{n=1}^{N^{(t)}} \frac{\partial \mathfrak{F}(\mathbf{V}_n^{(t)})}{\partial \mathbf{W_{vh}}} - \frac{\partial(-\ln Z_n^{(t)})}{\partial \mathbf{W_{vh}}} \\
&= \underbrace{\mathsf{E}_{P_{data}}[\frac{\partial \mathfrak{F}(\mathbf{V}_n^{(t)})}{\partial \mathbf{W_{vh}}}]}_{\text{data-dependent expectation}} - \underbrace{\mathsf{E}_{P_{model}}[\frac{\partial \mathfrak{F}(\mathbf{V}_n^{(t)})}{\partial \mathbf{W_{vh}}}]}_{\text{model's expectation}} \\
&\simeq \frac{1}{N^{(t)}} \sum_{n=1}^{N^{(t)}} \frac{\partial \mathfrak{F}(\mathbf{V}_n^{(t)})}{\partial \mathbf{W_{vh}}} - \frac{\partial \mathfrak{F}(\mathbf{V}_n^{(t)*})}{\partial \mathbf{W_{vh}}}
\end{aligned}$$

The second term is estimated by negative samples $\mathbf{V}_n^{(t)*}$ obtained from k-step Gibbs chain starting at $\mathbf{V}_n^{(t)}$ samples. $P_{data}(\widehat{\mathbf{V}}^{(t)}, \mathbf{h}^{(t)}) = P(\mathbf{h}^{(t)}|\widehat{\mathbf{V}}^{(t)}) P_{data}(\widehat{\mathbf{V}}^{(t)})$ and $P_{data}(\widehat{\mathbf{V}}^{(t)}) = \frac{1}{N^{(t)}} \sum_{n}^{N^{(t)}} \delta(\widehat{\mathbf{V}}^{(t)} - \mathbf{V}_n^{(t)})$ is the empirical distribution on the observable. $P_{model}(\mathbf{V}_n^{(t)*}, \mathbf{h}_n^{(t)})$ is defined in eq. 4. The free energy $\mathfrak{F}(\mathbf{V}_n^{(t)})$ is related to normalized probability of $\mathbf{V}_n^{(t)}$ as $P(\mathbf{V}_n^{(t)}) \equiv \exp^{-\mathfrak{F}(\mathbf{V}_n^{(t)})} / Z_n^{(t)}$ and as follows-

$$\begin{aligned}
\mathfrak{F}(\mathbf{V}_n^{(t)}) = &- \sum_{k=1}^{K} \hat{v}_n^{k,(t)} b_v^k - \sum_{j=1}^{F} \log(1+ \\
&\exp(D_n^{(t)} b_{h,j} + \sum_{k=1}^{K} \hat{v}_n^{k,(t)} W_j^k))
\end{aligned}$$

Gradient approximations w.r.t. RSM parameters,

$$\begin{aligned}
\frac{\partial C_t}{\partial \mathbf{b_v}^{(t)}} &\simeq \sum_{n=1}^{N^{(t)}} \hat{\mathbf{v}}_n^{(t)*} - \hat{\mathbf{v}}_n^{(t)} \\
\frac{\partial C_t}{\partial \mathbf{b_h}^{(t)}} &\simeq \sum_{n=1}^{N^{(t)}} \sigma(\mathbf{W_{vh}} \hat{\mathbf{v}}_n^{(t)*} - D_n^{(t)} \mathbf{b_h}^{(t)}) \\
&\quad - \sigma(\mathbf{W_{vh}} \hat{\mathbf{v}}_n^{(t)} - D_n^{(t)} \mathbf{b_h}^{(t)}) \\
\frac{\partial C_t}{\partial \mathbf{W_{vh}}} &\simeq \sum_{t=1}^{T} \sum_{n=1}^{N^{(t)}} \sigma(\mathbf{W_{vh}} \hat{\mathbf{v}}_n^{(t)*} - D_n^{(t)} \mathbf{b_h}^{(t)}) \\
&\quad \hat{\mathbf{v}}_n^{(t)*T} - \sigma(\mathbf{W_{vh}} \hat{\mathbf{v}}_n^{(t)} - D_n^{(t)} \mathbf{b_h}^{(t)}) \hat{\mathbf{v}}_n^{(t)T}
\end{aligned} \tag{5}$$

The estimated gradients w.r.t. RSM biases are back-propagated via hidden-to-bias parameters (eq 1) to compute gradients w.r.t. RNN connections ($\mathbf{W_{uh}}$, $\mathbf{W_{uv}}$, $\mathbf{W_{vu}}$ and $\mathbf{W_{uu}}$) and biases ($\mathbf{b_h}$, $\mathbf{b_v}$ and $\mathbf{b_u}$).

$$\begin{aligned}
\frac{\partial C}{\partial \mathbf{W_{uh}}} &= \sum_{t=1}^{T} \frac{\partial C_t}{\partial \mathbf{b_h}^{(t)}} \mathbf{u}^{(t-1)T} \\
\frac{\partial C}{\partial \mathbf{W_{uv}}} &= \sum_{t=1}^{T} \frac{\partial C_t}{\partial \mathbf{b_v}^{(t)}} \mathbf{u}^{(t-1)T} \\
\frac{\partial C}{\partial \mathbf{W_{vu}}} &= \sum_{t=1}^{T} \frac{\partial C_t}{\partial \mathbf{u}^{(t)}} \mathbf{u}^{(t)}(1 - \mathbf{u}^{(t)}) \sum_{n=1}^{N^{(t)}} \hat{\mathbf{v}}_n^{(t)T} \\
\frac{\partial C}{\partial \mathbf{b_h}} &= \sum_{t=1}^{T} \frac{\partial C_t}{\partial \mathbf{b_h}^{(t)}} \text{ and } \frac{\partial C}{\partial \mathbf{b_v}} = \sum_{t=1}^{T} \frac{\partial C_t}{\partial \mathbf{b_v}^{(t)}} \\
\frac{\partial C}{\partial \mathbf{b_u}} &= \sum_{t=1}^{T} \frac{\partial C_t}{\partial \mathbf{u}^{(t)}} \mathbf{u}^{(t)}(1 - \mathbf{u}^{(t)}) \\
\frac{\partial C}{\partial \mathbf{W_{uu}}} &= \sum_{t=1}^{T} \frac{\partial C_t}{\partial \mathbf{u}^{(t)}} \mathbf{u}^{(t)}(1 - \mathbf{u}^{(t)}) \mathbf{u}^{(t-1)T}
\end{aligned} \tag{6}$$

Parameter	Value(s)	Optimal
epochs	1000	1000
CD iterations	15	15
learning rate	0.1, 0.03, 0.001	0.001
hidden size	20, 30, 50	30

Table 2: Hyperparameters for RNN-RSM model

model	metric				
	SumPPL	*Err*	*mean-COH*	*median-COH*	*TTD*
DTM	10.9	8.10	0.1514	0.1379	0.084
RNN-RSM	**3.8**	**7.58**	**0.1620**	**0.1552**	0.268

Table 3: State-of-the-art Comparison: Generalization (*PPL* and *Err*), Topic Interpretation (*COH*) and Evolution (*TTD*) in DTM and RNN-RSM models

For the single-layer RNN-RSM, the BPTT recurrence relation for $0 \leq t < T$ is given by-

$$\frac{\partial C_t}{\partial \mathbf{u}^{(t)}} = \mathbf{W_{uu}} \frac{\partial C_{t+1}}{\partial \mathbf{u}^{(t+1)}} \mathbf{u}^{(t+1)}(1 - \mathbf{u}^{(t+1)}) + \mathbf{W_{uh}} \frac{\partial C_{t+1}}{\partial \mathbf{b_h}^{(t+1)}} + \mathbf{W_{uv}} \frac{\partial C_{t+1}}{\partial \mathbf{b_v}^{(t+1)}}$$

where $\mathbf{u}^{(0)}$ being a parameter and $\frac{\partial C_T}{\partial \mathbf{u}^{(T)}} = 0$.

See *Training RNN-RSM with BPTT* in Algo 1.

3 Evaluation

3.1 Dataset and Experimental Setup

We use the processed dataset (Gollapalli and Li, 2015), consisting of EMNLP and ACL conference papers from the year 1996 through 2014 (Table 1). We combine papers for each year from the two venues to prepare the document collections over time. We use ExpandRank (Wan and Xiao, 2008) to extract top 100 keyphrases for each paper, including unigrams and bigrams. We split the bigrams to unigrams to create a dictionary of all unigrams and bigrams. The dictionary size (K) and word count are 3390 and 5.19 M, respectively.

We evaluate RNN-RSM against static (RSM, LDA) and dynamic (DTM) topics models for topic and keyword evolution in NLP research over time. Individual 19 different RSM and LDA models are trained for each year, while DTM[2] and RNN-RSM are trained over the years with 19 time steps, where paper collections for a year is input at each time step. RNN-RSM is initialized with RSM ($\mathbf{W_{vh}}$, $\mathbf{b_v}$, $\mathbf{b_h}$) trained for the year 2014.

We use perplexity to choose the number of topics (=30). See Table 2 for hyperparameters.

3.2 Generalization in Dynamic Topic Models

Perplexity: We compute the perplexity on unobserved documents ($\widehat{\mathbf{V}}^{(t)}$) at each time step as

$$\text{PPL}(\widehat{\mathbf{V}}^{(t)}, t) = \exp\left(-\frac{1}{N^{(t)}} \frac{\sum_{n=1}^{N^{(t)}} \log P(\mathbf{V}_n^{(t)})}{\sum_{n=1}^{N^{(t)}} D_n^{(t)}}\right)$$

where t is the time step. $N^{(t)}$ is the number of documents in a collection ($\widehat{\mathbf{V}}^{(t)}$) at time t. Better models have lower perplexity values, suggesting less uncertainties about the documents. For held-out documents, we take 10 documents from each time step i.e. total 190 documents and compute perplexity for 30 topics. Fig 3d shows the comparison of perplexity values for unobserved documents from DTM and RNN-RSM at each time step. The *SumPPL* (Table 3) is the sum of PPL values for the held-out sets of each time step.

Document Time Stamp Prediction: To further assess the dynamic topics models, we split the document collections at each time step into 80-20% train-test, resulting in 1067 held-out documents. We predict the time stamp (dating) of a document by finding the most likely (with the lowest perplexity) location over the time line. See the *mean absolute error* (*Err*) in year for the held-out in Table 3. Note, we do not use the time stamp as observables during training.

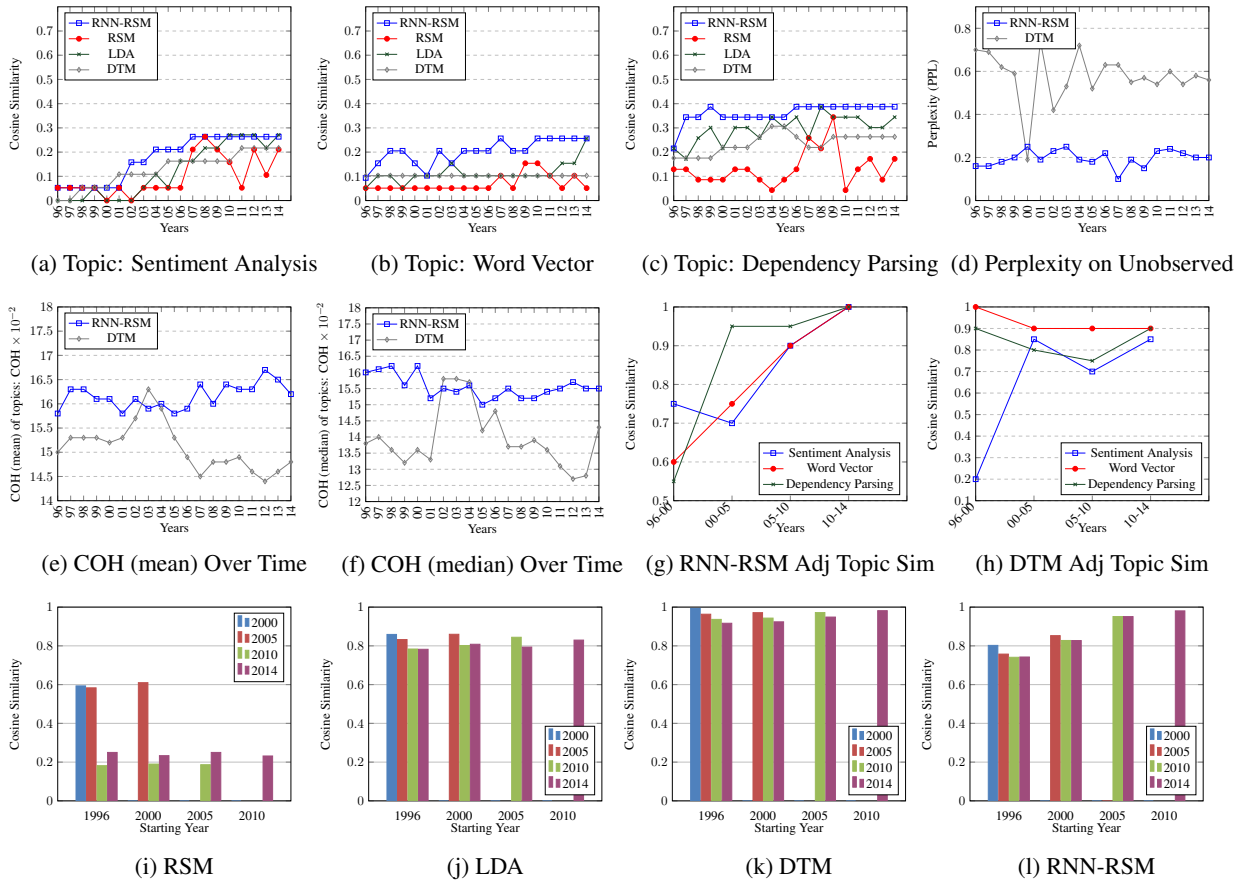
3.3 TSD, TED: Topic Evolution over Time

Topic Detection: To extract topics from each RSM, we compute posterior $P(\widehat{\mathbf{V}}^{(t)}|h_j = 1)$ by activating a hidden unit and deactivating the rest in a hidden layer. We extract the top 20 terms for every 30 topic set from 1996-2014, resulting in $|Q|_{max} = 19 \times 30 \times 20$ possible topic terms.

Topic Popularity: To determine topic *popularity*, we selected three popular topics (*Sentiment Analysis*, *Word Vector* and *Dependency Parsing*) in NLP research and create a set[3] of key-terms (including unigrams and bigrams) for each topic. We compute cosine similarity of the key-terms defined for each selected topic and topics discovered by the topic models over the years. We consider the discovered topic that is the most similar to the key-terms in the target topic and plot the similarity values in Figure 3a, 3b and 3b. Observe that RNN-RSM shows better topic evolution for the three emerging topics. LDA and RSM show

[2] https://radimrehurek.com/gensim/models/dtmmodel.html

[3] topic-terms to be released with code



(a) Topic: Sentiment Analysis (b) Topic: Word Vector (c) Topic: Dependency Parsing (d) Perplexity on Unobserved

(e) COH (mean) Over Time (f) COH (median) Over Time (g) RNN-RSM Adj Topic Sim (h) DTM Adj Topic Sim

(i) RSM (j) LDA (k) DTM (l) RNN-RSM

Figure 3: (a, b, c): Topic popularity by LDA, RSM, DTM and RNN-RSM over time (d): Perplexity on the unobserved document collections over time (e, f): Mean and Median Topic Coherence (g, h): Topic Evolution (i,j,k,l): Topic focus change over time. Adj- Adjacent; Sim- Similarity

topical locality in Figure 3c attributed to no correlation in topic dynamics over time, while in Figure 3b, DTM does not capture evolution of topic *Word Vector*.

Topic Drift (Focus Change): To compute the topic *focus* change over the years, we first split the time period 1996-2014 into five parts:{1996, 2000, 2005, 2010, 2014}. The cosine similarity scores are computed between the topic sets discovered in a particular year and the years preceding it in the above set, for example the similarity scores between the topic-terms in (1996, 2000), (1996, 2005), (1996, 2010) and (1996, 2014), respectively. Figure 3i, 3j, 3k and 3l demonstrate that RNN-RSM shows higher *convergence* in topic focus over the years, compared to LDA and RSM. In RNN-RSM, the topic similarity is gradually increased over time, however not in DTM. The higher similarities in the topic sets indicate that new/existing topics and words do not appear/disappear over time.

We compute topic-term drift (TTD) to show the changing topics from initial to final year, as

$$TTD = 1.0 - cosineSimilarity(\mathbf{Q}^{(t)}, \mathbf{Q}^{(t')})$$

where $\mathbf{Q}$ is the set of all topic-terms for time step t. Table 3 shows that TTD (where t=1996 and t'=2014) are 0.268 and 0.084 for RNN-RSM and DTM, respectively. It suggests that the higher number of new topic-terms evolved in RNN-RSM, compared to DTM. Qualitatively, the Table 4 shows the topics observed with the highest and lowest cosine drifts in DTM and RNN-RSM.

In Figure 3g and 3h, we also illustrate the temporal *evolution* (drift) in the selected topics by computing cosine similarity on their adjacent topic vectors over time. The topic vectors are selected similarly as in computing topic popularity. We observe better TED in RNN-RSM than DTM for the three emerging topics in NLP research. For instance, for the selected topic *Word Vector*, the red line in DTM (Fig 3h) shows no drift (for x-axis 00-05, 05-10 and 10-14), suggesting the topic-terms in the adjacent years are similar and does not evolve.

Drift	Model (year)	Topic Terms
0.20	**DTM** (1996)	document, retrieval, query, documents, information, search, information retrieval, queries, terms, words, system, results, performance, method, approach
	DTM (2014)	document, query, search, documents, queries, information, retrieval, method, results, information retrieval, research, terms, other, approach, knowledge
0.53	**DTM** (1996)	semantic, lexical, structure, syntactic, argument, frame, example, lexicon, information, approach, source, function, figure, verbs, semantic representation
	DTM (2014)	semantic, argument, frame, sentence, syntactic, semantic parsing, structure, semantic role, example, role labeling, language, learning, logical form, system, lexicon
0.20	**RNN-RSM** (1996)	reordering, statistical machine, translation model, translations, arabic, word align, translation probability, word alignment, translation system, source word, ibm model, source sentence, english translation, target language, word segmentation
	RNN-RSM (2014)	reordering, statistical machine, translation model, translations, arabic, word align, translation probability, word alignment, translation system, source word, reordering model, bleu score, smt system, english translation, target language
0.53	**RNN-RSM** (1996)	input, inference, semantic representation, distributional models, logical forms, space model, clustering algorithm, space models, similar word, frequent word, meaning representation, lexical acquisition, new algorithm, same context, multiple words
	RNN-RSM (2014)	input, inference, word vector, word vectors, vector representation, semantic representation, distributional models, semantic space, space model, semantic parser, vector representations, neural language, logical forms, cosine similarity, clustering algorithm

Table 4: Topics (top 15 words) with the highest and lowest drifts (cosine) observed in DTM and RNN-RSM

3.4 Topic Interpretability

Beyond perplexities, we also compute topic coherence (Chang et al., 2009; Newman et al., 2009; Das et al., 2015) to determine the meaningful topics captured. We use the coherence measure proposed by Aletras and Stevenson (2013) that retrieves co-occurrence counts for the set of topic words using Wikipedia as a reference corpus to identify context features (window=5) for each topic word. Relatedness between topic words and context features is measured using normalized pointwise mutual information (NPMI), resulting in a single vector for every topic word. The coherence (*COH*) score is computed as the arithmetic mean of the cosine similarities between all word pairs. Higher scores imply more coherent topics. We use Palmetto[4] library to estimate coherence.

Quantitative: We compute mean and median coherence scores for each time step using the corresponding topics, as shown in Fig 3e and 3f. Table 3 shows *mean-COH* and *median-COH* scores, computed by mean and median of scores from Fig 3e and 3f, respectively. Observe that RNN-RSM captures topics with higher coherence.

Qualitative: Table 5 shows topics (top-10 words) with the highest and lowest coherence scores.

3.5 TTC: Trending Keywords over time

We demonstrate the capability of RNN-RSM to capture word evolution (usage) in topics over time. We define: *keyword-trend* and SPAN. The *keyword-trend* is the appearance/disappearance of the keyword in topic-terms detected over time, while SPAN is the length of the longest sequence of the keyword appearance in its keyword trend.

[4] github.com/earthquakesan/palmetto-py

DTM (2001)	**RNN-RSM** (2001)	**DTM** (2012)	**RNN-RSM** (1997)
semantic	words	discourse	parse
frame	models	relation	cluster
argument	grammar	relations	clustering
syntactic	trees	structure	results
structure	dependency parsing	sentence	query
lexical	parsers	class	pos tag
example	dependency trees	lexical	queries
information	parsing	argument	retrieval
annotation	parse trees	corpus	coreference
lexicon	dependency parse	other	logical form
COH: 0.268	0.284	0.064	0.071

Table 5: Topics with the highest and lowest coherence

Let $\widehat{\mathbf{Q}}_{model} = \{\mathbf{Q}^{(t)}_{model}\}_{t=1}^{T}$ be a set of sets[5] of topic-terms discovered by the $model$ (LDA, RSM, DTM and RNN-RSM) over different time steps. Let $\mathbf{Q}^{(t)} \in \widehat{\mathbf{Q}}_{model}$ be the topic-terms at time step t. The keyword-trend for a keyword k is a time-ordered sequence of 0s and 1s, as

$$\text{trend}_k(\widehat{\mathbf{Q}}) = [\text{find}(k, \mathbf{Q}^{(t)})]_{t=1}^{T}$$

$$\text{where;} \quad \text{find}(k, \mathbf{Q}^{(t)}) = \begin{cases} 1 & \text{if } k \in \mathbf{Q}^{(t)} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

And the SPAN (S_k) for the kth keyword is-

$$S_k(\widehat{\mathbf{Q}}) = \text{length}\big(\text{longestOnesSeq}(\text{trend}_k(\widehat{\mathbf{Q}})\big)$$

We compute keyword-trend and SPAN for each term from the set of some popular terms. We define average-SPAN for all the topic-terms appearing in the topics discovered over the years,

$$\text{avg-SPAN}(\widehat{\mathbf{Q}}) = \frac{1}{||\widehat{\mathbf{Q}}||} \sum_{\{k|\mathbf{Q}^{(t)} \in \widehat{\mathbf{Q}} \wedge k \in \mathbf{Q}^{(t)}\}} \frac{S_k(\widehat{\mathbf{Q}})}{\hat{v}^k}$$

$$= \frac{1}{||\widehat{\mathbf{Q}}||} \sum_{\{k|\mathbf{Q}^{(t)} \in \widehat{\mathbf{Q}} \wedge k \in \mathbf{Q}^{(t)}\}} S_k^{dict}(\widehat{\mathbf{Q}})$$

[5] a set by **bold** and set of sets by $\widehat{\textbf{bold}}$

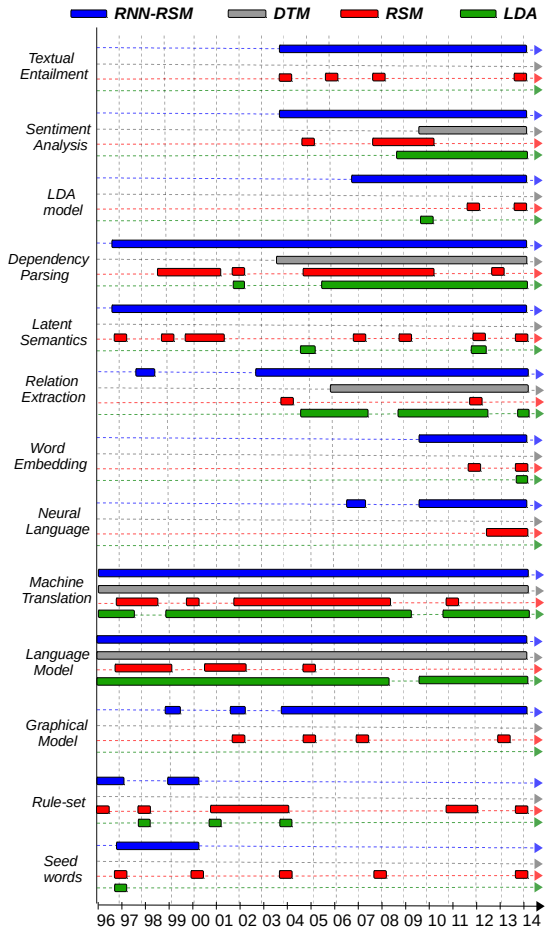


Figure 4: Keyword-trend by RNN-RSM, DTM, RSM, LDA. Bar: Keyword presence in topics for the year

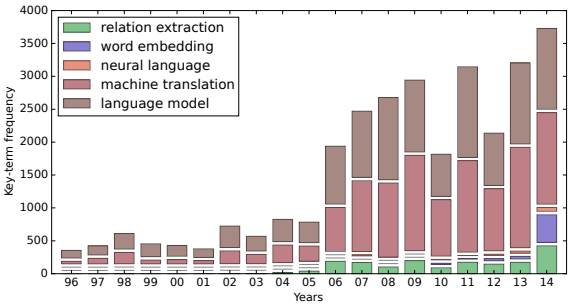


Figure 5: Key-term frequency in the input over years

Term	$\hat{v^k}$	LDA		RSM		DTM		RNN-RSM	
		S_k	S_k^{dict}	S_k	S_k^{dict}	S_k	S_k^{dict}	S_k	S_k^{dict}
Textual entailment	918	0	.000	1	.001	0	.000	**11**	.011
Sentiment analysis	1543	6	.004	3	.002	5	.0032	**11**	0.007
Lda model	392	1	.003	1	.002	0	.000	**8**	.020
Dependency parsing	3409	9	.003	5	.001	11	.0032	**18**	.005
Latent semantic	974	1	.001	2	.002	0	.000	**18**	.018
Relation extraction	1734	4	.002	1	.001	9	.0052	**12**	.007
Word embedding	534	1	.002	1	.002	0	.000	**5**	.009
Neural language	121	0	.000	3	.025	0	.000	**5**	.041
Machine translation	11741	11	.001	7	.001	**19**	.0016	**19**	.002
Language model	11768	13	.001	3	.000	**19**	.0016	**19**	.002
Graphical model	680	0	.000	1	.001	0	.000	**11**	.016
Rule set	589	1	.0017	**4**	.0068	0	.000	2	.0034
Seed words	396	1	.0025	1	.0025	0	.000	**4**	.0101
avg-SPAN($\widehat{\mathbf{Q}}$)			.002		.007		.003		**.018**
$\|\|\widehat{\mathbf{Q}}_{model}\|\|$			926		2274		335		731

Table 6: SPAN (S_k) for selected terms, avg-SPAN and set $||\widehat{\mathbf{Q}}||$ by LDA, RSM, DTM and RNN-RSM

where $||\widehat{\mathbf{Q}}|| = |\{k|\mathbf{Q}^{(t)} \in \widehat{\mathbf{Q}} \wedge k \in \mathbf{Q}^{(t)}\}|$ is the count of unique topic-terms and $\hat{v^k} = \sum_{t=1}^{T}\sum_{j=1}^{D_t} v_{j,t}^k$ denotes the count of k^{th} keyword.

In Figure 4, the keyword-trends indicate emergence (appearance/disappearance) of the selected popular terms in topics discovered in ACL and EMNLP papers over time. Observe that RNN-RSM captures longer SPANs for popular keywords and better word usage in NLP research. For example: *Word Embedding* is one of the top keywords, appeared locally (Figure 5) in the recent years. RNN-RSM detects it in the topics from 2010 to 2014, however DTM does not. Similarly, for *Neural Language*. However, *Machine Translation* and *Language Model* are globally appeared in the input document collections over time and captured in the topics by RNN-RSM and DTM. We also show keywords (*Rule-set* and *Seed Words*) that disappeared in topics over time.

Higher SPAN suggests that the model is capable in capturing trending keywords. Table 6 shows corresponding comparison of SPANs for the 13 selected keywords. The SPAN S_k for each keyword is computed from Figure 4. Observe that $||\widehat{\mathbf{Q}}||_{DTM} < ||\widehat{\mathbf{Q}}||_{RNN-RSM}$ suggests new topics and words emerged over time in RNN-RSM, while higher SPAN values in RNN-RSM suggest better trends. Figure 6 shows how the word usage, captured by DTM and RNN-RSM for the topic *Word Vector*, changes over 19 years in NLP research. RNN-RSM captures popular terms *Word Embedding* and *Word Representation* emerged in it.

4 Discussion: RNN-RSM vs DTM

Architecture: RNN-RSM treats document's stream as high dimensional sequences over time and models the complex conditional probability distribution i.e. *heteroscedasticity* in document collections and topics over time by a temporal stack of RSMs (undirected graphical model), conditioned on time-feedback connections using RNN (Rumelhart et al., 1985). It has two hidden layers: **h** (stochastic binary) to capture topical information, while **u** (deterministic) to convey temporal information via BPTT that models the topic dependence at a time step t on *all* the previous steps $\tau < t$. In contrast, DTM is built upon

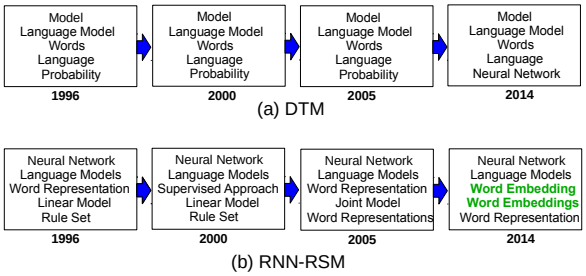


Figure 6: Word usage for emerging topic *Word Vector* over time, captured by DTM and RNN-RSM

LDA (directed model), where Dirichlet distribution on words is not amenable to sequential modeling, therefore its natural parameters (topic and topic proportion distributions) for each topic are chained, instead of latent topics that results in intractable inference in topic detection and chaining.

Topic Dynamics: The introduction of explicit connection in latent topics in RNN-RSM allow new topics and words for the underlying topics to appear or disappear over time by the dynamics of topic correlations. As discussed, the distinction of $\mathbf{h}$ and $\mathbf{u}$ permits the latent topic $\mathbf{h}^{(t)}$ to capture new topics, that may not be captured by $\mathbf{h}^{(t-1)}$.

DTM assumes a fixed number of global topics and models their distribution over time. However, there is no such assumption in RNN-RSM. We fixed the topic count in RNN-RSM at each time step, since $\mathbf{W_{vh}}$ is fixed over time and RSM biases turn off/on terms in each topic. However, this is fundamentally different for DTM. E.g. a unique label be assigned to each of the 30 topics at any time steps t and t'. DTM follows the sets of topic labels: $\{TopicLabels^{(t)}\}_{k=1}^{30} = \{TopicLabels^{(t')}\}_{k=1}^{30}$, due to eq (1) in Blei and Lafferty (2006) (discussed in section 5) that limits DTM to capture new (or local) topics or words appeared over time. It corresponds to the keyword-trends (section 3.5).

Optimization: The RNN-RSM is based on Gibbs sampling and BPTT for inference while DTM employs complex variational methods, since applying Gibbs sampling is difficult due to the nonconjugacy of the Gaussian and multinomial distributions. Thus, easier learning in RNN-RSM.

For all models, approximations are solely used to compute the likelihood, either using variational approaches or contrastive divergence; perplexity was then computed based on the approximated likelihood. More specifically, we use variational approximations to compute the likelihood for DTM (Blei and Lafferty, 2006). For RSM and RNN-RSM, the respective likelihoods are approximated using the standard Contrastive Divergence (CD). While there are substantial differences between variational approaches and CD, and thus in the manner the likelihood for different models is estimated - both approximations work well for the respective family of models in terms of approximating the true likelihood. Consequently, perplexities computed based on these approximated likelihoods are indeed comparable.

5 Conclusion and Future Work

We have proposed a neural temporal topic model which we name as RNN-RSM, based on probabilistic undirected graphical topic model RSM with time-feedback connections via deterministic RNN, to capture temporal relationships in historical documents. The model is the first of its kind that learns topic dynamics in collections of different-sized documents over time, within the generative and neural network framework. The experimental results have demonstrated that RNN-RSM shows better generalization (perplexity and time stamp prediction), topic interpretation (coherence) and evolution (popularity and drift) in scientific articles over time. We also introduced SPAN to illustrate topic characterization.

In future work, we forsee to investigate learning dynamics in variable number of topics over time. It would also be an interesting direction to investigate the effect of the skewness in the distribution of papers over all years. Further, we see a potential application of the proposed model in learning the time-aware i.e. dynamic word embeddings (Aitchison, 2001; Basile et al., 2014; Bamler and Mandt, 2017; Rudolph and Blei, 2018; Yao et al., 2018) in order to capture language evolution over time, instead of document topics.

Acknowledgments

We thank Sujatha Das Gollapalli for providing us with the data sets used in the experiments. We express appreciation for our colleagues Florian Buettner, Mark Buckley, Stefan Langer, Ulli Waltinger and Usama Yaseen, and anonymous reviewers for their in-depth review comments. This research was supported by Bundeswirtschaftsministerium (bmwi.de), grant 01MD15010A (Smart Data Web) at Siemens AG- CT Machine Intelligence, Munich Germany.

References

Jean Aitchison. 2001. *Language change: progress or decay?*. Cambridge University Press.

Nikolaos Aletras and Mark Stevenson. 2013. Evaluating topic coherence using distributional semantics. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS)*. Potsdam, Germany, pages 13–22.

James Allan. 2002. Introduction to topic detection and tracking. In *Topic detection and tracking*, Springer, pages 1–16.

James Allan, Jaime G Carbonell, George Doddington, Jonathan Yamron, and Yiming Yang. 1998. Topic detection and tracking pilot study final report. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*. Virginia, US, pages 194–218.

Robert Bamler and Stephan Mandt. 2017. Dynamic word embeddings. In *Proceedings of the 34th International Conference on Machine Learning*. Sydney, Australia, pages 380–389.

Pierpaolo Basile, Annalina Caputo, and Giovanni Semeraro. 2014. Analysing word meaning over time by exploiting temporal random indexing. In *Proceedings of the 1st Italian Conference on Computational Linguistics (CLiC-it)*. Pisa University Press, Pisa, Italy.

David M. Blei and John D. Lafferty. 2006. Dynamic topic models. In *Proceedings of the 23rd International Conference on Machine Learning*. Association for Computing Machinery, Pittsburgh, Pennsylvania USA, pages 113–120.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Proceedings of Machine Learning Research* 3(Jan):993–1022.

Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. 2012. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *Proceedings of the 29th International Conference on Machine Learning*. Edinburgh, Scotland UK.

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L. Boyd-Graber, and David M. Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., Vancouver, Canada, pages 288–296.

Rajarshi Das, Manzil Zaheer, and Chris Dyer. 2015. Gaussian lda for topic models with word embeddings. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, Beijing, China, volume 1, pages 795–804.

Peter V. Gehler, Alex D. Holub, and Max Welling. 2006. The rate adapting poisson model for information retrieval and object recognition. In *Proceedings of the 23rd International Conference on Machine Learning*. Association for Computing Machinery, Pittsburgh, Pennsylvania USA, pages 337–344.

Sujatha Das Gollapalli and Xiaoli Li. 2015. Emnlp versus acl: Analyzing nlp research over time. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 2002–2006.

Pankaj Gupta, Thomas Runkler, Heike Adel, Bernt Andrassy, Hans-Georg Zimmermann, and Hinrich Schütze. 2015a. Deep learning methods for the extraction of relations in natural language text. Technical report, Technical University of Munich, Germany.

Pankaj Gupta, Thomas Runkler, and Bernt Andrassy. 2015b. Keyword learning for classifying requirements in tender documents. Technical report, Technical University of Munich, Germany.

Pankaj Gupta, Hinrich Schütze, and Bernt Andrassy. 2016. Table filling multi-task recurrent neural network for joint entity and relation extraction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan, pages 2537–2547.

Pankaj Gupta, Udhayaraj Sivalingam, Sebastian Pölsterl, and Nassir Navab. 2015c. Identifying patients with diabetes using discriminative restricted boltzmann machines. Technical report, Technical University of Munich, Germany.

David Hall, Daniel Jurafsky, and Christopher D. Manning. 2008. Studying the history of ideas using topic models. In *Proceedings of the conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Honolulu, Hawaii, pages 363–371.

Geoffrey Hinton and Ruslan Salakhutdinov. 2009. Replicated softmax: an undirected topic model. In *Advances in Neural Information Processing Systems 22*. Curran Associates, Inc., Vancouver, Canada, pages 1607–1614.

Geoffrey E. Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation* 14(8):1771–1800.

Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural Computation* 18(7):1527–1554.

Tomoharu Iwata, Takeshi Yamada, Yasushi Sakurai, and Naonori Ueda. 2010. Online multiscale dynamic topic models. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, Washington DC, USA, pages 663–672.

David Newman, Sarvnaz Karimi, and Lawrence Cavedon. 2009. External evaluation of topic models. In *Proceedings of the 14th Australasian Document Computing Symposium*. Citeseer, Sydney, Australia.

Iulian Pruteanu-Malinici, Lu Ren, John Paisley, Eric Wang, and Lawrence Carin. 2010. Hierarchical bayesian modeling of topics in time-stamped documents. *IEEE transactions on pattern analysis and machine intelligence* 32(6):996–1011.

Maja Rudolph and David Blei. 2018. Dynamic bernoulli embeddings for language evolution. In *Proceedings of the 27th International Conference on World Wide Web Companion*. Lyon, France.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1985. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

Ankan Saha and Vikas Sindhwani. 2012. Learning evolving and emerging topics in social media: a dynamic nmf approach with temporal regularization. In *Proceedings of the 5th ACM International Conference on Web Search and Data Mining*. Association for Computing Machinery, Seattle, Washington USA, pages 693–702.

Aaron Schein, Hanna Wallach, and Mingyuan Zhou. 2016. Poisson-gamma dynamical systems. In *Advances in Neural Information Processing Systems 29*, Curran Associates, Inc., Barcelona, Spain, pages 5005–5013.

Paul Smolensky. 1986. Information processing in dynamical systems: Foundations of harmony theory. Technical report, Colorado University at Boulder Department of Computer Science.

Ilya Sutskever and Geoffrey Hinton. 2007. Learning multilevel distributed representations for high-dimensional sequences. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics*. San Juan, Puerto Rico, pages 548–555.

Ilya Sutskever, Geoffrey E. Hinton, and Graham W. Taylor. 2009. The recurrent temporal restricted boltzmann machine. In *Advances in Neural Information Processing Systems 22*. Curran Associates, Inc., Vancouver, Canada, pages 1601–1608.

Graham W. Taylor, Geoffrey E. Hinton, and Sam T. Roweis. 2007. Modeling human motion using binary latent variables. In *Advances in Neural Information Processing Systems 20*. Curran Associates, Inc., Vancouver, Canada, pages 1345–1352.

Ngoc Thang Vu, Heike Adel, Pankaj Gupta, and Hinrich Schütze. 2016a. Combining recurrent and convolutional neural networks for relation classification. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California USA, pages 534–539.

Ngoc Thang Vu, Pankaj Gupta, Heike Adel, and Hinrich Schütze. 2016b. Bi-directional recurrent neural network with ranking loss for spoken language understanding. In *Proceedings of the Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, Shanghai, China, pages 6060–6064.

Xiaojun Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of the 23rd National Conference on Artificial Intelligence*. Chicago, Illinois USA, volume 8, pages 855–860.

Xuerui Wang and Andrew McCallum. 2006. Topics over time: a non-markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, Philadelphia, Pennsylvania USA, pages 424–433.

Eric P. Xing, Rong Yan, and Alexander G. Hauptmann. 2005. Mining associated text and images with dual-wing harmoniums. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*. AUAI Press, Edinburgh, Scotland UK.

Zijun Yao, Yifan Sun, Weicong Ding, Nikhil Rao, and Hui Xiong. 2018. Dynamic word embeddings for evolving semantic discovery. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining (WSDM)*. Association for Computing Machinery, Los Angeles, California USA, pages 673–681.

Lessons from the Bible on Modern Topics: Low-Resource Multilingual Topic Model Evaluation

Shudong Hao
Computer Science
University of Colorado
Boulder, CO
shudong@colorado.edu

Jordan Boyd-Graber
Computer Science, iSchool, LSC, and UMIACS,
University of Maryland
College Park, MD
jbg@umiacs.umd.edu

Michael J. Paul
Information Science
University of Colorado
Boulder, CO
mpaul@colorado.edu

Abstract

Multilingual topic models enable document analysis across languages through coherent multilingual summaries of the data. However, there is no standard and effective metric to evaluate the quality of multilingual topics. We introduce a new intrinsic evaluation of multilingual topic models that correlates well with human judgments of multilingual topic coherence as well as performance in downstream applications. Importantly, we also study evaluation for low-resource languages. Because standard metrics fail to accurately measure topic quality when robust external resources are unavailable, we propose an adaptation model that improves the accuracy and reliability of these metrics in low-resource settings.

1 Introduction

Topic models provide a high-level view of the main themes of a document collection (Boyd-Graber et al., 2017). Document collections, however, are often not in a single language, driving the development of **multilingual** topic models. These models discover topics that are consistent across languages, providing useful tools for multilingual text analysis (Vulić et al., 2015), such as detecting cultural differences (Gutiérrez et al., 2016) and bilingual dictionary extraction (Liu et al., 2015).

Monolingual topic models can be evaluated through likelihood (Wallach et al., 2009b) or coherence (Newman et al., 2010), but topic model evaluation is not well understood in multilingual settings. Our contributions are two-fold. We introduce an improved intrinsic evaluation metric for multilingual topic models, called Crosslingual Normalized Pointwise Mutual Information (CNPMI, Section 2). We explore the behaviors of CNPMI at both the model and topic levels with six language pairs and varying model specifications. This metric correlates well with human judgments and crosslingual classification results (Sections 5 and 6).

We also focus on evaluation in low-resource languages, which lack large parallel corpora, dictionaries, and other tools that are often used in learning and evaluating topic models. To adapt CNPMI to these settings, we create a coherence estimator (Section 3) that extrapolates statistics derived from antiquated, specialized texts like the Bible: often the only resource available for many languages.

2 Evaluating Multilingual Coherence

A multilingual topic contains one topic for each language. For a multilingual topic to be meaningful to humans (Figure 1), the meanings should be consistent across the languages, in addition to coherent within each language (*i.e.*, all words in a topic are related).

This section describes our approach to evaluating the quality of multilingual topics. After defining the multilingual topic model, we describe topic model evaluation extending standard monolingual approaches to multilingual settings.

2.1 Multilingual Topic Modeling

Probabilistic topic models associate each document in a corpus with a distribution over latent topics, while each topic is associated with a distribution over words in the vocabulary. The most widely used topic model, latent Dirichlet allocation (Blei et al., 2003, LDA), can be extended to connect languages. These extensions require additional knowledge to link languages together.

One common encoding of multilingual knowledge is **document links** (indicators that documents are parallel or comparable), used in polylingual topic models (Mimno et al., 2009; Ni et al., 2009). In these models, each document d indexes a tuple of parallel/comparable language-specific documents,

Proceedings of NAACL-HLT 2018, pages 1090–1100
New Orleans, Louisiana, June 1 - 6, 2018.

$d^{(\ell)}$, and the language-specific "views" of a document share the document-topic distribution θ_d. The generative story for the document-links model is:

for *each topic* k *and each language* ℓ **do**
  Draw a distribution over words $\phi_{\ell k} \sim \text{Dirichlet}(\beta)$;
for *each document tuple* $d = \left(d^{(1)}, \ldots, d^{(L)}\right)$ **do**
  Draw a distribution over topics $\theta_d \sim \text{Dirichlet}(\alpha)$;
  for *each language* $\ell = 1, \ldots, L$ **do**
    for *each token* $t \in d^{(\ell)}$ **do**
      Draw a topic $z_n \sim \theta_d$;
      Draw a word $w_n \sim \phi_{\ell z}$;

Alternatively, word translations (Jagarlamudi and Daumé III, 2010), concept links (Gutiérrez et al., 2016; Yang et al., 2017), and multi-level priors (Krstovski et al., 2016) can also provide multilingual knowledges. Since the polylingual topic model is the most common approach for building multilingual topic models (Vulić et al., 2013, 2015; Liu et al., 2015; Krstovski and Smith, 2016), our study will focus on this model.

2.2 Monolingual Evaluation

Most automatic topic model evaluation metrics use co-occurrence statistics of word pairs from a reference corpus to evaluate topic coherence, assuming that coherent topics contain words that often appear together (Newman et al., 2010). The most successful (Lau et al., 2014) is normalized pointwise mutual information (Bouma, 2009, NPMI). NPMI compares the joint probability of words appearing together $\Pr(w_i, w_j)$ to their probability assuming independence $\Pr(w_i)\Pr(w_j)$, normalized by the joint probability:

$$\text{NPMI}(w_i, w_j) = \frac{\log \frac{\Pr(w_i, w_j)}{\Pr(w_i)\Pr(w_j)}}{\log \Pr(w_i, w_j)}. \quad (1)$$

The word probabilities are calculated from a **reference corpus**, $\mathcal{R}$, typically a large corpus such as Wikipedia that can provide meaningful co-occurrence patterns that are independent of the target dataset.

The quality of topic k is the average NPMI of all word pairs (w_i, w_j) in the topic:

$$\text{NPMI}_k = \frac{-1}{\binom{C}{2}} \sum_{i \in \mathcal{W}(k,C)} \sum_{j \neq i} \text{NPMI}(w_i, w_j), \quad (2)$$

where $\mathcal{W}(k, C)$ are the C most probable words in the topic-word distribution ϕ_k (the number of words is the topic's **cardinality**). Higher NPMI_k means the topic's top words are more coupled.

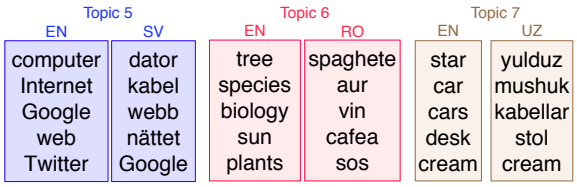


Figure 1: Topic 5 is multilingually coherent: both the English and Swedish topics are about technology. Topic 6 is about biology in English but food in Romanian, so it is low quality although coherent monolingually. Topic 7 is monolingually incoherent, so it is a low quality topic even if it contains word translations.

2.3 Existing Multilingual Evaluations

While automatic evaluation has been well-studied for monolingual topic models, there are no robust evaluations for multilingual topic models. We first consider two straightforward metrics that could be used for multilingual evaluation, both with limitations. We then propose an extension of NPMI that addresses these limitations.

Internal Coherence. A simple adaptation of NPMI is to calculate the monolingual NPMI score for each language independently and take the average. We refer this as internal NPMI (INPMI) as it evaluates coherence *within* a language. However, this metric does not consider whether the topic is coherent *across* languages—that is, whether a language-specific word distribution $\phi_{\ell_1 k}$ is related to the corresponding distribution in another language, $\phi_{\ell_2 k}$.

Crosslingual Consistency. Another straightforward measurement is Matching Translation Accuracy (Boyd-Graber and Blei, 2009, MTA), which counts the number of word translations in a topic between two languages using a bilingual dictionary. This metric can measure whether a topic is well-aligned across languages *literally*, but cannot capture non-literal more holistic similarities across languages.

2.4 New Metric: Crosslingual NPMI

We extend NPMI to multilingual models, with a metric we call crosslingual normalized pointwise mutual information (CNPMI). This metric will be the focus of our experiments.

A multilingually coherent topic means that if w_{i,ℓ_1} in language ℓ_1 and w_{j,ℓ_2} in language ℓ_2 are in the same topic, they should appear in similar contexts in comparable or parallel corpora $\mathcal{R}^{(\ell_1,\ell_2)}$.

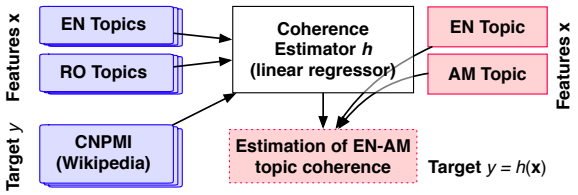


Figure 2: The coherence estimator takes multilingual topics and features from them then outputs an estimated topic coherence.

Our adaptation of NPMI is based on the same principles as the monolingual version, but focuses on the co-occurrences of *bilingual* word pairs. Given a bilingual word pair $(w_{i,\ell_1}, w_{j,\ell_2})$ the co-occurrence of this word pair is the event where word w_{i,ℓ_1} appears in a document in language ℓ_1 and the word w_{j,ℓ_2} appears in a comparable or parallel document in language ℓ_2.

The co-occurrence probability of each bilingual word pair is:

$$\Pr\left(w_{i,\ell_1}, w_{j,\ell_2}\right) \triangleq \frac{\left|\left\{\mathbf{d} : w_{i,\ell_1} \in d^{(\ell_1)}, w_{j,\ell_2} \in d^{(\ell_2)}\right\}\right|}{\left|\mathcal{R}^{(\ell_1,\ell_2)}\right|}, \quad (3)$$

where $\mathbf{d} = \left(d^{(\ell_1)}, d^{(\ell_2)}\right)$ is a pair of parallel/comparable documents in the reference corpus $\mathcal{R}^{(\ell_1,\ell_2)}$. When one or both words in a bilingual pair do not appear in the reference corpus, the co-occurrence score is zero.

Similar to monolingual settings, CNPMI for a bilingual topic k is the average of the NPMI scores of all C^2 bilingual word pairs,

$$\text{CNPMI}(\ell_1, \ell_2, k) = \frac{\sum_{i,j}^{C} \text{NPMI}\left(w_{i,\ell_1}, w_{j,\ell_2}\right)}{C^2}. \quad (4)$$

It is straightforward to generalize CNPMI from a language pair to multiple languages by averaging $\text{CNPMI}(\ell_i, \ell_j, k)$ over all language pairs (ℓ_i, ℓ_j).

3 Adapting to Low-Resource Languages

CNPMI needs a reference corpus for co-occurrence statistics. Wikipedia, which has good coverage of topics and vocabularies is a common choice (Lau and Baldwin, 2016). Unfortunately, Wikipedia is often unavailable or not large enough for low-resource languages. It only covers 282 languages,[1] and only 249 languages have more than 1,000 pages: many of pages are short or unlinked to a high-resource language. Since CNPMI requires comparable documents, the usable reference corpus is defined by *paired* documents.

Another option for a parallel reference corpus is the Bible (Resnik et al., 1999), which is available in most world languages;[2] however, it is small and archaic. It is good at evaluating topics such as family and religion, but not "modern" topics like biology and Internet. Without reference co-occurrence statistics relevant to these topics, CNPMI will fail to judge topic coherence—it must give the ambiguous answer of zero. Such a score could mean a totally incoherent topic where each word pair never appears together (Topics 6 in Figure 1), or an unjudgeable topic (Topic 5).

Our goal is to obtain a reliable estimation of topic coherence for low-resource languages when the Bible is the only reference. We propose a model that can correct the drawbacks of a Bible-derived CNPMI. While we assume bilingual topics paired with English, our approach can be applied to any high-resource/low-resource language pair.

We take Wikipedia's CNPMI from high-resource languages as accurate estimations. We then build a coherence *estimator* on topics from high-resource languages, with the Wikipedia CNPMI as the target output. We use linear regression using the below features. Given a topic in low-resource language, the estimator produces an estimated coherence (Figure 2).

3.1 Estimator Features

The key to the estimator is to find features that capture whether we should trust the Bible. For generality, we focus on features independent of the available resources other than the Bible. This section describes the features, which we split into four groups.

Base Features (BASE) Our base features include information we can collect from the Bible and the topic model: cardinality C, CNPMI and INPMI, MTA, and topic word coverage (TWC), which counts the percentage of topic words in a topic that appear in a reference corpus.

Crosslingual Gap (GAP) A low CNPMI score could indicate a topic pair where each language has a monolingually coherent topic but that are not about the same theme (Topic 6 in Figure 1). Thus, we add two features to capture this information

[1] https://meta.wikimedia.org/wiki/List_of_Wikipedias

[2] The Bible is available in 2,530 languages.

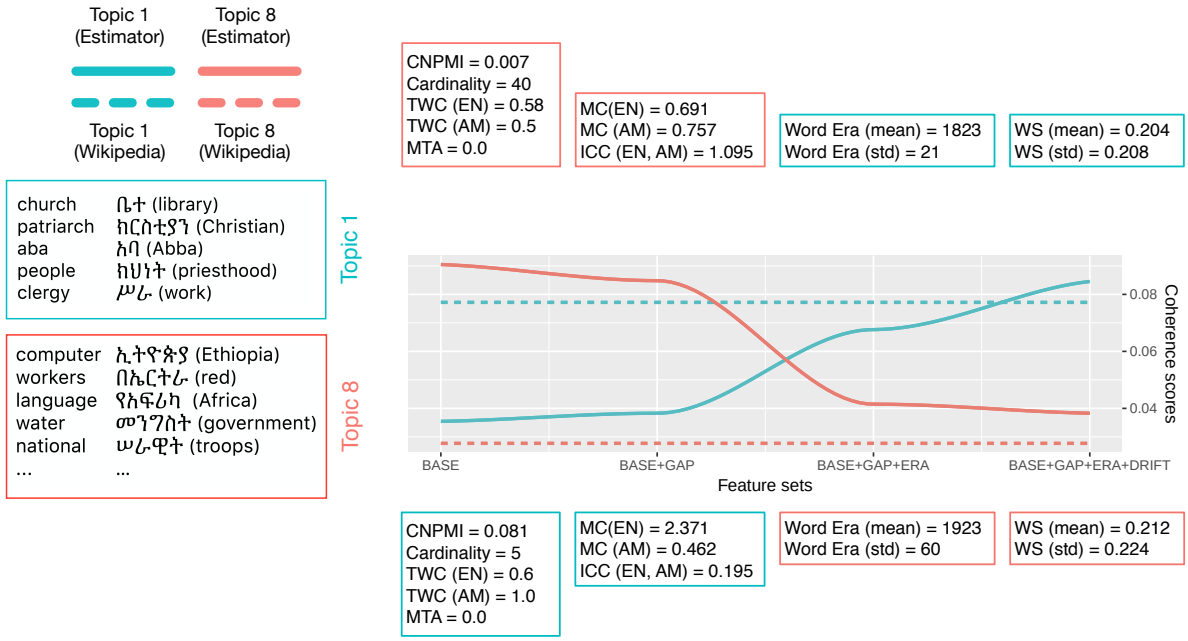


Figure 3: As the estimator adds additional features, the estimated topic coherence scores (solid lines) approach to Wikipedia CNPMI (dashed lines).

using the Bible: mismatch coefficients (MC) and internal comparison coefficients (ICC):

$$\text{MC}(\ell_1; \ell_2, k) = \frac{\text{CNPMI}(\ell_1, \ell_2, k)}{\text{INPMI}(\ell_1, k) + \alpha}, \quad (5)$$

$$\text{ICC}(\ell_1, \ell_2, k) = \frac{\text{INPMI}(\ell_1, k) + \alpha}{\text{INPMI}(\ell_2, k) + \alpha}, \quad (6)$$

where α is a smoothing factor ($\alpha = 0.001$ in our experiments). MC recognizes the gap between crosslingual and monolingual coherence, so a higher MC score indicates a gap between coherence within and across languages. Similarly, ICC compares monolingual coherence to tell if both languages are coherent: the closer to 1 the ICC is, the more comparable internal coherence both languages have.

Word Era (ERA) Because the Bible's vocabulary is unable to evaluate modern topics, we must tell the model what the modern words are. The **word era** features are the earliest usage year [3] for each word in a topic. We use both the mean and standard deviation as features.

Meaning Drift (DRIFT). The meaning of a word can expand and drift over time. For example, in the Bible, "web" appears in Isaiah 59:5:

> They hatch cockatrice' eggs, and weave the spider's **web**.

[3] https://oxforddictionaries.com/

The word "web" could be evaluated correctly in an animal topic. For modern topics, however, Bible fails to capture modern meanings of "web", as in Topic 5 (Figure 1).

To address this **meaning drift**, we use a method similar to Hamilton et al. (2016). For each English word, we calculate the context vector from Bible and from Wikipedia with a window size of five and calculate the cosine similarity between them as **word similarity**. Similar context vectors mean that the usage in the Bible is consistent with Wikipedia. We calculate word similarities for all the English topic words in a topic and use the average and standard deviation as features.

3.2 Example

In Figure 3, Topic 1 is coherent while Topic 8 is not. From left to right, we incrementally add new feature sets, and show how the estimated topic coherence scores (dashed lines) approach the ideal CNPMI (dotted lines). When only using the BASE features, the estimator gives a higher prediction to Topic 8 than to Topic 1. Their low MTA and TWC prevent accurate evaluations. Adding GAP does not help much. However, $\text{ICC}(\text{EN}, \text{AM}, k = 1)$ is much smaller, which might indicate a large gap of internal coherence between the two languages.

Adding ERA makes the estimated scores flip between the two topics. Topic 1 has word era of 1823, much older than Topic 8's word era of 1923, in-

Pair	Training	Reference		
		Wikipedia	The Bible	Wiktionary
EN-RO	1,272	8,126	1,189	29,836
EN-SV	3,378	9,067	1,189	42,953
EN-AM	421	1,581	1,189	1,091
EN-TL	542	4,166	1,189	10,970
EN-TR	874	5,524	1,189	16,853
EN-ZH	874	10,000	1,189	22,946

Table 1: Number of document pairs in the training and reference datasets and number of dictionary entries for each language pair.

dicating that Topic 8 includes modern words the Bible lacks (*e.g.*, "computer"). Using all the features, the estimator gives more accurate topic coherence evaluations.

4 Experiments: Bible to Wikipedia

We experiment on six languages (Table 1) from three corpora: Romanian (RO) and Swedish (SV) from EuroParl as representative of well-studied and rich-resource languages (Koehn, 2005); Amharic (AM) and Tagalog (TL) from collected news, as low-resource languages (Huang et al., 2002a,b); and Chinese (ZH) and Turkish (TR) from TED Talks 2013 (Tiedemann, 2012), adding language variety to our experiments. Each language is paired with English as a bilingual corpus.

Typical preprocessing methods (stemming, stop word removal, *etc.*) are often unavailable for low-resource languages. For a meaningful comparison across languages, we do not apply any stemming or lemmatization strategies, including English, except removing digit numbers and symbols. However, we remove words that appear in more than 30% of documents for each language.

Each language pair is separately trained using the `MALLET` (McCallum, 2002) implementation of the polylingual topic model. Each experiment runs five Gibbs sampling chains with 1,000 iterations per chain with twenty topics. The hyperparameters are set to the default values ($\alpha = 0.1$, $\beta = 0.01$), and are optimized every 50 iterations in `MALLET` using slice sampling (Wallach et al., 2009a).

4.1 Evaluating Multilingual Topics

We use Wikipedia and the Bible as reference corpora for calculating co-occurrence statistics. Different numbers of Wikipedia articles are available for each language pair (Table 1), while the Bible contains a complete set of 1,189 chapters for all of its translations (Christodoulopoulos and Steedman, 2015). We use Wiktionary as the dictionary to calculate MTA.

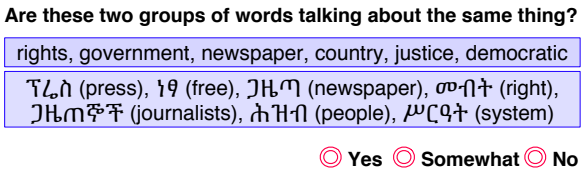


Figure 4: The interface for topic quality judgments. Users read the topic first, and make a judgment on whether the words in this pair are talking about the same thing. The translations are here for illustration; they are not shown to the users.

	Wikipedia		The Bible		MTA
	CNPMI	INPMI	CNPMI	INPMI	
EN-RO	0.490	0.118	−0.096	0.031	**0.592**
EN-SV	**0.453**	−0.295	0.164	−0.351	0.248
EN-AM	0.110	0.019	**0.289**	0.249	0.172
EN-TL	**0.512**	0.277	0.166	0.002	0.289
EN-TR	0.664	0.243	0.209	−0.246	**0.677**
EN-ZH	**0.436**	0.297	0.274	0.157	0.411

Table 2: Pearson correlations between human judgments and CNPMI are higher than INPMI, while MTA correlations are comparable to CNPMI.

4.2 Training the Estimator

In addition to experimenting on Wikipedia-based CNPMI, we also re-evaluate the topics' Bible coherence using our estimator. In the following experiments, we use an AdaBoost regressor with linear regression as the coherence estimator (Friedman, 2002; Collins et al., 2000). The estimator takes a topic and low-quality CNPMI score as input and outputs (hopefully) an improved CNPMI score.

To make our testing scenario more realistic, we treat one language as our estimator's test language and train on multilingual topics from the other languages. We use three-fold cross-validation over languages to select the best hyperparameters, including the learning rate and loss function in `AdaBoost.R2` (Drucker, 1997).

5 Topic-Level Evaluation

We first study CNPMI at the topic level: does a particular topic make sense? An effective evaluation should be consistent with human judgment of the topics (Chang et al., 2009). In this section, we measure gold-standard human interpretability of multilingual topics to establish which automatic measures of topic interpretability work best.

Test	Bible	Train		
		RO+SV	ZH+TR	RO+SV+ZH+TR
AM	−0.015	0.332	0.315	0.333
TL	−0.309	0.767	0.631	0.705
		AM+TL	ZH+TR	AM+TL+ZH+TR
RO	−0.269	0.736	0.681	0.713
SV	0.000	0.787	0.645	0.683
		RO+SV	AM+TL	RO+SV+AM+TL
ZH	0.217	0.751	0.732	0.741
TR	0.113	0.680	0.642	0.666

Table 3: Correlations between the Wikipedia-based CNPMI and the Bible-based CNPMI, before and after using the coherence estimator, at the topic level. Strong correlations indicate that the estimator improves CNPMI estimates.

5.1 Task Design

Following monolingual coherence evaluations (Lau et al., 2014), we present topic pairs to bilingual CrowdFlower users. Each task is a topic pair with the top ten topic words ($C = 10$) for each language. We ask if both languages' top words in a multilingual topic are talking about the same concept (Figure 4), and make a judgment on a three-point scale—coherent (2 points), somewhat coherent (1 point), and incoherent (0 points). To ensure the users have adequate language competency, we insert several topics that are easily identifiable as incoherent as a qualification test.

We randomly select sixty topics from each language pair (360 topics total), and each topic is judged by five users. We take the average of the judgment points and calculate Pearson correlations with the proposed evaluation metrics (Table 2). NPMI-based scores are separately calculated from each reference corpus.

5.2 Agreement with Human Judgments

CNPMI (the extended metric) has higher correlations with human judgments than INPMI (the naive adaptation of monolingual NPMI), while MTA (matching translation accuracy) correlations are comparable to CNPMI.

Unsurprisingly, when using Wikipedia as the reference, the correlations are usually higher than when using the Bible. The Bible's archaic content limits its ability to estimate human judgments in modern corpora (Section 3).

Next, we compare CNPMI to two baselines: INPMI and MTA. As expected, CNPMI outperforms INPMI regardless of reference corpus overall, because INPMI only considers monolingual coherence. MTA has higher correlations than CNPMI scores from the Bible, because the Bible fails to give accurate estimates due to limited topic coverage. MTA, on the other hand, only depends on dictionaries, which are more comprehensive than the Bible. It is also possible that users are judging coherence based on translations across a topic pair, rather than the overall coherence, which would closely correlate with MTA.

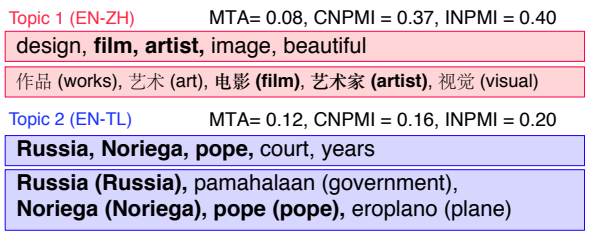


Figure 5: MTA fails to capture semantically related words (Topic 1) and only looks at translation pairs regardless of internal coherence (Topic 2).

5.3 Re-Estimating Topic-Level Coherence

The Bible—by itself—produces CNPMI values that do not correlate well with human judgments (Table 2). After training an estimator (Section 4.2), we calculate Pearson's correlation between Wikipedia's CNPMI and the estimated topic coherence score (Table 3). A higher correlation with Wikipedia's CNPMI means more accurate coherence.

As a baseline, the correlation of Bible-based CNPMI without adaptation has negative and near-zero correlations with Wikipedia;[4] it does not capture coherence. After training the estimator, the correlations become stronger, indicating the estimated scores are closer to Wikipedia's CNPMI.

5.4 When MTA Falls Short

We analyze MTA from two aspects—the inability to capture semantically-related *non-translation* topic words, and insensitivity to cardinality—to show why MTA is not an ideal measurement, even though it correlates well with human judgments.

Semantics We take two examples with EN-ZH (Topic 1) and EN-TL (Topic 2) in Figure 5. Topic 1 has fewer translation pairs than Topic 2, which leads to a lower MTA score for Topic 1. However, all words in Topic 1 talk about art, while it is hard to interpret Topic 2. Wikipedia CNPMI scores reveals

[4]Normally one would not estimate CNPMI on rich-resource languages using low-resource languages. For completeness, however, we also include these situations.

Topic 1 is more coherent. Because our experiments are on datasets with little divergence between the themes discussed across languages, this is uncommon for us but could appear in noisier datasets.

Cardinality Increasing cardinality diminishes a topic's coherence (Lau and Baldwin, 2016). We vary the cardinality of topics from ten to fifty at intervals of ten (Figure 6). As cardinality increases, more low-probability and irrelevant words appear the topic, which lowers CNPMI scores. However, MTA stays stable or increases with increasing cardinality. Thus, MTA fails to fulfill a critical property of topic model evaluation.

Finally, MTA requires a comprehensive multilingual dictionary, which may be unavailable for low-resource languages. Additionally, most languages often only have one dictionary, which makes it problematic to use the same resource (a language's single multilingual dictionary) for training and evaluating models that use a dictionary to build multilingual topics (Hu et al., 2014). Given these concerns, we continue the paper's focus on CNPMI as a data-driven alternative to MTA. However, for many applications MTA may suffice as a simple, adequate evaluation metric.

6 Model-Level Evaluation

While the previous section looked at individual topics, we also care about how well CNPMI characterizes the quality of *models* through an average of a model's constituent topics.

6.1 Training Knowledge

Adding more knowledge to multilingual topic models improves topics (Hu et al., 2014), so an effective evaluation should reflect this improvement as knowlege is added to the model. For polylingual topic models, this knowledge takes the form of the *number* of linked documents.

We start by experimenting with no multilingual knowledge: no document pairs share a topic distribution θ_d (but the documents are in the collection as unlinked documents). We then increase the number of document pairs that share θ_d from 20% of the corpus to 100%. Fixing the topic cardinality at ten, CNPMI captures the improvements in models (Figure 7) through a higher coherence score.

6.2 Agreement with Machines

Topic models are often used as a feature extraction technique for downstream machine learning applications, and topic model evaluations should reflect whether these features are useful (Ramage et al., 2009). For each model, we apply a document classifier trained on the model parameters to test whether CNPMI is consistent with classification accuracy.

Test	Bible	Train		
		RO+SV	ZH+TR	RO+SV+ZH+TR
AM	0.607	0.677	0.707	0.694
TL	0.796	0.875	0.924	0.918
		AM+TL	ZH+TR	AM+TL+ZH+TR
RO	0.631	0.912	0.919	0.931
SV	0.797	0.959	0.848	0.878
		RO+SV	AM+TL	RO+SV+AM+TL
ZH	0.907	0.918	0.951	0.939
TR	0.911	0.862	0.898	0.887

Table 4: At the model level, the estimator improves correlations between CNPMI and downstream classification for all languages except for Turkish.

Specifically, we want our classifier to transfer information from training on one language to testing on another (Smet et al., 2011; Heyman et al., 2016). We train a classifier on one language's documents, where each document's feature vector is the document-topic distribution θ_d. We apply this to TED Talks, where each document is labeled with multiple categories. We choose the most frequent seven categories across the corpus as labels,[5] and only have labeled documents in one side of a bilingual topic model. CNPMI has very strong correlations with classification results, though using the Bible as the reference corpus gives slightly lower correlation—with higher variance—than Wikipedia (Figure 8).

6.3 Re-Estimating Model-Level Coherence

In Section 5.3, we improve Bible-based CNPMI scores for individual topics. Here, we show the estimator also improves model-level coherence. We apply the estimator on the models created in Section 6.2 and calculate the correlation between estimated scores and Wikipedia's CNPMI (Table 4).

The coherence estimator substantially improves scores except for Turkish: the correlation is better *before* applying the estimator (0.911). We suspect a lack of overlap between topics between Turkish and languages other than Chinese is to blame (Figure 9); the features used by the estimator do not generalize well to other kinds of features; training on many languages pairs would hopefully solve this

[5] design, global issues, art, science, technology, business, and culture

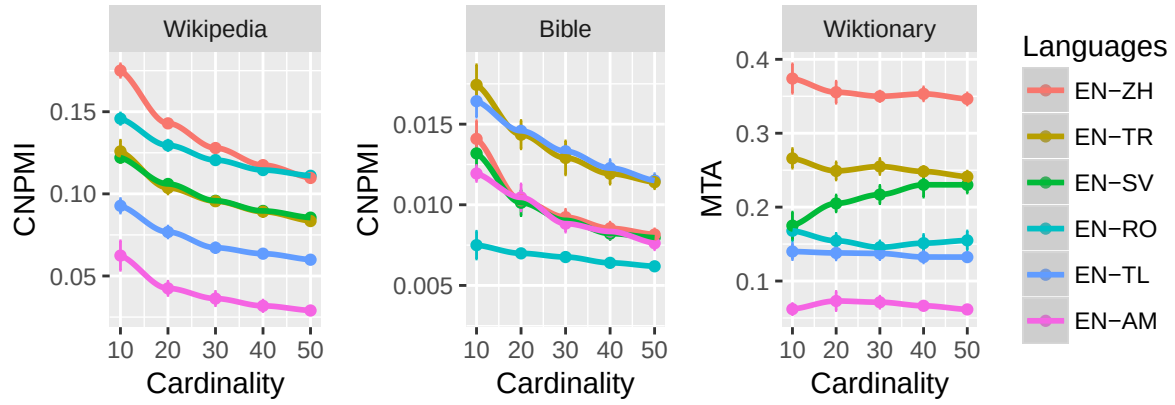


Figure 6: Increasing cardinality of topic pairs makes it harder to judge the coherence. Decreasing CNPMI scores reflect the diminished interpretability of topics, while MTA scores do not.

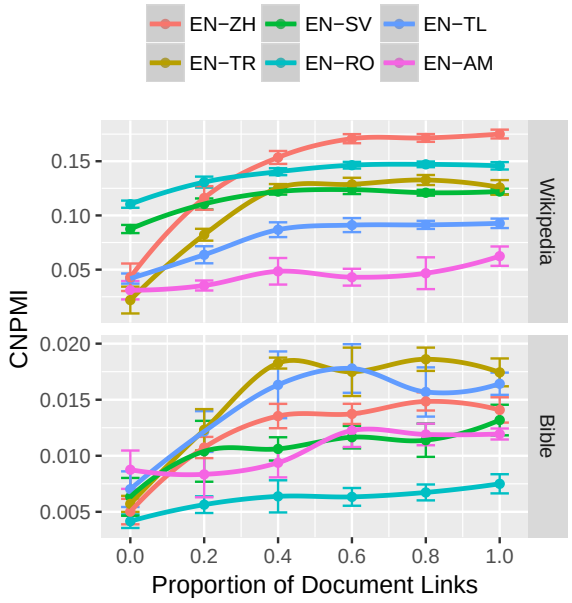


Figure 7: Adding more document links to the model produces more multilingually coherent topics. CNPMI captures this improvement.

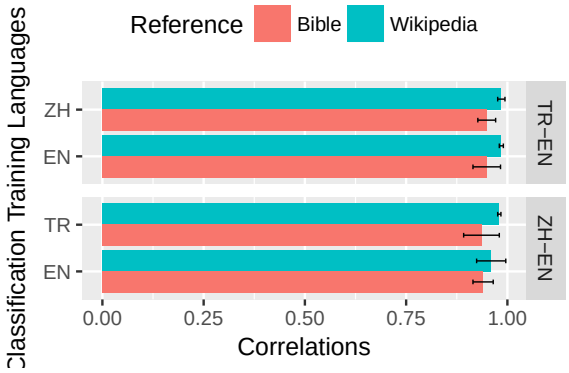


Figure 8: Pearson correlation between classification F1 scores and CNPMI: both CNPMI data sources predict whether a classifier using topic features will work well, but Wikipedia has slightly higher correlation with lower variance.

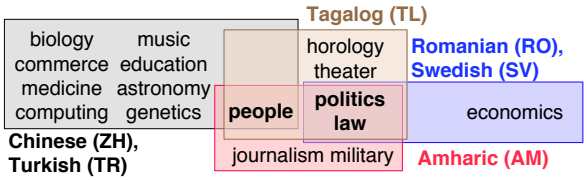


Figure 9: The overlap of topics and domain: only one out of nine Turkish and Chinese topics have domain overlap with Tagalog and Amharic topics. This hinders the Turkish estimator from capturing model-level properties.

issue. Turkish is also morphologically rich, and our preprocessing completely ignores morphology.

6.4 Reference Size

One challenge with low-resource languages is that even if Wikipedia is available, it may have too few documents to accurately calculate coherence. As a final analysis, we examine how the reliability of CNPMI degrades with a smaller reference corpus.

We randomly sample 20% to 100% of document pairs from the reference corpora and evaluate the polylingual topic model with all document links (Figure 10), again fixing the cardinality as 10.

CNPMI is stable across different amounts of reference documents, as long as the number of reference documents is sufficiently large. If there are too few reference documents (for example, 20% of Amharic Wikipedia is only 316 documents), then CNPMI degrades.

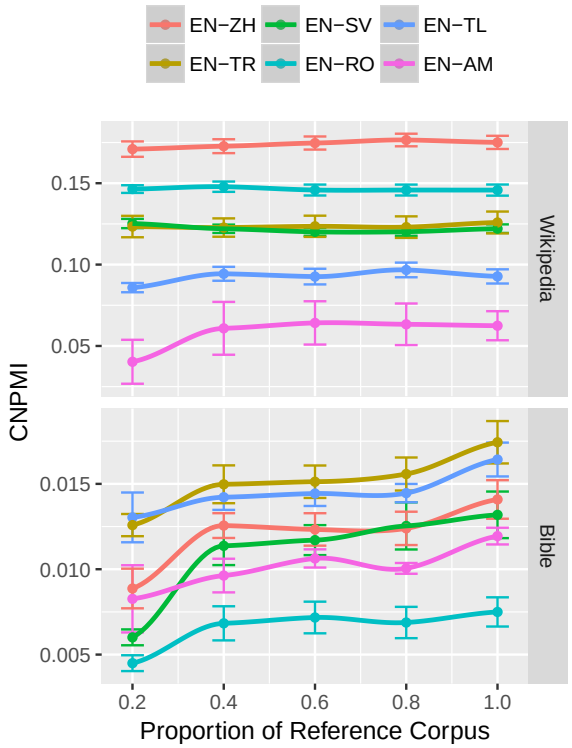


Figure 10: CNPMI is stable once the number of reference documents is large enough (around five thousand documents).

7 Related Work

Topic Coherence Many coherence metrics based on co-occurrence statistics have been proposed besides NPMI. Similar metrics—such as asymmetrical word pair metrics (Mimno et al., 2011) and combinations of existing measurements (Lau et al., 2014; Röder et al., 2015)—correlate well with human judgments. NPMI has been the current gold standard for evaluation and improvements of monolingual topic models (Pecina, 2010; Newman et al., 2011).

External Tasks Another approach is to use a model for predictive tasks: the better the results are on external tasks, the better a topic model is assumed to be. A common task is held-out likelihood (Wallach et al., 2009b; Jagarlamudi and Daumé III, 2010; Fukumasu et al., 2012), but as Chang et al. (2009) show, this does not always reflect human interpretability. Other specific tasks have also been used, such as bilingual dictionary extraction (Liu et al., 2015; Ma and Nasukawa, 2017), cultural difference deteciton (Gutiérrez et al., 2016), and crosslingual document clustering (Vulić et al., 2015).

Representation Learning Topic models are one example of a broad class of techniques of learning representations of documents (Bengio et al., 2013). Other approaches learn respresentations at the word (Klementiev et al., 2012; Vyas and Carpuat, 2016), paragraph (Mogadala and Rettinger, 2016), or corpus level (Søgaard et al., 2015). However, neural representation learning approaches are often data hungry and not adaptable to low-resource languages. The approaches here could help improve the evaluation of all multilingual representation learning algorithms (Schnabel et al., 2015).

8 Conclusion

We have provided a comprehensive analysis of topic model evaluation in multilingual settings, including for low-resource languages. While evaluation is an important area of topic model research, no previous work has studied evaluation of multilingual topic models. Our work provided two primary contributions to this area, including a new intrinsic evaluation metric, CNPMI, as well as a model for adapting this metric to low-resource languages without large reference corpora.

As the first study on evaluation for multilingual topic models, there is still room for improvement and further applications. For example, human judgment is more difficult to measure than in monolingual settings, and it is still an open question on how to design a reliable and accurate survey for multilingual quality judgments. As a measurement of multilingual coherence, we plan to extend CNPMI to high-dimensional representations, *e.g.,* multilingual word embeddings, particularly in low-resource languages (Ruder et al., 2017).

Acknowledgement

We thank the anonymous reviewers for their insightful and constructive comments. Hao has been supported under subcontract to Raytheon BBN Technologies, by DARPA award HR0011-15-C-0113. Boyd-Graber and Paul were supported by NSF grant IIS-1564275. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsors.

References

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new

perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* 35(8):1798–1828. https://doi.org/10.1109/TPAMI.2013.50.

David M. Blei, Andrew Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research* 3:993–1022.

Gerlof Bouma. 2009. Normalized (oointwise) mutual information in collocation dxtraction. In *Proceedings of the German Society for Computational Linguistics and Language Technology Conference*.

Jordan Boyd-Graber and David M. Blei. 2009. Multilingual topic models for unaligned text. In *Proceedings of Uncertainty in Artificial Intelligence*.

Jordan Boyd-Graber, Yuening Hu, and David Mimno. 2017. *Applications of Topic Models*, volume 11 of *Foundations and Trends in Information Retrieval*. NOW Publishers. http://www.nowpublishers.com/article/Details/INR-030.

Jonathan Chang, Jordan Boyd-Graber, Sean Gerrish, Chong Wang, and David M. Blei. 2009. Reading tea leaves: how humans interpret topic models. In *Proceedings of Advances in Neural Information Processing Systems*.

Christos Christodoulopoulos and Mark Steedman. 2015. A massively parallel corpus: the Bible in 100 languages. *Language Resources and Evaluation* 49(2):375–395.

Michael Collins, Robert E. Schapire, and Yoram Singer. 2000. Logistic regression, AdaBoost and Bregman distances. In *Proceedings of Conference on Learning Theory*.

Harris Drucker. 1997. Improving regressors using boosting techniques. In *Proceedings of the International Conference of Machine Learning*.

Jerome H Friedman. 2002. Stochastic gradient boosting. *Computational Statistics & Data Analysis* 38(4):367–378.

Kosuke Fukumasu, Koji Eguchi, and Eric P. Xing. 2012. Symmetric correspondence topic models for multilingual text analysis. In *Proceedings of Advances in Neural Information Processing Systems*.

E. Dario Gutiérrez, Ekaterina Shutova, Patricia Lichtenstein, Gerard de Melo, and Luca Gilardi. 2016. Detecting cross-cultural differences using a multilingual topic model. *Transactions of the Association for Computational Linguistics* 4:47–60.

William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Cultural shift or linguistic drift? Comparing two computational measures of semantic change. In *Proceedings of Empirical Methods in Natural Language Processing*.

Geert Heyman, Ivan Vulic, and Marie-Francine Moens. 2016. C-BiLDA: Extracting cross-lingual topics from non-parallel texts by distinguishing shared from unshared content. *Data Mining and Knowledge Discovery* 30(5).

Yuening Hu, Ke Zhai, Vladimir Eidelman, and Jordan L. Boyd-Graber. 2014. Polylingual tree-based topic models for translation domain adaptation. In *Proceedings of the Association for Computational Linguistics*.

Shudong Huang, David Graff, and George Doddington. 2002a. NMSU Amharic language pack from REFLEX V1.1. Web download file. Philadelphia: Linguistic Data Consortium.

Shudong Huang, David Graff, and George Doddington. 2002b. Tagalog language pack from reflex V1.1. Web download file. Philadelphia: Linguistic Data Consortium.

Jagadeesh Jagarlamudi and Hal Daumé III. 2010. Extracting multilingual topics from unaligned comparable corpora. In *Proceedings of the European Conference on Information Retrieval*.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *Proceedings of International Conference on Computational Linguistics*.

Philipp Koehn. 2005. Europarl: a parallel corpus for statistical machine translation.

Kriste Krstovski and David A. Smith. 2016. Bootstrapping translation detection and sentence extraction from comparable corpora. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Kriste Krstovski, David A. Smith, and Michael J. Kurtz. 2016. Online multilingual topic models with multilevel hyperpriors. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Jey Han Lau and Timothy Baldwin. 2016. The sensitivity of topic coherence evaluation to topic cardinality. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Jey Han Lau, David Newman, and Timothy Baldwin. 2014. Machine reading tea leaves: automatically evaluating topic coherence and topic model quality. In *Proceedings of the European Chapter of the Association for Computational Linguistics*.

Xiaodong Liu, Kevin Duh, and Yuji Matsumoto. 2015. Multilingual topic models for bilingual dictionary extraction. *ACM Transactions on Asian & Low-Resource Language Information Processing* 14(3):11:1–11:22.

Tengfei Ma and Tetsuya Nasukawa. 2017. Inverted bilingual topic models for lexicon extraction from non-parallel data. In *International Joint Conference on Artificial Intelligence*.

Andrew Kachites McCallum. 2002. MALLET: a machine learning for language toolkit. `http://mallet.cs.umass.edu`.

David M. Mimno, Hanna M. Wallach, Jason Naradowsky, David A. Smith, and Andrew McCallum. 2009. Polylingual topic models. In *Proceedings of Empirical Methods in Natural Language Processing*.

David M. Mimno, Hanna M. Wallach, Edmund M. Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Proceedings of Empirical Methods in Natural Language Processing*.

Aditya Mogadala and Achim Rettinger. 2016. Bilingual word embeddings from parallel and non-parallel corpora for cross-language text classification. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

David Newman, Edwin V. Bonilla, and Wray L. Buntine. 2011. Improving topic coherence with regularized topic models. In *Proceedings of Advances in Neural Information Processing Systems*.

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010. Automatic evaluation of topic coherence. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Xiaochuan Ni, Jian-Tao Sun, Jian Hu, and Zheng Chen. 2009. Mining multilingual topics from Wikipedia. In *Proceedings of the World Wide Web Conference*.

Pavel Pecina. 2010. Lexical association measures and collocation extraction. *Proceedings of the Language Resources and Evaluation Conference* 44(1-2).

Daniel Ramage, David Leo Wright Hall, Ramesh Nallapati, and Christopher D. Manning. 2009. Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of Empirical Methods in Natural Language Processing*.

Philip Resnik, Mari Broman Olsen, and Mona Diab. 1999. The Bible as a parallel corpus: annotating the 'book of 2000 tongues'. *Computers and the Humanities* 33(1/2):129–153.

Michael Röder, Andreas Both, and Alexander Hinneburg. 2015. Exploring the space of topic coherence measures. In *Proceedings of ACM International Conference on Web Search and Data Mining*.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2017. A survey of cross-lingual word embedding models. *CoRR* abs/1706.04902.

Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 298–307.

Wim De Smet, Jie Tang, and Marie-Francine Moens. 2011. Knowledge transfer across multilingual corpora via latent topics. In *Pacific-Asia Advances in Knowledge Discovery and Data Mining*.

Anders Søgaard, Željko Agić, Héctor Martínez Alonso, Barbara Plank, Bernd Bohnet, and Anders Johannsen. 2015. Inverted indexing for cross-lingual nlp. In *Proceedings of the Association for Computational Linguistics*.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the Language Resources and Evaluation Conference*.

Ivan Vulić, Wim De Smet, and Marie-Francine Moens. 2013. Cross-language information retrieval models based on latent topic models trained with document-aligned comparable corpora. *Information Retrieval* 16(3):331–368.

Ivan Vulić, Wim De Smet, Jie Tang, and Marie-Francine Moens. 2015. Probabilistic topic modeling in multilingual settings: an overview of its methodology and applications. *Information Processing & Management* 51(1):111–147.

Yogarshi Vyas and Marine Carpuat. 2016. Sparse bilingual word representations for cross-lingual lexical entailment. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Hanna Wallach, David Mimno, and Andrew McCallum. 2009a. Rethinking LDA: Why priors matter. In *Proceedings of Advances in Neural Information Processing Systems*.

Hanna M. Wallach, Iain Murray, Ruslan Salakhutdinov, and David M. Mimno. 2009b. Evaluation methods for topic models. In *Proceedings of the International Conference of Machine Learning*.

Weiwei Yang, Jordan L. Boyd-Graber, and Philip Resnik. 2017. Adapting topic models using lexical associations with tree priors. In *Proceedings of Empirical Methods in Natural Language Processing*.

Explainable Prediction of Medical Codes from Clinical Text

James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, Jacob Eisenstein
Georgia Institute of Technology
{jmullenbach3, swiegreffe6, jon.duke}@gatech.edu
jsun@cc.gatech.edu, jacobe@gatech.edu

Abstract

Clinical notes are text documents that are created by clinicians for each patient encounter. They are typically accompanied by medical codes, which describe the diagnosis and treatment. Annotating these codes is labor intensive and error prone; furthermore, the connection between the codes and the text is not annotated, obscuring the reasons and details behind specific diagnoses and treatments. We present an attentional convolutional network that predicts medical codes from clinical text. Our method aggregates information across the document using a convolutional neural network, and uses an attention mechanism to select the most relevant segments for each of the thousands of possible codes. The method is accurate, achieving precision@8 of 0.71 and a Micro-F1 of 0.54, which are both better than the prior state of the art. Furthermore, through an interpretability evaluation by a physician, we show that the attention mechanism identifies meaningful explanations for each code assignment.

1 Introduction

Clinical notes are free text narratives generated by clinicians during patient encounters. They are typically accompanied by a set of metadata codes from the International Classification of Diseases (ICD), which present a standardized way of indicating diagnoses and procedures that were performed during the encounter. ICD codes have a variety of uses, ranging from billing to predictive modeling of patient state (Choi et al., 2016; Ranganath et al., 2015; Denny et al., 2010; Avati et al., 2017). Because manual coding is time-consuming and error-prone, automatic coding has been studied since at least the 1990s (de Lima et al., 1998). The task is difficult for two main reasons. First, the label space is very high-dimensional, with over 15,000 codes in the ICD-9 taxonomy, and over 140,000 codes combined in the newer ICD-10-CM and ICD-10-PCS taxonomies (World Health Organization, 2016). Second, clinical text includes irrelevant information, misspellings and non-standard abbreviations, and a large medical vocabulary. These features combine to make the prediction of ICD codes from clinical notes an especially difficult task, for computers and human coders alike (Birman-Deych et al., 2005).

In this application paper, we develop convolutional neural network (CNN)-based methods for automatic ICD code assignment based on text discharge summaries from intensive care unit (ICU) stays. To better adapt to the multi-label setting, we employ a per-label attention mechanism, which allows our model to learn distinct document representations for each label. We call our method **C**onvolutional **A**ttention for **M**ulti-**L**abel classification (CAML). Our model design is motivated by the conjecture that important information correlated with a code's presence may be contained in short snippets of text which could be anywhere in the document, and that these snippets likely differ for different labels. To cope with the large label space, we exploit the textual descriptions of each code to guide our model towards appropriate parameters: in the absence of many labeled examples for a given code, its parameters should be similar to those of codes with similar textual descriptions.

We evaluate our approach on two versions of MIMIC (Johnson et al., 2016), an open dataset of ICU medical records. Each record includes a variety of narrative notes describing a patient's stay, including diagnoses and procedures. Our approach substantially outperforms previous results on medical code prediction on both MIMIC-II and MIMIC-III datasets.

We consider applications of this work in a decision support setting. Interpretability is important for any decision support system, especially in the

Proceedings of NAACL-HLT 2018, pages 1101–1111
New Orleans, Louisiana, June 1 - 6, 2018.

934.1: "Foreign body in main bronchus"

CAML (HI)	*...line placed bronchoscopy performed showing* **large mucus plug on** *the left on transfer to...*
Cosine Sim	*...also needed medication to help* **your body maintain your** *blood pressure after receiving iv...*
CNN	*...found to have a large* **lll lingular pneumonia on** *chest x ray he was...*
Logistic Regression	*...impression confluent consolidation involving nearly* **the entire left lung** *with either broncho-centric or vascular...*

442.84: "Aneurysm of other visceral artery"

CAML (I)	*...and gelfoam embolization of right* **hepatic artery branch pseudoaneurysm** *coil embolization of the gastroduodenal...*
Cosine Sim	*...coil embolization of the gastroduodenal* **artery history of present** *illness the pt is a...*
CNN	*...foley for hemodynamic monitoring and* **serial hematocrits angio was** *performed and his gda was...*
Logistic Regression (I)	*...and gelfoam embolization of right* **hepatic artery branch pseudoaneurysm** *coil embolization of the gastroduodenal...*

428.20: "Systolic heart failure, unspecified"

CAML	*...no mitral valve prolapse moderate* **to severe mitral regurgitation** *is seen the tricuspid valve...*
Cosine Sim	*...is seen the estimated pulmonary* **artery systolic pressure is** *normal there is no pericardial...*
CNN	*...and suggested starting hydralazine imdur* **continue aspirin arg admitted** *at baseline cr appears patient...*
Logistic Regression (HI)	*...anticoagulation monitored on tele pump* **systolic dysfunction with ef** *of seen on recent echo...*

Table 1: Presentation of example qualitative evaluations. In real evaluation, system names generating the 4-gram are not given. An 'I' marking indicates a snippet evaluated as informative, and 'HI' indicates that it is highly informative; see § 4 for more details.

medical domain. The system should be able to explain why it predicted each code; even if the codes are manually annotated, it is desirable to explain what parts of the text are most relevant to each code. These considerations further motivate our per-label attention mechanism, which assigns importance values to n-grams in the input document, and which can therefore provide explanations for each code, in the form of extracted snippets of text from the input document. We perform a human evaluation of the quality of the explanations provided by the attention mechanism, asking a physician to rate the informativeness of a set of automatically generated explanations.[1]

2 Method

We treat ICD-9 code prediction as a multilabel text classification problem (McCallum, 1999).[2] Let $\mathcal{L}$ represent the set of ICD-9 codes; the labeling problem for instance i is to determine $y_{i,\ell} \in \{0, 1\}$ for all $\ell \in \mathcal{L}$. We train a neural network which passes text through a convolutional layer to compute a base representation of the text of each document (Kim, 2014), and makes $|\mathcal{L}|$ binary classification decisions. Rather than aggregating across this representation with a pooling operation, we apply an attention mechanism to select the parts of the document that are most relevant for each possible code. These attention weights are then applied to the base representation, and the result is passed through an output layer, using a sigmoid transformation to compute the likelihood of each code. We employ a regularizer to encourage each code's parameters to be similar to those of codes with similar textual descriptions. We now describe each of these elements in more detail.

2.1 Convolutional architecture

At the base layer of the model, we have d_e-dimensional pre-trained embeddings for each word in the document, which are horizontally concatenated into the matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N]$, where N is the length of the document. Adjacent word embeddings are combined using a convolutional filter $\boldsymbol{W}_c \in \mathbb{R}^{k \times d_e \times d_c}$, where k is the filter width, d_e the size of the input embedding, and d_c the size of the filter output. At each step n, we compute

$$\boldsymbol{h}_n = g(\boldsymbol{W}_c * \boldsymbol{x}_{n:n+k-1} + \boldsymbol{b}_c), \quad (1)$$

where $*$ denotes the convolution operator, g is an element-wise nonlinear transformation, and $\boldsymbol{b}_c \in \mathbb{R}^{d_c}$ is the bias. We additionally pad each side of

[1] Our code, data splits, and pre-trained models are available at github.com/jamesmullenbach/caml-mimic.

[2] We focus on codes from the ICD-9 taxonomy, rather than the more recent ICD-10, for the simple reason that this is the version of ICD used in the MIMIC datasets.

the input with zeros so that the resulting matrix $\boldsymbol{H}$ has dimension $\mathbb{R}^{d_c \times N}$.

2.2 Attention

After convolution, the document is represented by the matrix $\boldsymbol{H} \in \mathbb{R}^{d_c \times N}$. It is typical to reduce this matrix to a vector by applying pooling across the length of document, by selecting the maximum or average value at each row (Kim, 2014). However, our goal is to assign multiple labels (i.e., medical codes) for each document, and different parts of the base representation may be relevant for different labels. For this reason, we apply a per-label attention mechanism. An additional benefit is that it selects the k-grams from the text that are most relevant to each predicted label.

Formally, for each label ℓ, we compute the matrix-vector product, $\boldsymbol{H}^\top \boldsymbol{u}_\ell$, where $\boldsymbol{u}_\ell \in \mathbb{R}^{d_c}$ is a vector parameter for label ℓ. We then pass the resulting vector through a softmax operator, obtaining a distribution over locations in the document,

$$\boldsymbol{\alpha}_\ell = \text{SoftMax}(\boldsymbol{H}^\top \boldsymbol{u}_\ell), \tag{2}$$

where $\text{SoftMax}(\boldsymbol{x}) = \frac{\exp(\boldsymbol{x})}{\sum_i \exp(x_i)}$, and $\exp(\boldsymbol{x})$ is the element-wise exponentiation of the vector $\boldsymbol{x}$. The attention vector $\boldsymbol{\alpha}$ is then used to compute vector representations for each label,

$$\boldsymbol{v}_\ell = \sum_{n=1}^{N} \alpha_{\ell,n} \boldsymbol{h}_n. \tag{3}$$

As a baseline model, we instead use max-pooling to compute a single vector $\boldsymbol{v}$ for all labels,

$$v_j = \max_n h_{n,j}. \tag{4}$$

2.3 Classification

Given the vector document representation $\boldsymbol{v}_\ell$, we compute a probability for label ℓ using another linear layer and a sigmoid transformation:

$$\hat{y}_\ell = \sigma(\boldsymbol{\beta}_\ell^\top \boldsymbol{v}_\ell + b_\ell), \tag{5}$$

where $\boldsymbol{\beta}_\ell \in \mathbb{R}^{d_c}$ is a vector of prediction weights, and b_ℓ is a scalar offset. The overall model is illustrated in Figure 1.

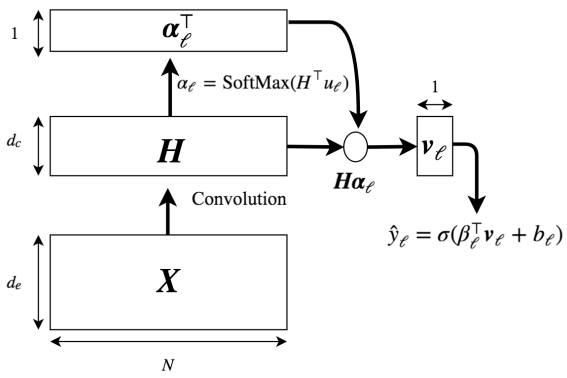


Figure 1: CAML architecture with per-label attention shown for one label. In a max-pooling architecture, $\boldsymbol{H}$ is mapped directly to the vector $\boldsymbol{v}_\ell$ by maximizing over each dimension.

2.4 Training

The training procedure minimizes the binary cross-entropy loss,

$$L_{\text{BCE}}(\boldsymbol{X}, \boldsymbol{y}) = -\sum_{\ell=1}^{\mathcal{L}} y_\ell \log(\hat{y}_\ell) + (1 - y_\ell) \log(1 - \hat{y}_\ell), \tag{6}$$

plus the L2 norm of the model weights, using the Adam optimizer (Kingma and Ba, 2015).

2.5 Embedding label descriptions

Due to the dimensionality of the label space, many codes are rarely observed in the labeled data. To improve performance on these codes, we use text descriptions of each code from the World Health Organization (2016). Examples can be found in Table 1, next to the code numbers. We use these descriptions to build a secondary module in our network that learns to embed them as vectors. These vectors are then used as the target of regularization on the model parameters $\boldsymbol{\beta}_\ell$. If code ℓ is rarely observed in the training data, this regularizer will encourage its parameters to be similar to those of other codes with similar descriptions.

The code embedding module consists of a max-pooling CNN architecture. Let $\boldsymbol{z}_\ell$ be a max-pooled vector, obtained by passing the description for code ℓ into the module. Let n_y be the number of true labels in a training example. We add the following regularizing objective to our loss L,

$$L(\boldsymbol{X}, \boldsymbol{y}) = L_{\text{BCE}} + \lambda \frac{1}{n_y} \sum_{\ell : y_\ell = 1}^{\mathcal{L}} \|\boldsymbol{z}_\ell - \boldsymbol{\beta}_\ell\|_2, \tag{7}$$

where λ is a tradeoff hyperparameter that calibrates the performance of the two objectives. We call this model variant Description Regularized-CAML (DR-CAML).

3 Evaluation of code prediction

This section evaluates the accuracy of code prediction, comparing our models against several competitive baselines.

3.1 Datasets

MIMIC-III (Johnson et al., 2016) is an open-access dataset of text and structured records from a hospital ICU. Following previous work, we focus on discharge summaries, which condense information about a stay into a single document. In MIMIC-III, some admissions have addenda to their summary, which we concatenate to form one document.

Each admission is tagged by human coders with a set of ICD-9 codes, describing both diagnoses and procedures which occurred during the patient's stay. There are 8,921 unique ICD-9 codes present in our datasets, including 6,918 diagnosis codes and 2,003 procedure codes. Some patients have multiple admissions and therefore multiple discharge summaries; we split the data by patient ID, so that no patient appears in both the training and test sets.

In this full-label setting, we use a set of 47,724 discharge summaries from 36,998 patients for training, with 1,632 summaries and 3,372 summaries for validation and testing, respectively.

Secondary evaluations For comparison with prior work, we also follow Shi et al. (2017) and train and evaluate on a label set consisting of the 50 most frequent labels. In this setting, we filter each dataset down to the instances that have at least one of the top 50 most frequent codes, and subset the training data to equal the size of the training set of Shi et al. (2017), resulting in 8,067 summaries for training, 1,574 for validation, and 1,730 for testing.

We also run experiments with the MIMIC-II dataset, to compare with prior work by Baumel et al. (2018) and Perotte et al. (2013). We use the train/test split of Perotte et al. (2013), which consists of 20,533 training examples and 2,282 testing examples. Detailed statistics for the three settings are summarized in Table 2.

Preprocessing We remove tokens that contain no alphabetic characters (e.g., removing "500" but keeping "250mg"), lowercase all tokens, and replace tokens that appear in fewer than three training documents with an 'UNK' token. We pretrain word embeddings of size $d_e = 100$ using the word2vec CBOW method (Mikolov et al., 2013) on the preprocessed text from all discharge summaries. All documents are truncated to a maximum length of 2500 tokens.

3.2 Systems

We compare against the following baselines:

- a single-layer one-dimensional convolutional neural network (Kim, 2014);
- a bag-of-words logistic regression model;
- a bidirectional gated recurrent unit (Bi-GRU).[3]

For the CNN and Bi-GRU, we initialize the embedding weights using the same pretrained word2vec vectors that we use for the CAML models. All neural models are implemented using PyTorch[4]. The logistic regression model consists of $|\mathcal{L}|$ binary one-vs-rest classifiers acting on unigram bag-of-words features for all labels present in the training data. If a label is not present in the training data, the model will never predict it in the held-out data.

Parameter tuning We tune the hyperparameters of the CAML model and the neural baselines using the Spearmint Bayesian optimization package (Snoek et al., 2012; Swersky et al., 2013).[5] We allow Spearmint to sample parameter values for the L2 penalty on the model weights ρ and learning rate η, as well as filter size k, number of filters d_c, and dropout probability q for the convolutional models, and number of hidden layers s of dimension v for the Bi-GRU, using precision@8 on the MIMIC-III full-label validation set as the performance measure. We use these parameters for DR-CAML as well, and port the optimized parameters to the MIMIC-II full-label and MIMIC-III 50-label models, and manually fine-tune the learning rate in these settings. We select λ for DR-CAML based on pilot experiments on the validation sets. Hyperparameter tuning is summarized in Table 3. Convolutional models are trained with dropout after the

[3] Our pilot experiments found that GRU was stronger than long short-term memory (LSTM) for this task.

[4] https://github.com/pytorch/pytorch

[5] https://github.com/HIPS/Spearmint

	MIMIC-III full	MIMIC-III 50	MIMIC-II full
# training documents	47,724	8,067	20,533
Vocabulary size	51,917	51,917	30,688
Mean # tokens per document	1,485	1,530	1,138
Mean # labels per document	15.9	5.7	9.2
Total # labels	8,922	50	5,031

Table 2: Descriptive statistics for MIMIC discharge summary training sets.

	Range	CAML	CNN	Bi-GRU
d_c	50-500	50	500	–
k	2-10	10	4	–
q	0.2-0.8	0.2	0.2	–
ρ	0, 0.001, 0.01, 0.1	0	0	0
η	0.0001, 0.0003, 0.001, 0.003	0.0001	0.003	0.003
s	1-4	–	–	1
v	32-512	–	–	512

Table 3: Hyperparameter ranges and optimal values for each neural model selected by Spearmint.

embedding layer. We use a fixed batch size of 16 for all models and datasets. Models are trained with early stopping on the validation set; training terminates after the precision@8 does not improve for 10 epochs, and the model at the time of the highest precision@8 is used on the test set.

3.3 Evaluation Metrics

To facilitate comparison with both future and prior work, we report a variety of metrics, focusing on the micro-averaged and macro-averaged F1 and area under the ROC curve (AUC). Micro-averaged values are calculated by treating each (text, code) pair as a separate prediction. Macro-averaged values, while less frequently reported in the multi-label classification literature, are calculated by averaging metrics computed per-label. For recall, the metrics are distinguished as follows:

$$\text{Micro-R} = \frac{\sum_{\ell=1}^{|\mathcal{L}|} \text{TP}_\ell}{\sum_{\ell=1}^{|\mathcal{L}|} \text{TP}_\ell + \text{FN}_\ell} \tag{8}$$

$$\text{Macro-R} = \frac{1}{|\mathcal{L}|} \sum_{\ell=1}^{|\mathcal{L}|} \frac{\text{TP}_\ell}{\text{TP}_\ell + \text{FN}_\ell}, \tag{9}$$

where TP denotes true positive examples and FN denotes false negative examples. Precision is computed analogously. The macro-averaged metrics place much more emphasis on rare label prediction.

We also report precision at n (denoted as 'P@n'), which is the fraction of the n highest-scored labels that are present in the ground truth. This is motivated by the potential use case as a decision support application, in which a user is presented with a fixed number of predicted codes to review. In such a case, it is more suitable to select a model with high precision than high recall. We choose $n = 5$ and $n = 8$ to compare with prior work (Vani et al., 2017; Prakash et al., 2017). For the MIMIC-III full label setting, we also compute precision@15, which roughly corresponds to the average number of codes in MIMIC-III discharge summaries (Table 2).

3.4 Results

Our main quantitative evaluation involves predicting the full set of ICD-9 codes based on the text of the MIMIC-III discharge summaries. These results are shown in Table 4. The CAML model gives the strongest results on all metrics. Attention yields substantial improvements over the "vanilla" convolutional neural network (CNN). The recurrent Bi-GRU architecture is comparable to the vanilla CNN, and the logistic regression baseline is substantially worse than all neural architectures. The best-performing CNN model has 9.86M tunable parameters, compared with 6.14M tunable parameters for CAML. This is due to the hyperparameter search preferring a larger number of filters for the CNN. Finally, we observe that the DR-CAML performs worse on most metrics than CAML, with a tuned regularization coefficient of $\lambda = 0.01$.

Among prior work, only Scheurwegs et al. (2017) evaluate on the full ICD-9 code set for MIMIC-III. Their reported results distinguished between diagnosis codes and procedure codes. The CAML models are stronger on both sets. Additionally, our method does not make use of any external information or structured data, while

	AUC		F1				P@n	
Model	Macro	Micro	Macro	Micro	Diag	Proc	8	15
Scheurwegs et. al (2017)	–	–	–	–	0.428	0.555	–	–
Logistic Regression	0.561	0.937	0.011	0.272	0.242	0.398	0.542	0.411
CNN	0.806	0.969	0.042	0.419	0.402	0.491	0.581	0.443
Bi-GRU	0.822	0.971	0.038	0.417	0.393	0.514	0.585	0.445
CAML	0.895	**0.986***	**0.088**	**0.539***	**0.524***	**0.609***	**0.709***	**0.561***
DR-CAML	**0.897**	0.985	0.086	0.529	0.515	0.595	0.690	0.548

Table 4: Results on MIMIC-III full, 8922 labels. Here, "Diag" denotes Micro-F1 performance on diagnosis codes only, and "Proc" denotes Micro-F1 performance on procedure codes only. Here and in all tables, (*) by the bold (best) result indicates significantly improved results compared to the next best result, $p < 0.001$.

Scheurwegs et al. use structured data and various medical ontologies in their text representation.

We feel that precision@8 is the most informative of the metrics, as it measures the ability of the system to return a small high-confidence subset of codes. Even with a space of thousands of labels, our models achieve relatively high precision: of the eight most confident predictions, on average 5.5 are correct. It is also apparent how difficult it is to achieve high Macro-F1 scores, due to the metric's emphasis on rare-label performance. To put these results in context, a hypothetical system that performs perfectly on the 500 most common labels, and ignores all others, would achieve a Macro-F1 of 0.052 and a Micro-F1 of 0.842.

Secondary evaluations To compare with prior published work, we also evaluate on the 50 most common codes in MIMIC-III (Table 5), and on MIMIC-II (Table 6). We report DR-CAML results on the 50-label setting of MIMIC-III with $\lambda = 10$, and on MIMIC-II with $\lambda = 0.1$, which were determined by grid search on a validation set. The other hyperparameters were left at the settings for the main MIMIC-III evaluation, as described in Table 3. In the 50-label setting of MIMIC-III, we see strong improvement over prior work in all reported metrics, as well as against the baselines, with the exception of precision@5, on which the CNN baseline performs best. We hypothesize that this is because the relatively large value of $k = 10$ for CAML leads to a larger network that is more suited to larger datasets; tuning CAML's hyperparameters on this dataset would be expected to improve performance on all metrics. Baumel et al. (2018) additionally report a micro-F1 score of 0.407 by training on MIMIC-III, and evaluating on MIMIC-II. Our model achieves better performance using only the (smaller) MIMIC-II training set, leaving this alternative training protocol for future work.

4 Evaluation of Interpretability

We now evaluate the explanations generated by CAML's attention mechanism, in comparison with three alternative heuristics. A physician was presented with explanations from four methods, using a random sample of 100 predicted codes from the MIMIC-III full-label test set. The most important k-gram from each method was extracted, along with a window of five words on either side for context. We select $k = 4$ in this setting to emulate a span of attention over words likely to be given by a human reader. Examples can be found in Table 1. Observe that the snippets may overlap in multiple words. We prompted the evaluator to select all text snippets which he felt adequately explained the presence of a given code, provided the code and its description, with the option to distinguish snippets as "highly informative" should they be found particularly informative over others.

4.1 Extracting informative text snippets

CAML The attention mechanism allows us to extract k-grams from the text that are most influential in the prediction of each label, by taking the argmax of the SoftMax output $\boldsymbol{\alpha}_\ell$.

Max-pooling CNN We select the k-grams that provide the maximum value selected by max-pooling at least once and weighting by the final layer weights. Defining an argmax vector $\boldsymbol{a}$ which

Model	AUC Macro	AUC Micro	F1 Macro	F1 Micro	P@5
C-MemNN (Prakash et al., 2017)	0.833	–	–	–	0.42
Shi et al. (2017)	–	0.900	–	0.532	–
Logistic Regression	0.829	0.864	0.477	0.533	0.546
CNN	0.876	0.907	**0.576***	0.625	**0.620**
Bi-GRU	0.828	0.868	0.484	0.549	0.591
CAML	0.875	0.909	0.532	0.614	0.609
DR-CAML	**0.884***	**0.916**	**0.576***	**0.633**	0.618

Table 5: Results on MIMIC-III, 50 labels.

results from the max-pooling step as

$$\boldsymbol{a}_i = \underset{j \in \{1,\ldots,m-k+1\}}{\arg\max} (\boldsymbol{H}_{ij}), \tag{10}$$

we can compute the importance of position i for label ℓ,

$$\alpha_{i\ell} = \sum_{j:\boldsymbol{a}_j=i}^{d_c} \beta_{\ell,j}. \tag{11}$$

We then select the most important k-gram for a given label as $\arg\max_i \alpha_{i\ell}$.

Logistic regression The informativeness of each k-gram with respect to label ℓ is scored by the sum of the coefficients of the weight matrix for ℓ, over the words in the k-gram. The top-scoring k-gram is then returned as the explanation.

Code descriptions Finally, we calculate a word similarity metric between each stemmed k-gram and the stemmed ICD-9 code description. We compute the idf-weighted cosine similarity, with idf weights calculated on the corpus consisting of all notes and relevant code descriptions. We then select the argmax over k-grams in the document, breaking ties by selecting the first occurrence. We remove those note-label pairs for which no k-gram has a score greater than 0, which gives an "unfair" advantage to this baseline.

4.2 Results

The results of the interpretability evaluation are presented in Table 7. Our model selects the greatest number of "highly informative" explanations, and selects more "informative" explanations than both the CNN baseline and the logistic regression model. While the cosine similarity metric also performs well, the examples in Table 1 demonstrate the strengths of CAML in extracting text snippets in line with more intuitive explanations for the presence of a code. As noted above, there exist some cases, which we exclude, where the cosine similarity method is unable to provide any explanation, because no k-grams in a note have a nonzero similarity for a given label description. This occurs for about 12% of all note-label pairs in the test set.

5 Related Work

Attentional Convolution for NLP CNNs have been successfully applied to tasks such as sentiment classification (Kim, 2014) and language modeling (Dauphin et al., 2017). Our work combines convolution with attention (Bahdanau et al., 2015; Yang et al., 2016) to select the most relevant parts of the discharge summary. Other recent work has combined convolution and attention (e.g., Allamanis et al., 2016; Yin et al., 2016; dos Santos et al., 2016; Yin and Schütze, 2017). Our attention mechanism is most similar to those of Yang et al. (2016) and Allamanis et al. (2016), in that we use context vectors to compute attention over specific locations in the text. Our work differs in that we compute separate attention weights for each label in our label space, which is better tuned to our goal of selecting locations in a document which are most important for predicting specific labels.

Automatic ICD coding ICD coding is a long-standing task in the medical informatics community, which has been approached with machine learning and handcrafted methods (Scheurwegs et al., 2015). Many recent approaches, like ours, use unstructured text data as the only source of information (e.g., Kavuluru et al., 2015; Subotin and Davis, 2014), though some incorporates struc-

Model	AUC		F1		
	Macro	Micro	Macro	Micro	P@8
Flat SVM (Perotte et al., 2013)	–	–	–	0.293	–
HA-GRU (Baumel et al., 2018)	–	–	–	0.366	–
Logistic Regression	0.690	0.934	0.025	0.314	0.425
CNN	0.742	0.941	0.030	0.332	0.388
Bi-GRU	0.780	0.954	0.024	0.359	0.420
CAML	0.820	**0.966***	0.048	0.442	**0.523***
DR-CAML	**0.826**	**0.966***	**0.049**	**0.457***	0.515

Table 6: Results on MIMIC-II full, 5031 labels.

Method	Informative	Highly informative
CAML	46	22
Code Descriptions	48	20
Logistic Regression	41	18
CNN	36	13

Table 7: Qualitative evaluation results. The columns show the number of examples (out of 100) for which each method was selected as "informative" or "highly informative".

tured data as well (e.g., Scheurwegs et al., 2017; Wang et al., 2016). Most previous methods have either evaluated only on a strict subset of the full ICD label space (Wang et al., 2016), relied on datasets that focus on a subset of medical scenarios (Zhang et al., 2017), or evaluated on data that are not publicly available, making direct comparison difficult (Subotin and Davis, 2016). A recent shared task for ICD-10 coding focused on coding of death certificates in English and French (Névéol et al., 2017). This dataset also contains shorter documents than those we consider, with an average of 18 tokens per certificate in the French corpus. We use the open-access MIMIC datasets containing de-identified, general-purpose records of intensive care unit stays at a single hospital.

Perotte et al. (2013) use "flat" and "hierarchical" SVMs; the former treats each code as an individual prediction, while the latter trains on child codes only if the parent code is present, and predicts on child codes only if the parent code was positively predicted. Scheurwegs et al. (2017) use a feature selection approach to ICD-9 and ICD-10 classification, incorporating structured and unstructured text information from EHRs. They evaluate over various medical specialties and on the MIMIC-III dataset. We compare directly to their results on the full label set of MIMIC-III.

Other recent approaches have employed neural network architectures. Baumel et al. (2018) apply recurrent networks with hierarchical sentence and word attention (the HA-GRU) to classify ICD9 diagnosis codes while providing insights into the model decision process. Similarly, Shi et al. (2017) applied character-aware LSTMs to generate sentence representations from specific subsections of discharge summaries, and apply attention to form a soft matching between the representations and the top 50 codes. Prakash et al. (2017) use memory networks that draw from discharge summaries as well as Wikipedia, to predict top-50 and top-100 codes. Another recent neural architecture is the Grounded Recurrent Neural Network (Vani et al., 2017), which employs a modified GRU with dimensions dedicated to predicting the presence of individual labels. We compare directly with published results from all of these papers, except Vani et al. (2017), who evaluate on only a 5000 code subset of ICD-9. Empirically, the CAML architecture proposed in this paper yields stronger results across all experimental conditions. We attribute these improvements to the attention mechanism, which focuses on the most critical features for each code, rather than applying a uniform pooling operation for all codes. We also observed that convolution-based models are at least as effective, and significantly more computationally efficient, than recurrent neural networks such as the Bi-GRU.

Explainable text classification A goal of this work is that the code predictions be explainable from features of the text. Prior work has also em-

phasized explainability. Lei et al. (2016) model "rationales" through a latent variable, which tags each word as relevant to the document label. Li et al. (2016) compute the salience of individual words by the derivative of the label score with respect to the word embedding. Ribeiro et al. (2016) use submodular optimization to select a subset of features that closely approximate a specific classification decision (this work is also notable for extensive human evaluations). In comparison to these approaches, we employ a relatively simple attentional architecture; this simplicity is motivated by the challenge of scaling to multi-label classification with thousands of possible labels. Other prior work has emphasized the use of attention for highlighting salient features of the text (e.g., Rush et al., 2015; Rocktäschel et al., 2016), although these papers did not perform human evaluations of the interpretability of the features selected by the attention mechanism.

6 Conclusions and Future Work

We present CAML, a convolutional neural network for multi-label document classification, which employs an attention mechanism to adaptively pool the convolution output for each label, learning to identify highly-predictive locations for each label. CAML yields strong improvements over previous metrics on several formulations of the ICD-9 code prediction task, while providing satisfactory explanations for its predictions. Although we focus on a clinical setting, CAML is extensible without modification to other multi-label document tagging tasks, including ICD-10 coding. We see a number of directions for future work. From the linguistic side, we plan to integrate the document structure of discharge summaries in MIMIC-III, and to better handle non-standard writing and other sources of out-of-vocabulary tokens. From the application perspective, we plan to build models that leverage hierarchy of ICD codes (Choi et al., 2016), and to attempt the more difficult task of predicting diagnosis and treatment codes for *future* visits from discharge summaries.

Acknowledgments Helpful feedback was provided by the anonymous reviewers, and by the members of the Georgia Tech Computational Linguistics lab. The project was partially supported by project HDTRA1-15-1-0019 from the Defense Threat Reduction Agency, by the National Science Foundation under awards IIS-1418511 and CCF-1533768, by the National Institutes of Health under awards 1R01MD011682-01 and R56HL138415, by Children's Healthcare of Atlanta, and by UCB.

References

Miltiadis Allamanis, Hao Peng, and Charles Sutton. 2016. A convolutional attention network for extreme summarization of source code. In *International Conference on Machine Learning*. pages 2091–2100.

Anand Avati, Kenneth Jung, Stephanie Harman, Lance Downing, Andrew Ng, and Nigam H. Shah. 2017. Improving palliative care with deep learning. *arXiv preprint arXiv:1711.06402* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Tal Baumel, Jumana Nassour-Kassis, Michael Elhadad, and Noémie Elhadad. 2018. Multi-label classification of patient notes a case study on ICD code assignment. In *AAAI Workshop on Health Intelligence*.

Elena Birman-Deych, Amy D. Waterman, Yan Yan, David S. Nilasena, Martha J. Radford, and Brian F Gage. 2005. Accuracy of ICD-9-CM codes for identifying cardiovascular and stroke risk factors. *Medical care* 43(5):480–485.

Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F Stewart, and Jimeng Sun. 2016. Doctor AI: Predicting clinical events via recurrent neural networks. In *Machine Learning for Healthcare Conference*. pages 301–318.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning*. pages 933–941. http://proceedings.mlr.press/v70/dauphin17a.html.

Luciano R.S. de Lima, Alberto H.F. Laender, and Berthier A. Ribeiro-Neto. 1998. A hierarchical approach to the automatic categorization of medical documents. In *Proceedings of the seventh international conference on Information and knowledge management*. ACM, pages 132–139.

Joshua C. Denny, Marylyn D. Ritchie, Melissa A. Basford, Jill M. Pulley, Lisa Bastarache, Kristin Brown-Gentry, Deede Wang, Dan R. Masys, Dan M. Roden, and Dana C. Crawford. 2010. PheWAS: demonstrating the feasibility of a phenome-wide scan to discover gene–disease associations. *Bioinformatics* 26(9):1205–1210.

Cıcero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *CoRR, abs/1602.03609* .

Alistair E.W. Johnson, Tom J. Pollard, Lu Shen, Li-wei H. Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G. Mark. 2016. MIMIC-III, a freely accessible critical care database. *Scientific data* 3.

Ramakanth Kavuluru, Anthony Rios, and Yuan Lu. 2015. An empirical evaluation of supervised learning approaches in assigning diagnosis codes to electronic medical records. *Artificial intelligence in medicine* 65(2):155–166.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 1746–1751.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 107–117. https://aclweb.org/anthology/D16-1011.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in NLP. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 681–691. http://www.aclweb.org/anthology/N16-1082.

Andrew McCallum. 1999. Multi-label text classification with a mixture model trained by EM. In *AAAI workshop on Text Learning*. pages 1–7.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Aurélie Névéol, Robert N Anderson, K Bretonnel Cohen, Cyril Grouin, Thomas Lavergne, Grégoire Rey, Aude Robert, Claire Rondet, and Pierre Zweigenbaum. 2017. CLEF ehealth 2017 multilingual information extraction task overview: ICD10 coding of death certificates in english and french. In *CLEF 2017 Evaluation Labs and Workshop: Online Working Notes, CEUR-WS*. page 17.

Adler Perotte, Rimma Pivovarov, Karthik Natarajan, Nicole Weiskopf, Frank Wood, and Noémie Elhadad. 2013. Diagnosis code assignment: models and evaluation metrics. *Journal of the American Medical Informatics Association* 21(2):231–237.

Aaditya Prakash, Siyuan Zhao, Sadid A Hasan, Vivek V Datla, Kathy Lee, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2017. Condensed memory networks for clinical diagnostic inferencing. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. pages 3274–3280.

Rajesh Ranganath, Adler J. Perotte, Noémie Elhadad, and David M. Blei. 2015. The survival filter: Joint survival analysis with a latent time series. In *UAI*. pages 742–751.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pages 1135–1144.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *International Conference on Learning Representations*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 379–389. http://aclweb.org/anthology/D15-1044.

Elyne Scheurwegs, Boris Cule, Kim Luyckx, Léon Luyten, and Walter Daelemans. 2017. Selecting relevant features from the electronic health record for clinical code prediction. *Journal of Biomedical Informatics* 74:92–103.

Elyne Scheurwegs, Kim Luyckx, Léon Luyten, Walter Daelemans, and Tim Van den Bulcke. 2015. Data integration of structured and unstructured sources for assigning clinical codes to patient stays. *Journal of the American Medical Informatics Association* 23(e1):e11–e19.

Haoran Shi, Pengtao Xie, Zhiting Hu, Ming Zhang, and Eric P Xing. 2017. Towards automated ICD coding using deep learning. *arXiv preprint arXiv:1711.04075* .

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, Curran Associates, Inc., pages 2951–2959.

Michael Subotin and Anthony R. Davis. 2014. A system for predicting ICD-10-PCS codes from electronic health records. In *Proceedings of the 2014 Workshop on Biomedical Natural Language Processing*.

Michael Subotin and Anthony R. Davis. 2016. A method for modeling co-occurrence propensity of clinical codes with application to ICD-10-PCS auto-coding. *Journal of the American Medical Informatics Association* 23(5):866–871.

Kevin Swersky, Jasper Snoek, and Ryan P Adams. 2013. Multi-task bayesian optimization. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., pages 2004–2012.

Ankit Vani, Yacine Jernite, and David Sontag. 2017. Grounded recurrent neural networks. *arXiv preprint arXiv:1705.08557* .

Sen Wang, Xiaojun Chang, Xue Li, Guodong Long, Lina Yao, and Quan Z Sheng. 2016. Diagnosis code assignment using sparsity-based disease correlation embedding. *IEEE Transactions on Knowledge and Data Engineering* 28(12):3191–3202.

World Health Organization. 2016. International statistical classification of diseases and related health problems 10th revision. `http://apps.who.int/classifications/icd10/browse/2016/en`.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 1480–1489.

Wenpeng Yin and Hinrich Schütze. 2017. Attentive convolution. *arXiv preprint arXiv:1710.00519* .

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. ABCNN: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics* 4:259–272.

Danchen Zhang, Daqing He, Sanqiang Zhao, and Lei Li. 2017. Enhancing automatic ICD-9-CM code assignment for medical texts with pubmed. *BioNLP 2017* pages 263–271.

A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference

Adina Williams[1]
adinawilliams@nyu.edu

Nikita Nangia[2]
nikitanangia@nyu.edu

Samuel R. Bowman[1,2,3]
bowman@nyu.edu

[1]Department of Linguistics
New York University

[2]Center for Data Science
New York University

[3]Department of Computer Science
New York University

Abstract

This paper introduces the Multi-Genre Natural Language Inference (MultiNLI) corpus, a dataset designed for use in the development and evaluation of machine learning models for sentence understanding. At 433k examples, this resource is one of the largest corpora available for natural language inference (a.k.a. *recognizing textual entailment*), improving upon available resources in both its coverage and difficulty. MultiNLI accomplishes this by offering data from ten distinct genres of written and spoken English, making it possible to evaluate systems on nearly the full complexity of the language, while supplying an explicit setting for evaluating cross-genre domain adaptation. In addition, an evaluation using existing machine learning models designed for the Stanford NLI corpus shows that it represents a substantially more difficult task than does that corpus, despite the two showing similar levels of inter-annotator agreement.

1 Introduction

Many of the most actively studied problems in NLP, including question answering, translation, and dialog, depend in large part on natural language understanding (NLU) for success. While there has been a great deal of work that uses representation learning techniques to pursue progress on these applied NLU problems directly, in order for a representation learning model to fully succeed at one of these problems, it must simultaneously succeed both at NLU, and at one or more additional hard machine learning problems like structured prediction or memory access. This makes it difficult to accurately judge the degree to which current models extract reasonable representations of language meaning in these settings.

The task of natural language inference (NLI) is well positioned to serve as a benchmark task for research on NLU. In this task, also known as *recognizing textual entailment* (Cooper et al., 1996; Fyodorov et al., 2000; Condoravdi et al., 2003; Bos and Markert, 2005; Dagan et al., 2006; MacCartney and Manning, 2009), a model is presented with a pair of sentences—like one of those in Figure 1—and asked to judge the relationship between their meanings by picking a label from a small set: typically ENTAILMENT, NEUTRAL, and CONTRADICTION. Succeeding at NLI does not require a system to solve any difficult machine learning problems except, crucially, that of extracting effective and thorough representations for the meanings of sentences (i.e., their lexical and compositional semantics). In particular, a model must handle phenomena like lexical entailment, quantification, coreference, tense, belief, modality, and lexical and syntactic ambiguity.

As the only large human-annotated corpus for NLI currently available, the Stanford NLI Corpus (SNLI; Bowman et al., 2015) has enabled a good deal of progress on NLU, serving as a major benchmark for machine learning work on sentence understanding and spurring work on core representation learning techniques for NLU, such as attention (Wang and Jiang, 2016; Parikh et al., 2016), memory (Munkhdalai and Yu, 2017), and the use of parse structure (Mou et al., 2016b; Bowman et al., 2016; Chen et al., 2017). However, SNLI falls short of providing a sufficient testing ground for machine learning models in two ways.

Proceedings of NAACL-HLT 2018, pages 1112–1122
New Orleans, Louisiana, June 1 - 6, 2018.

Met my first girlfriend that way.	FACE-TO-FACE **contradiction** C C N C	I didn't meet my first girlfriend until later.
8 million in relief in the form of emergency housing.	GOVERNMENT **neutral** N N N N	The 8 million dollars for emergency housing was still not enough to solve the problem.
Now, as children tend their gardens, they have a new appreciation of their relationship to the land, their cultural heritage, and their community.	LETTERS **neutral** N N N N	All of the children love working in their gardens.
At 8:34, the Boston Center controller received a third transmission from American 11	9/11 **entailment** E E E E	The Boston Center controller got a third transmission from American 11.
I am a lacto-vegetarian.	SLATE **neutral** N N E N	I enjoy eating cheese too much to abstain from dairy.
someone else noticed it and i said well i guess that's true and it was somewhat melodious in other words it wasn't just you know it was really funny	TELEPHONE **contradiction** C C C C	No one noticed and it wasn't funny at all.

Table 1: Randomly chosen examples from the development set of our new corpus, shown with their genre labels, their selected gold labels, and the validation labels (abbreviated E, N, C) assigned by individual annotators.

First, the sentences in SNLI are derived from only a single text genre—image captions—and are thus limited to descriptions of concrete visual scenes, rendering the hypothesis sentences used to describe these scenes short and simple, and rendering many important phenomena—like temporal reasoning (e.g., *yesterday*), belief (e.g., *know*), and modality (e.g., *should*)—rare enough to be irrelevant to task performance. Second, because of these issues, SNLI is not sufficiently demanding to serve as an effective benchmark for NLU, with the best current model performance falling within a few percentage points of human accuracy and limited room left for fine-grained comparisons between strong models.

This paper introduces a new challenge dataset, the Multi-Genre NLI Corpus (MultiNLI), whose chief purpose is to remedy these limitations by making it possible to run large-scale NLI evaluations that capture more of the complexity of modern English. While its size (433k pairs) and mode of collection are modeled closely on SNLI, unlike that corpus, MultiNLI represents both written and spoken speech in a wide range of styles, degrees of formality, and topics.

Our chief motivation in creating this corpus is to provide a benchmark for ambitious machine learning research on the core problems of NLU, but we are additionally interested in constructing a corpus that facilitates work on domain adaptation and cross-domain transfer learning. These techniques—which use labeled training data for a source domain, and aim to train a model that performs well on test data from a target domain with a different distribution—have resulted in gains across many tasks (Daume III and Marcu, 2006; Ben-David et al., 2007), including sequence and part-of-speech tagging (Blitzer et al., 2006; Peng and Dredze, 2017). Moreover, in application areas outside NLU, artificial neural network techniques have made it possible to train general-purpose feature extractors that, with no or minimal retraining, can extract useful features for a variety of styles of data (Krizhevsky et al., 2012; Zeiler and Fergus, 2014; Donahue et al., 2014). However, attempts to bring this kind of general purpose representation learning to NLU have seen only very limited success (see, for example, Mou et al., 2016a). Nearly all successful applications of representation learning to NLU have involved models that are trained on data closely resembling the target evaluation data in both task and style. This fact limits the usefulness of these tools for problems involving styles of language not represented in large annotated training sets.

With this in mind, we construct MultiNLI so as to make it possible to explicitly evaluate models both on the quality of their sentence representations within the training domain and on their ability to derive reasonable representations in unfamiliar domains. The corpus is derived from ten different genres of written and spoken English, which are collectively meant to approximate the full diversity of ways in which modern standard

This task will involve reading a line from a non-fiction article and writing three sentences that relate to it. The line will describe a situation or event. Using only this description and what you know about the world:

- Write one sentence that is definitely correct about the situation or event in the line.
- Write one sentence that might be correct about the situation or event in the line.
- Write one sentence that is definitely incorrect about the situation or event in the line.

Figure 1: The main text of a prompt (truncated) that was presented to our annotators. This version is used for the written non-fiction genres.

American English is used. All of the genres appear in the test and development sets, but only five are included in the training set. Models thus can be evaluated on both the *matched* test examples, which are derived from the same sources as those in the training set, and on the *mismatched* examples, which do not closely resemble any of those seen at training time.

2 The Corpus

2.1 Data Collection

The data collection methodology for MultiNLI is similar to that of SNLI: We create each sentence pair by selecting a premise sentence from a preexisting text source and asking a human annotator to compose a novel sentence to pair with it as a hypothesis. This section discusses the sources of our premise sentences, our collection method for hypotheses, and our validation (relabeling) strategy.

Premise Text Sources The MultiNLI premise sentences are derived from ten sources of freely available text which are meant to be maximally diverse and roughly represent the full range of American English. We selected nine sources from the second release of the Open American National Corpus (OANC; Fillmore et al., 1998; Macleod et al., 2000; Ide and Macleod, 2001; Ide and Suderman, 2006, downloaded 12/2016[1]), balancing the volume of source text roughly evenly across genres, and avoiding genres with content that would be too difficult for untrained annotators.

OANC data constitutes the following nine genres: transcriptions from the Charlotte Narrative and Conversation Collection of two-sided, in-person conversations that took place in the early 2000s (FACE-TO-FACE); reports, speeches, letters, and press releases from public domain government websites (GOVERNMENT); letters from the Indiana Center for Intercultural Communication of Philanthropic Fundraising Discourse written in the late 1990s–early 2000s (LETTERS); the public report from the National Commission on Terrorist Attacks Upon the United States released on July 22, 2004[2] (9/11); five non-fiction works on the textile industry and child development published by the Oxford University Press (OUP); popular culture articles from the archives of Slate Magazine (SLATE) written between 1996–2000; transcriptions from University of Pennsylvania's Linguistic Data Consortium Switchboard corpus of two-sided, telephone conversations that took place in 1990 or 1991 (TELEPHONE); travel guides published by Berlitz Publishing in the early 2000s (TRAVEL); and short posts about linguistics for non-specialists from the Verbatim archives written between 1990 and 1996 (VERBATIM).

For our tenth genre, FICTION, we compile several freely available works of contemporary fiction written between 1912 and 2010, spanning various genres, including mystery (*The Mysterious Affair at Styles*,[3] Christie, 1921; *The Secret Adversary*,[4] Christie, 1922; *Murder in the Gun Room*,[5] Piper, 1953), humor (*Password Incorrect*,[6] Name, 2008), western (*Rebel Spurs*,[7] Norton, 1962), science fiction (*Seven Swords*,[8] Shea, 2008; *Living History*,[9] Essex, 2016; *The Sky Is Falling*,[10] Del Rey, 1973; *Youth*,[11] Asimov, May 1952), and adventure (*Captain Blood*,[12] Sabatini, 1922).

We construct premise sentences from these ten source texts with minimal preprocessing; unique the sentences within genres, exclude very short

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

[2] https://9-11commission.gov/

[3] gutenberg.org/files/863/863-0.txt

[4] gutenberg.org/files/1155/1155-0.txt

[5] gutenberg.org/files/17866/17866.txt

[6] http://manybooks.net/pages/namenother09password_incorrect/0.html

[7] gutenberg.org/files/20840/20840-0.txt

[8] http://mikeshea.net/stories/seven_swords.html, shared with the author's permission.

[9] manybooks.net/pages/essexbother10living_history/0.html

[10] gutenberg.org/cache/epub/18768/pg18768.txt

[11] gutenberg.org/cache/epub/31547/pg31547.txt

[12] gutenberg.org/files/1965/1965-0.txt

sentences (under eight characters), and manually remove certain types of non-narrative writing, such as mathematical formulae, bibliographic references, and lists.

Although SNLI is collected in largely the same way as MultiNLI, and is also permissively licensed, we do not include SNLI in the MultiNLI corpus distribution. SNLI can be appended and treated as an unusually large additional CAPTIONS genre, built on image captions from the Flickr30k corpus (Young et al., 2014).

Hypothesis Collection To collect a sentence pair, we present a crowdworker with a sentence from a source text and ask them to compose three novel sentences (the hypotheses): one which is necessarily true or appropriate whenever the premise is true (paired with the premise and labeled ENTAILMENT), one which is necessarily false or inappropriate whenever the premise is true (CONTRADICTION), and one where neither condition applies (NEUTRAL). This method of data collection ensures that the three classes will be represented equally in the raw corpus.

The prompts that surround each premise sentence during hypothesis collection are slightly tailored to fit the genre of that premise sentence. We pilot these prompts prior to data collection to ensure that the instructions are clear and that they yield hypothesis sentences that fit the intended meanings of the three classes. There are five unique prompts in total: one for written non-fiction genres (SLATE, OUP, GOVERNMENT, VERBATIM, TRAVEL; Figure 1), one for spoken genres (TELEPHONE, FACE-TO-FACE), one for each of the less formal written genres (FICTION, LETTERS), and a specialized one for 9/11, tailored to fit its potentially emotional content. Each prompt is accompanied by example premises and hypothesis that are specific to each genre.

Below the instructions, we present three text fields—one for each label—followed by a field for reporting issues, and a link to the frequently asked questions (FAQ) page. We provide one FAQ page per prompt. FAQs are modeled on their SNLI counterparts (supplied by the authors of that work) and include additional curated examples, answers to genre-specific questions arising from our pilot phase, and information about logistical concerns like payment.

For both hypothesis collection and validation, we present prompts to annotators using Hybrid (gethybrid.io), a crowdsoucring platform similar to the Amazon Mechanical Turk platform used for SNLI. We used this platform to hire an organized group of workers. 387 annotators contributed through this group, and at no point was any identifying information about them, including demographic information, available to the authors.

Statistic	SNLI	MultiNLI
Pairs w/ unanimous gold label	58.3%	58.2%
Individual label = gold label	89.0%	88.7%
Individual label = author's label	85.8%	85.2%
Gold label = author's label	91.2%	92.6%
Gold label ≠ author's label	6.8%	5.6%
No gold label (no 3 labels match)	2.0%	1.8%

Table 2: Key validation statistics for SNLI (copied from Bowman et al., 2015) and MultiNLI.

Validation We perform an additional round of annotation on test and development examples to ensure accurate labelling. The validation phase follows the same procedure used for SICK (Marelli et al., 2014b) and SNLI: Workers are presented with pairs of sentences and asked to supply a single label (ENTAILMENT, CONTRADICTION, NEUTRAL) for the pair. Each pair is relabeled by four workers, yielding a total of five labels per example. Validation instructions are tailored by genre, based on the main data collection prompt (Figure 1); a single FAQ, modeled after the validation FAQ from SNLI, is provided for reference. In order to encourage thoughtful labeling, we manually label one percent of the validation examples and offer a $1 bonus each time a worker selects a label that matches ours.

For each validated sentence pair, we assign a *gold label* representing a majority vote between the initial label assigned to the pair by the original annotator, and the four additional labels assigned by validation annotators. A small number of examples did not receive a three-vote consensus on any one label. These examples are included in the distributed corpus, but are marked with '–' in the gold label field, and should not be used in standard evaluations. Table 2 shows summary statistics capturing the results of validation, alongside corresponding figures for SNLI. These statistics indicate that the labels included in MultiNLI are about as reliable as those included in SNLI, despite MultiNLI's more diverse text contents.

Genre	#Examples Train	Dev.	Test	#Wds. Prem.	'S' parses Prem.	Hyp.	Agrmt.	Model Acc. ESIM	CBOW
SNLI	*550,152*	*10,000*	*10,000*	*14.1*	*74%*	*88%*	*89.0%*	*86.7%*	*80.6 %*
FICTION	77,348	2,000	2,000	14.4	94%	97%	89.4%	73.0%	67.5%
GOVERNMENT	77,350	2,000	2,000	24.4	90%	97%	87.4%	74.8%	67.5%
SLATE	77,306	2,000	2,000	21.4	94%	98%	87.1%	67.9%	60.6%
TELEPHONE	83,348	2,000	2,000	25.9	71%	97%	88.3%	72.2%	63.7%
TRAVEL	77,350	2,000	2,000	24.9	97%	98%	89.9%	73.7%	64.6%
9/11	0	2,000	2,000	20.6	98%	99%	90.1%	71.9%	63.2%
FACE-TO-FACE	0	2,000	2,000	18.1	91%	96%	89.5%	71.2%	66.3%
LETTERS	0	2,000	2,000	20.0	95%	98%	90.1%	74.7%	68.3%
OUP	0	2,000	2,000	25.7	96%	98%	88.1%	71.7%	62.8%
VERBATIM	0	2,000	2,000	28.3	93%	97%	87.3%	71.9%	62.7%
MultiNLI Overall	**392,702**	**20,000**	**20,000**	**22.3**	**91%**	**98%**	**88.7%**	**72.2%**	**64.7%**

Table 3: Key statistics for the corpus by genre. The first five genres represent the *matched* section of the development and test sets, and the remaining five represent the *mismatched* section. The first three statistics provide the number of examples in each genre. *#Wds. Prem.* is the mean token count among premise sentences. *'S' parses* is the percentage of sentences for which the Stanford Parser produced a parse rooted with an 'S' (sentence) node. *Agrmt.* is the percent of individual labels that match the gold label in validated examples. *Model Acc.* gives the test accuracy for ESIM and CBOW models (trained on either SNLI or MultiNLI), as described in Section 3.

2.2 The Resulting Corpus

Table 1 shows randomly chosen development set examples from the collected corpus. Hypotheses tend to be fluent and correctly spelled, though not all are complete sentences. Punctuation is often omitted. Hypotheses can rely heavily on knowledge about the world, and often don't correspond closely with their premises in syntactic structure.

Unlabeled test data is available on Kaggle for both matched and mismatched sets as competitions that will be open indefinitely; Evaluations on a subset of the test set have previously been conducted with different leaderboards through the RepEval 2017 Workshop (Nangia et al., 2017).

The corpus is available in two formats—tab separated text and JSON Lines (`jsonl`), following SNLI. For each example, premise and hypothesis strings, unique identifiers for the pair and prompt, and the following additional fields are specified:

- `gold_label`: label used for classification. In examples rejected during the validation process, the value of this field will be '–'.
- `sentence{1,2}_parse`: Each sentence as parsed by the Stanford PCFG Parser 3.5.2 (Klein and Manning, 2003).
- `sentence{1,2}_binary_parse`: parses in unlabeled binary-branching format.
- `label[1]`: The label assigned during the creation of the sentence pair. In rare cases this may be different from `gold_label`, if a consensus of annotators chose a different label during the validation phase.
- `label[2...5]`: The four labels assigned during validation by individual annotators to each development and test example. These fields will be empty for training examples.

The current version of the corpus is freely available at `nyu.edu/projects/bowman/multinli/` for typical machine learning uses, and may be modified and redistributed. The majority of the corpus is released under the OANC's license, which allows all content to be freely used, modified, and shared under permissive terms. The data in the FICTION section falls under several permissive licenses; *Seven Swords* is available under a Creative Commons Share-Alike 3.0 Unported License, and with the explicit permission of the author, *Living History* and *Password Incorrect* are available under Creative Commons Attribution 3.0 Unported Licenses; the remaining works of fiction are in the public domain in the United States (but may be licensed differently elsewhere).

Partition The distributed corpus comes with an explicit train/test/development split. The test and development sets contain 2,000 randomly selected examples each from each of the genres, resulting in a total of 20,000 examples per set. No premise sentence occurs in more than one set.

Train	Model	SNLI	MNLI Match.	MNLI Mis.
	Most freq.	34.3	36.5	35.6
SNLI	CBOW	80.6	-	-
	BiLSTM	81.5	-	-
	ESIM	**86.7**	-	-
MNLI	CBOW	51.5	64.8	64.5
	BiLSTM	50.8	66.9	66.9
	ESIM	60.7	**72.3**	**72.1**
MNLI+ SNLI	CBOW	74.7	65.2	64.6
	BiLSTM	74.0	67.5	67.1
	ESIM	79.7	**72.4**	**71.9**

Table 4: Test set accuracies (%) for all models; *Match.* represents test set performance on the MultiNLI genres that are also represented in the training set, *Mis.* represents test set performance on the remaining ones; *Most freq.* is a trivial 'most frequent class' baseline.

Statistics Table 3 shows some additional statistics. Premise sentences in MultiNLI tend to be longer (max 401 words, mean 22.3 words) than their hypotheses (max 70 words, mean 11.4 words), and much longer, on average, than premises in SNLI (mean 14.1 words); premises in MultiNLI also tend to be parsed as complete sentences at a much higher rate on average (91%) than their SNLI counterparts (74%). We observe that the two spoken genres differ in this—with FACE-TO-FACE showing more complete sentences (91%) than TELEPHONE (71%)—and speculate that the lack of visual feedback in a telephone setting may result in a high incidence of interrupted or otherwise incomplete sentences.

Hypothesis sentences in MultiNLI generally cannot be derived from their premise sentences using only trivial editing strategies. While 2.5% of the hypotheses in SNLI differ from their premises by deletion, only 0.9% of those in MultiNLI (170 examples total) are constructed in this way. Similarly, in SNLI, 1.6% of hypotheses differ from their premises by addition, substitution, or shuffling a single word, while in MultiNLI this only happens in 1.2% of examples. The percentage of hypothesis-premise pairs with high token overlap (>37%) was comparable between MultiNLI (30% of pairs) and SNLI (29%). These statistics suggest that MultiNLI's annotations are comparable in quality to those of SNLI.

3 Baselines

To test the difficulty of the corpus, we experiment with three neural network models. The first is a simple continuous bag of words (CBOW) model in which each sentence is represented as the sum of the embedding representations of its words. The second computes representations by averaging the states of a bidirectional LSTM RNN (BiLSTM; Hochreiter and Schmidhuber, 1997) over words. For the third, we implement and evaluate Chen et al.'s Enhanced Sequential Inference Model (ESIM), which is roughly tied for the state of the art on SNLI at the time of writing. We use the base ESIM without ensembling with a TreeLSTM (as in the 'HIM' runs in that work).

The first two models produce separate vector representations for each sentence and compute label predictions for pairs of representations. To do this, they concatenate the representations for premise and hypothesis, their difference, and their element-wise product, following Mou et al. (2016b), and pass the result to a single tanh layer followed by a three-way softmax classifier.

All models are initialized with 300D reference GloVe vectors (840B token version; Pennington et al., 2014). Out-of-vocabulary (OOV) words are initialized randomly and word embeddings are fine-tuned during training. The models use 300D hidden states, as in most prior work on SNLI. We use Dropout (Srivastava et al., 2014) for regularization. For ESIM, we use a dropout rate of 0.5, following the paper. For CBOW and BiLSTM models, we tune Dropout on the SNLI development set and find that a drop rate of 0.1 works well. We use the Adam (Kingma and Ba, 2015) optimizer with default parameters. Code is available at `github.com/nyu-mll/multiNLI/`.

We train models on SNLI, MultiNLI, and a mixture; Table 4 shows the results. In the mixed setting, we use the full MultiNLI training set and randomly select 15% of the SNLI training set at each epoch, ensuring that each available genre is seen during training with roughly equal frequency.

We also train a separate CBOW model on each individual genre to establish the degree to which simple models already allow for effective transfer across genres, using a dropout rate of 0.2. When training on SNLI, a single random sample of 15% of the original training set is used. For each genre represented in the training set, the model that performs best on it was trained on that genre; a model trained only on SNLI performs worse on every genre than comparable models trained on any genre from MultiNLI.

Models trained on a single genre from MultiNLI perform well on similar genres; for example, the model trained on TELEPHONE attains the best accuracy (63%) on FACE-TO-FACE, which was nearly one point better than it received on itself. SLATE seems to be a difficult and relatively unusual genre and performance on it is relatively poor in this setting; when averaging over runs trained on SNLI and all genres in the matched section of the training set, average performance on SLATE was only 57.5%. Sentences in SLATE cover a wide range of topics and phenomena, making it hard to do well on, but also forcing models trained on it be broadly capable; the model trained on SLATE achieves the highest accuracy of any model on 9/11 (55.6%) and VERBATIM (57.2%), and relatively high accuracy on TRAVEL (57.4%) and GOVERNMENT (58.3%). We also observe that our models perform similarly on both the matched and mismatched test sets of MultiNLI. We expect genre mismatch issues to become more conspicuous as models are developed that can better fit MultiNLI's training genres.

To evaluate the contribution of sentence length to corpus difficulty, we binned premises and hypotheses by length in 25-word increments for premises and 10-word increments for hypotheses. Using the ESIM model, our strong baseline, we find a small effect (stronger for matched than mismatched) of premise length on model accuracy: accuracy decreases slightly as premise sentences increase in length. We find no effect of hypothesis length on accuracy.

4 Discussion and Analysis

4.1 Data Collection

In data collection for NLI, different annotator decisions about the coreference between entities and events across the two sentences in a pair can lead to very different assignments of pairs to labels (de Marneffe et al., 2008; Marelli et al., 2014a; Bowman et al., 2015). Drawing an example from Bowman et al., the pair *"a boat sank in the Pacific Ocean"* and *"a boat sank in the Atlantic Ocean"* can be labeled either CONTRADICTION or NEUTRAL depending on (among other things) whether the two mentions of boats are assumed to refer to the same entity in the world. This uncertainty can present a serious problem for inter-annotator agreement, since it is not clear that it is possible to define an explicit set of rules around coreference that would be easily intelligible to an untrained annotator (or any non-expert).

Bowman et al. attempt to avoid this problem by using an annotation prompt that is highly dependent on the concreteness of image descriptions; but, as we engage with the much more abstract writing that is found in, for example, government documents, there is no reason to assume *a priori* that any similar prompt and annotation strategy can work. We are surprised to find that this is not a major issue. Through a relatively straightforward trial-and-error piloting phase, followed by discussion with our annotators, we manage to design prompts for abstract genres that yield high inter-annotator agreement scores nearly identical to those of SNLI (see Table 2). These high scores suggest that our annotators agreed on a single task definition, and were able to apply it consistently across genres.

4.2 Overall Difficulty

As expected, both the increase in the diversity of linguistic phenomena in MultiNLI and its longer average sentence length conspire to make MultiNLI dramatically more difficult than SNLI. Our three baseline models perform better on SNLI than MultiNLI by about 15% when trained on the respective datasets. All three models achieve accuracy above 80% on the SNLI test set when trained only on SNLI. However, when trained on MultiNLI, only ESIM surpasses 70% accuracy on MultiNLI's test sets. When we train models on MultiNLI and downsampled SNLI, we see an expected significant improvement on SNLI, but no significant change in performance on the MultiNLI test sets, suggesting including SNLI in training doesn't drive substantial improvement. These results attest to MultiNLI's difficulty, and with its relatively high inter-annotator agreement, suggest that it presents a problem with substantial headroom for future work.

4.3 Analysis by Linguistic Phenomenon

To better understand the types of language understanding skills that MultiNLI tests, we analyze the collected corpus using a set of annotation tags chosen to reflect linguistic phenomena which are known to be potentially difficult. We use two methods to assign tags to sentences. First, we use the Penn Treebank (PTB; Marcus et al., 1993) part-of-speech tag set (via the included Stanford Parser parses) to automatically isolate sentences

Tag	Dev. Freq. SNLI	Dev. Freq. MultiNLI	Diff.	Most Frequent Label Label	Most Frequent Label %	Model Acc. CBOW	Model Acc. BiLSTM	Model Acc. ESIM
Entire Corpus	100	100	0	entailment	*∼35*	*∼65*	*∼67*	*∼72*
Pronouns (PTB)	34	68	34	entailment	34	66	68	73
Quantifiers	33	63	30	contradiction	36	66	68	73
Modals (PTB)	<1	28	28	entailment	35	65	67	72
Negation (PTB)	5	31	26	contradiction	**48**	67	70	75
WH terms (PTB)	5	30	25	entailment	35	64	65	72
Belief Verbs	<1	19	18	entailment	34	64	67	71
Time Terms	19	36	17	neutral	35	64	66	71
Discourse Mark.	<1	14	14	neutral	34	62	64	70
Presup. Triggers	8	22	14	neutral	34	65	67	73
Compr./Supr.(PTB)	3	17	14	neutral	39	61	63	69
Conditionals	4	15	11	neutral	35	65	68	73
Tense Match (PTB)	62	69	7	entailment	37	67	68	73
Interjections (PTB)	<1	5	5	entailment	36	67	70	75
>20 words	<1	5	5	entailment	**42**	65	67	76

Table 5: Dev. Freq. is the percentage of dev. set examples that include each phenomenon, ordered by greatest difference in frequency of occurrence (Diff.) between MultiNLI and SNLI. Most Frequent Label specifies which label is the most frequent for each tag in the MultiNLI dev. set, and % is its incidence. Model Acc. is the dev. set accuracy (%) by annotation tag for each baseline model (trained on MultiNLI only). (PTB) marks a tag as derived from Penn Treebank-style parser output tags (Marcus et al., 1993).

containing a range of easily-identified phenomena like comparatives. Second, we isolate sentences that contain hand-chosen key words indicative of additional interesting phenomena.

The hand-chosen tag set covers the following phenomena: QUANTIFIERS contains single words with quantificational force (see, for example, Heim and Kratzer, 1998; Szabolcsi, 2010, e.g., *many, all, few, some*); BELIEF VERBS contains sentence-embedding verbs denoting mental states (e.g., *know, believe, think*), including irregular past tense forms; TIME TERMS contains single words with abstract temporal interpretation, (e.g., *then, today*) and month names and days of the week; DISCOURSE MARKERS contains words that facilitate discourse coherence (e.g., *yet, however, but, thus, despite*); PRESUPPOSITION TRIGGERS contains words with lexical presuppositions (Stalnaker, 1974; Schlenker, 2016, e.g., *again, too, anymore*[13]); CONDITIONALS contains the word *if*. Table 5 presents the frequency of the tags in SNLI and MultiNLI, and model accuracy on MultiNLI (trained only on MultiNLI).

The incidence of tags varies by genre; the percentage of sentence pairs containing a particular annotation tag differs by a maximum over 30% across genres. Sentence pairs containing pronouns are predictably common for all genres, with 93% of Government and Face-to-face pairs including at least one. The Telephone genre has the highest percentage of sentence pairs containing one occurrence of negation, WH-words, *belief*-verbs and time terms, Verbatim has the highest percentage of pairs containing quantifiers and conversational pivots, and Letters has the highest percentage of pairs that contain one or more modals. Pairs containing comparatives and/or superlatives, which is the tag that our baseline models perform worst on, are most common in the Oxford University Press genre. Based on this, we conclude that the genres are sufficiently different, because they are not uniform with respect to the percentages of sentence pairs that contain each of the annotation tags.

The distributions of labels within each tagged subset of the corpus roughly mirrors the balanced overall distribution. The most frequent class overall (in this case, ENTAILMENT) occurs with a frequency of roughly one third (see Table 4) in most. Only two annotation tags differ from the baseline percentage of the most frequent class in the corpus by at least 5%: sentences containing negation, and sentences exceeding 20 words. Sentences that contain negation are slightly more likely than average to be labeled CONTRADICTION, reflecting a similar finding in SNLI, while long sentences are slightly more likely to be labeled ENTAILMENT.

None of the baseline models perform substantially better on any tagged set than they do on the corpus overall, with average model accuracies on sentences containing specific tags falling within

[13]Because their high frequency in the corpus, extremely common triggers like *the* were excluded from this tag.

about 3 points of overall averages. Using baseline model test accuracy overall as a metric (see Table 4), our baseline models had the most trouble on sentences containing comparatives or superlatives (losing 3-4 points each). Despite the fact that 17% of sentence pairs in the corpus contained at least one instance of comparative or superlative, our baseline models don't utilize the information present in these sentences to predict the correct label for the pair, although presence of a comparative or superlative is slightly more predictive of a NEUTRAL label.

Moreover, the baseline models perform below average on discourse markers, such as *despite* and *however*, losing roughly 2 to 3 points each. Unsurprisingly, the attention-based ESIM model performs better than the other two on sentences with greater than 20 words. Additionally, our baseline models do show slight improvements in accuracy on negation, suggesting that they may be tracking it as a predictor of CONTRADICTION.

5 Conclusion

Natural language inference makes it easy to judge the degree to which neural network models for sentence understanding capture the full meanings for natural language sentences. Existing NLI datasets like SNLI have facilitated substantial advances in modeling, but have limited headroom and coverage of the full diversity of meanings expressed in English. This paper presents a new dataset that offers dramatically greater linguistic difficulty and diversity, and also serves as a benchmark for cross-genre domain adaptation.

Our new corpus, MultiNLI, improves upon SNLI in its empirical coverage—because it includes a representative sample of text and speech from ten different genres, as opposed to just simple image captions—and its difficulty, containing a much higher percentage of sentences tagged with one or more elements from our tag set of thirteen difficult linguistic phenomena. This greater diversity is reflected in the dramatically lower baseline model performance on MultiNLI than on SNLI (see Table 5) and comparable inter-annotator agreement, suggesting that MultiNLI has a lot of headroom remaining for future work.

The MultiNLI corpus was first released in draft form in the first half of 2017, and in the time since its initial release, work by others (Conneau et al., 2017) has shown that NLI can also be an effective source task for pre-training and transfer learning in the context of sentence-to-vector models, with models trained on SNLI and MultiNLI substantially outperforming all prior models on a suite of established transfer learning benchmarks. We hope that this corpus will continue to serve for many years as a resource for the development and evaluation of methods for sentence understanding.

Acknowledgments

This work was made possible by a Google Faculty Research Award. SB also gratefully acknowledges support from Tencent Holdings and Samsung Research. We also thank George Dahl, the organizers of the RepEval 2016 and RepEval 2017 workshops, Andrew Drozdov, Angeliki Lazaridou, and our other NYU colleagues for help and advice.

References

Isaac Asimov. May 1952. *Youth*. Space Science Fiction Magazine, Republic Features Syndicate, Inc.

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. 2007. Analysis of representations for domain adaptation. In *Advances in neural information processing systems*. pages 137–144.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 120–128. http://www.aclweb.org/anthology/W06-1615.

Johan Bos and Katja Markert. 2005. Recognising textual entailment with logical inference. In *Proceedings of the 2005 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 628–635. http://www.aclweb.org/anthology/H05-1079.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 632–642. https://doi.org/10.18653/v1/D15-1075.

Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1:*

Long Papers). Association for Computational Linguistics, pages 1466–1477. https://doi.org/10.18653/v1/P16-1139.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1657–1668. https://doi.org/10.18653/v1/P17-1152.

Agatha Christie. 1921. *Mysterious Affair at Styles*. The Bodley Head.

Agatha Christie. 1922. *The Secret Adversary*. Dodd, Mead.

Cleo Condoravdi, Dick Crouch, Valeria de Paiva, Reinhard Stolle, and Daniel G. Bobrow. 2003. Entailment, intensionality and text understanding. In *Proceedings of the Human Language Technology-North American Association for Computational Linguistics 2003 Workshop on Text Meaning*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364* .

Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Johan Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Steve Pulman Ted Briscoe Holger Maier Poesio, and Karsten Konrad. 1996. Using the framework. Technical report, Technical Report LRE 62-051 D-16, The FraCaS Consortium.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In *Machine learning challenges. Evaluating predictive uncertainty, visual object classification, and recognising textual entailment*, Springer, pages 177–190.

Hal Daume III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research* 26:101–126. https://doi.org/doi:10.1613/jair.1872.

Marie-Catherine de Marneffe, Anna N. Rafferty, and Christopher D. Manning. 2008. Finding contradictions in text. In *Proceedings of Association for Computational Linguistics-08: Human Language Technology*. Association for Computational Linguistics, pages 1039–1047. http://www.aclweb.org/anthology/P08-1118.

Lester Del Rey. 1973. *The Sky Is Falling*. Ace Books.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2014. DeCAF: A deep convolutional activation feature for generic visual recognition. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Ben Essex. 2016. *Living History*. CreateSpace Independent Publishing Platform.

Charles Fillmore, Nancy Ide, Daniel Jurafsky, and Catherine Macleod. 1998. An American National Corpus: A proposal. In *Proceedings of the First Annual Conference on Language Resources and Evaluation*. pages 965–969.

Yaroslav Fyodorov, Yoad Winter, and Nissim Francez. 2000. A natural logic inference system. In *Proceedings of the 2nd Workshop on Inference in Computational Semantics*.

Irene Heim and Angelika Kratzer. 1998. *Semantics in generative grammar*. Blackwell Publishers.

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

Nancy Ide and Catherine Macleod. 2001. The American National Corpus: A standardized resource of American English. In *Proceedings of Corpus Linguistics*. Lancaster University Centre for Computer Corpus Research on Language, volume 3, pages 1–7.

Nancy Ide and Keith Suderman. 2006. Integrating linguistic resources: The national corpus model. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC)*. European Language Resources Association (ELRA). http://www.aclweb.org/anthology/L06-1138.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proc. ACL*. https://doi.org/10.3115/1075096.1075150.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105.

Bill MacCartney and Christopher D Manning. 2009. An extended model of natural logic. In *Proceedings of the of the Eighth International Conference on Computational Semantics*. pages 140–156.

Catherine Macleod, Nancy Ide, and Ralph Grishman. 2000. The American National Corpus: A standardized resource for American English. In *Conference on Language Resources and Evaluation (LREC)*.

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

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014a. Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Association for Computational Linguistics, pages 1–8. https://doi.org/10.3115/v1/S14-2001.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014b. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Twelfth International Conference on Language Resources and Evaluation (LREC)*.

Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016a. How transferable are neural networks in NLP applications? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 479–489. https://doi.org/10.18653/v1/D16-1046.

Lili Mou, Men Rui, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016b. Natural language inference by tree-based convolution and heuristic matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 130–136. https://doi.org/10.18653/v1/P16-2022.

Tsendsuren Munkhdalai and Hong Yu. 2017. Neural semantic encoders. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, pages 397–407. http://www.aclweb.org/anthology/E17-1038.

Nick Name. 2008. *Password Incorrect*. Published online. Author also known as Piotr Kowalczyk.

Nikita Nangia, Adina Williams, Angeliki Lazaridou, and Samuel R Bowman. 2017. The repeval 2017 shared task: Multi-genre natural language inference with sentence representations. In *Proceedings of RepEval 2017: The Second Workshop on Evaluating Vector Space Representations for NLP*.

Andre Norton. 1962. *The Rebel Spurs*. The World Publishing Company, Cleveland & New York.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 2249–2255. https://doi.org/10.18653/v1/D16-1244.

Nanyun Peng and Mark Dredze. 2017. Multi-task domain adaptation for sequence tagging. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*. pages 91–100.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 1532–1543. https://doi.org/10.3115/v1/D14-1162.

Henry Beam Piper. 1953. *Murder In the Gun Room*. H.B. Piper, New York.

Rafael Sabatini. 1922. *Captain Blood*. Houghton Mifflin Company.

Philippe Schlenker. 2016. *The Cambridge Handbook of Formal Semantics*, Cambridge University Press, chapter The Semantics/Pragmatics Interface, pages 664–727. https://doi.org/10.1017/CBO9781139236157.023.

Michael Shea. 2008. *Seven Swords*. Published online.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)* 15:1929–1958.

Robert Stalnaker. 1974. *Semantics and Philosophy*, New York University Press, chapter Pragmatic Presupposition, pages 329–355.

Anna Szabolcsi. 2010. *Quantification*. Cambridge University Press.

Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with LSTM. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 1442–1451. https://doi.org/10.18653/v1/N16-1170.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association of Computational Linguistics* 2:67–78. http://www.aclweb.org/anthology/Q14-1006.

Matthew D. Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*. pages 818–833.

Filling Missing Paths: Modeling Co-occurrences of Word Pairs and Dependency Paths for Recognizing Lexical Semantic Relations

Koki Washio and **Tsuneaki Kato**
Department of Language and Information Sciences
Graduate School of Arts and Sciences
The University of Tokyo
3-8-1, Komaba, Meguroku, Tokyo 153-8902 Japan
{kokiwashio@g.ecc, kato@boz.c}.u-tokyo.ac.jp

Abstract

Recognizing lexical semantic relations between word pairs is an important task for many applications of natural language processing. One of the mainstream approaches to this task is to exploit the lexico-syntactic paths connecting two target words, which reflect the semantic relations of word pairs. However, this method requires that the considered words co-occur in a sentence. This requirement is hardly satisfied because of Zipf's law, which states that most content words occur very rarely. In this paper, we propose novel methods with a neural model of $P(path|w_1, w_2)$ to solve this problem. Our proposed model of $P(path|w_1, w_2)$ can be learned in an unsupervised manner and can generalize the co-occurrences of word pairs and dependency paths. This model can be used to augment the path data of word pairs that do not co-occur in the corpus, and extract features capturing relational information from word pairs. Our experimental results demonstrate that our methods improve on previous neural approaches based on dependency paths and successfully solve the focused problem.

1 Introduction

The semantic relations between words are important for many natural language processing tasks, such as recognizing textual entailment (Dagan et al., 2010) and question answering (Yang et al., 2017). Moreover, these relations have been also used as features for neural methods in machine translation (Sennrich and Haddow, 2016) and relation extraction (Xu et al., 2015). This type of information is provided by manually-created semantic taxonomies, such as WordNet (Fellbaum, 1998). However, these resources are expensive to expand manually and have limited domain coverage. Thus, the automatic detection of lexico-semantic relations has been studied for several decades.

One of the most popular approaches is based on patterns that encode a specific kind of relationship (synonym, hypernym, etc.) between adjacent words. This type of approach is called a path-based method. Lexico-syntactic patterns between two words provide information on semantic relations. For example, if we see the pattern, "animals such as a dog" in a corpus, we can infer that *animal* is a hypernym of *dog*. On the basis of this assumption, Hearst (1992) detected the hypernymy relation of two words from a corpus based on several handcrafted lexico-syntactic patterns, e.g., *X such as Y*. Snow et al. (2004) used as features indicative dependency paths, in which target word pairs co-occurred, and trained a classifier with data to detect hypernymy relations.

In recent studies, Shwartz et al. (2016) proposed a neural path-based model that encoded dependency paths between two words into low-dimensional dense vectors with recurrent neural networks (RNN) for hypernymy detection. This method can prevent sparse feature space and generalize indicative dependency paths for detecting lexico-semantic relations. Their model outperformed the previous state-of-the-art path-based method. Moreover, they demonstrated that these dense path representations capture complementary information with word embeddings that contain individual word features. This was indicated by the experimental result that showed the combination of path representations and word embeddings improved classification performance. In addition, Shwartz and Dagan (2016) showed that the neural path-based approach, combined with word embeddings, is effective in recognizing multiple semantic relations.

Although path-based methods can capture the relational information between two words, these methods can obtain clues only for word pairs that

Proceedings of NAACL-HLT 2018, pages 1123–1133
New Orleans, Louisiana, June 1 - 6, 2018.

co-occur in a corpus. Even with a very large corpus, it is almost impossible to observe a co-occurrence of arbitrary word pairs. Thus, path-based methods are still limited in terms of the number of word pairs that are correctly classified.

To address this problem, we propose a novel method with modeling $P(path|w_1, w_2)$ in a neural unsupervised manner, where w_1 and w_2 are the two target words, and $path$ is a dependency path that can connect the joint co-occurrence of w_1 and w_2. A neural model of $P(path|w_1, w_2)$ can generalize co-occurrences of word pairs and dependency paths, and infer plausible dependency paths which connect two words that do not co-occur in a corpus. After unsupervised learning, this model can be used in two ways:

- Path data augmentation through predicting dependency paths that are most likely to co-occur with a given word pair.
- Feature extraction of word pairs, capturing the information of dependency paths as contexts where two words co-occur.

While previous supervised path-based methods used only a small portion of a corpus, combining our models makes it possible to use an entire corpus for learning process.

Experimental results for four common datasets of multiple lexico-semantic relations show that our methods improve the classification performance of supervised neural path-based models.

2 Background

2.1 Supervised Lexical Semantic Relation Detection

Supervised lexical semantic relation detection represents word pairs (w_1, w_2) as feature vectors $\boldsymbol{v}$ and trains a classifier with these vectors based on training data. For word pair representations $\boldsymbol{v}$, we can use the distributional information of each word and path information in which two words co-occur.

Several methods exploit word embeddings (Mikolov et al., 2013; Levy and Goldberg, 2014; Pennington et al., 2014) as distributional information. These methods use a combination of each word's embeddings, such as vector concatenation (Baroni et al., 2012; Roller and Erk, 2016) or vector difference (Roller et al., 2014; Weeds et al., 2014; Vylomova et al., 2016), as word pair representations. While these distributional supervised methods do not require co-occurrences of two words in a sentence, Levy et al. (2015) notes that these methods do not learn the relationships between two words but rather the separate property of each word, i.e., whether or not each word tends to have a target relation.

In contrast, supervised path-based methods can capture relational information between two words. These methods represent a word pair as the set of lexico-syntactic paths, which connect two target words in a corpus (Snow et al., 2004). However, these methods suffer from sparse feature space, as they cannot capture the similarity between indicative lexico-syntactic paths, e.g., *X is a species of Y* and *X is a kind of Y*.

2.2 Neural Path-based Method

A neural path-based method can avoid the sparse feature space of the previous path-based methods (Shwartz et al., 2016; Shwartz and Dagan, 2016). Instead of treating an entire dependency path as a single feature, this model encodes a sequence of edges of a dependency path into a dense vector using a long short-term memory network (LSTM) (Hochreiter and Schmidhuber, 1997).

A dependency path connecting two words can be extracted from the dependency tree of a sentence. For example, given the sentence "A dog is a mammal," with *X* = *dog* and *Y* = *mammal*, the dependency path connecting the two words is `X/NOUN/nsubj/> be/VERB/ROOT/- Y/NOUN/attr/<`. Each edge of a dependency path is composed of a lemma, part of speech (POS), dependency label, and dependency direction.

Shwartz et al. (2016) represents each edge as the concatenation of its component embeddings:

$$\boldsymbol{e} = [\boldsymbol{v}_l; \boldsymbol{v}_{pos}; \boldsymbol{v}_{dep}; \boldsymbol{v}_{dir}] \quad (1)$$

where $\boldsymbol{v}_l, \boldsymbol{v}_{pos}, \boldsymbol{v}_{dep}$,and $\boldsymbol{v}_{dir}$ represent the embedding vectors of the lemma, POS, dependency label, and dependency direction respectively. This edge vector $\boldsymbol{e}$ is an input of the LSTM at each time step. Here, $\boldsymbol{h_t}$, the hidden state at time step t, is abstractly computed as:

$$\boldsymbol{h}_t = LSTM(\boldsymbol{h}_{t-1}, \boldsymbol{e}_t) \quad (2)$$

where $LSTM$ computes the current hidden state given the previous hidden state $\boldsymbol{h}_{t-1}$ and the current input edge vector $\boldsymbol{e}_t$ along with the LSTM architecture. The final hidden state vector $\boldsymbol{o}_p$ is

treated as the representation of the dependency path p.

When classifying a word pair (w_1, w_2), the word pair is represented as the average of the dependency path vectors that connect two words in a corpus:

$$\begin{aligned} \boldsymbol{v}_{(w_1,w_2)} &= \boldsymbol{v}_{paths(w_1,w_2)} \\ &= \frac{\sum_{p \in paths(w_1,w_2)} f_{p,(w_1,w_2)} \cdot \boldsymbol{o}_p}{\sum_{p \in paths(w_1,w_2)} f_{p,(w_1,w_2)}} \end{aligned} \quad (3)$$

where $paths(w_1, w_2)$ is the set of dependency paths that connects w_1 and w_2 in the corpus, and $f_{p,(w_1,w_2)}$ is the frequency of p in $paths(w_1, w_2)$. The final output of the network is calculated as follows:

$$\boldsymbol{y} = softmax(\boldsymbol{W}\boldsymbol{v}_{(w_1,w_2)} + \boldsymbol{b}) \quad (4)$$

where $\boldsymbol{W} \in \mathbb{R}^{|c| \times d}$ is a linear transformation matrix, $\boldsymbol{b} \in \mathbb{R}^{|c|}$ is a bias parameter, $|c|$ is the number of the output class, and d is the size of $\boldsymbol{v}_{(w_1,w_2)}$.

This neural path-based model can be combined with distributional methods. Shwartz et al. (2016) concatenated $\boldsymbol{v}_{paths(w_1,w_2)}$ to the word embeddings of w_1 and w_2, redefining $\boldsymbol{v}_{(w_1,w_2)}$ as:

$$\boldsymbol{v}_{(w_1,w_2)} = [\boldsymbol{v}_{w_1}; \boldsymbol{v}_{paths(w_1,w_2)}; \boldsymbol{v}_{w_2}] \quad (5)$$

where $\boldsymbol{v}_{w_1}$ and $\boldsymbol{v}_{w_2}$ are word embeddings of w_1 and w_2, respectively. This integrated model, named LexNET, exploits both path information and distributional information, and has high generalization performance for lexical semantic relation detection.

2.3 Missing Path Problem

All path-based methods, including the neural ones, suffer from data sparseness as they depend on word pair co-occurrences in a corpus. However, we cannot observe all co-occurrences of semantically related words even with a very large corpus because of Zipf's law, which states that the frequency distribution of words has a long tail; in other words, most words occur very infrequently (Hanks, 2009). In this paper, we refer to this phenomenon as the missing path problem.

This missing path problem leads to the fact that path-based models cannot find any clues for two words that do not co-occur. Thus, in the neural path-based method, $paths(w_1, w_2)$ for these word pairs is padded with an empty path, like `UNK-lemma/UNK-POS/UNK-dep/UNK-dir`. However, this process makes path-based classifiers unable to distinguish between semantically-related pairs with no co-occurrences and those that have no semantic relation.

In an attempt to solve this problem, Necsulescu et al. (2015) proposed a method that used a graph representation of a corpus. In this graph, words and dependency relations were denoted as nodes and labeled directed edges, respectively. From this graph representation, paths linking two target words can be extracted through bridging words, even if the two words do not co-occur in the corpus. They represent word pairs as the sets of paths linking word pairs on the graph and train a support vector machine classifier with training data, thereby improving recall. However, the authors reported that this method still suffered from data sparseness.

In this paper, we address this missing path problem, which generally restricts path-based methods, by neural modeling $P(path|w_1, w_2)$.

3 Method

We present a novel method for modeling $P(path|w_1, w_2)$. The purpose of this method is to address the missing path problem by generalizing the co-occurrences of word pairs and dependency paths. To model $P(path|w_1, w_2)$, we used the context-prediction approach (Collobert and Weston, 2008; Mikolov et al., 2013; Levy and Goldberg, 2014; Pennington et al., 2014), which is a widely used method for learning word embeddings. In our proposed method, word pairs and dependency paths are represented as embeddings that are updated with unsupervised learning through predicting $path$ from w_1 and w_2 (Section 3.1).

After the learning process, our model can be used to (1) augment path data by predicting the plausibility of the co-occurrence of two words and a dependency path (Section 3.2); and to (2) extract useful features from word pairs, which reflect the information of co-occurring dependency paths (Section 3.3).

3.1 Unsupervised Learning

There are many possible ways to model $P(path|w_1, w_2)$. In this paper, we present a straightforward and efficient architecture, similar to the skip-gram with negative sampling (Mikolov et al., 2013).

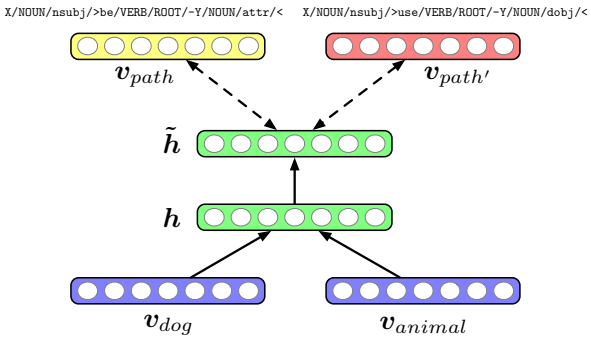


Figure 1: An illustration of our network for modeling $P(path|w_1, w_2)$. Given a word pair $(dog, animal)$, our model makes $\tilde{\boldsymbol{h}}$ of $(dog, animal)$ similar to $\boldsymbol{v}_{path}$ of the observed co-occurring dependency path `X/NOUN/nsubj/> be/VERB/ROOT/- Y/NOUN/attr/<` and dissimilar to $\boldsymbol{v}_{path'}$ of the unobserved paths, such as `X/NOUN/nsubj/> use/VERB/ROOT/- Y/NOUN/dobj/<`, through unsupervised learning.

Figure 1 depicts our network structure, which is described below.

Data and Network Architecture

We are able to extract many triples $(w_1, w_2, path)$ from a corpus after dependency parsing. We denote a set of these triples as D. These triples are the instances used for the unsupervised learning of $P(path|w_1, w_2)$. Given $(w_1, w_2, path)$, our model learns through predicting $path$ from w_1 and w_2.

We encode word pairs into dense vectors as follows:

$$\boldsymbol{h}_{(w1,w2)} = tanh(\boldsymbol{W}_1[\boldsymbol{v}_{w_1}; \boldsymbol{v}_{w_2}] + \boldsymbol{b}_1) \quad (6)$$

$$\tilde{\boldsymbol{h}}_{(w_1,w_2)} = tanh(\boldsymbol{W}_2\boldsymbol{h}_{(w1,w2)} + \boldsymbol{b}_2) \quad (7)$$

where $[\boldsymbol{v}_{w_1}; \boldsymbol{v}_{w_2}]$ is the concatenation of the word embeddings of w_1 and w_2; $\boldsymbol{W}_1$, $\boldsymbol{b}_1$, $\boldsymbol{W}_2$, and $\boldsymbol{b}_2$ are the parameter matrices and bias parameters of the two linear transformations; and $\tilde{\boldsymbol{h}}_{(w_1,w_2)}$ is the representation of the word pair.

We associate each $path$ with the embedding $\boldsymbol{v}_{path}$, initialized randomly. While we use a simple way to represent dependency paths in this paper, LSTM can be used to encode each path in the way described in Section 2.2. If LSTM is used, learning time increases but similarities among paths will be captured.

Objective

We used the negative sampling objective for training (Mikolov et al., 2013). Given the word pair representations $\tilde{\boldsymbol{h}}_{(w_1,w_2)}$ and the dependency path representations $\boldsymbol{v}_{path}$, our model was trained to distinguish real $(w_1, w_2, path)$ triples from incorrect ones. The log-likelihood objective is as follows:

$$L = \sum_{(w_1,w_2,path)\in D} \log \sigma(\boldsymbol{v}_{path} \cdot \tilde{\boldsymbol{h}}_{(w_1,w_2)}) + \sum_{(w_1,w_2,path')\in D'} \log \sigma(-\boldsymbol{v}_{path'} \cdot \tilde{\boldsymbol{h}}_{(w_1,w_2)}) \quad (8)$$

where, D' is the set of randomly generated negative samples. We constructed n triples $(w_1, w_2, path')$ for each $(w_1, w_2, path) \in D$, where n is a hyperparameter and each $path'$ is drawn according to its unigram distribution raised to the $3/4$ power. The objective L was maximized using the stochastic gradient descent algorithm.

3.2 Path Data Augmentation

After the unsupervised learning described above, our model of $P(path|w_1, w_2)$ can assign the plausibility score $\sigma(\boldsymbol{v}_{path} \cdot \tilde{\boldsymbol{h}}_{(w_1,w_2)})$ to the co-occurrences of a word pair and a dependency path. We can then append the plausible dependency paths to $paths(w_1, w_2)$, the set of dependency paths that connects w_1 and w_2 in the corpus, based on these scores.

We calculate the score of each dependency path given $(X = w_1, Y = w_2)$ and append the k dependency paths with the highest scores to $paths(w_1, w_2)$, where k is a hyperparameter. We perform the same process given $(X = w_2, Y = w_1)$ with the exception of swapping the `X` and `Y` in the dependency paths to be appended. As a result, we add $2k$ dependency paths to the set of dependency paths for each word pair. Through this data augmentation, we can obtain plausible dependency paths even when word pairs do not co-occur in the corpus. Note that we retain the empty path indicators of $paths(w_1, w_2)$, as we believe that this information contributes to classifying two unrelated words.

3.3 Feature Extractor of Word Pairs

Our model can be used as a feature extractor of word pairs. We can exploit $\tilde{\boldsymbol{h}}_{(w_1,w_2)}$ to represent the word pair (w_1, w_2). This representation captures the information of co-occurrence dependency paths of (w_1, w_2) in a generalized fashion. Thus, $\tilde{\boldsymbol{h}}_{(w_1,w_2)}$ is used to construct the pseudo-path representation $\boldsymbol{v}_{p-paths(w_1,w_2)}$. With our model, we represent the word pair (w_1, w_2) as

datasets	relations
K&H+N	hypernym, meronym, co-hyponym, random
BLESS	hypernym, meronym, co-hyponym, random
ROOT09	hypernym, co-hyponym, random
EVALution	hypernym, meronym, attribute, synonym, antonym, holonym, substance meronym

Table 1: The relation types in each dataset.

follows:

$$\boldsymbol{v}_{p-paths(w_1,w_2)} = [\tilde{\boldsymbol{h}}_{(w_1,w_2)}; \tilde{\boldsymbol{h}}_{(w_2,w_1)}] \quad (9)$$

This representation can be used for word pair classification tasks, such as lexical semantic relation detection.

4 Experiment

In this section, we examine how our method improves path-based models on several datasets for recognizing lexical semantic relations. In this paper, we focus on major noun relations, such as hypernymy, co-hypernymy, and meronymy.

4.1 Dataset

We relied on the datasets used in Shwartz and Dagan (2016); K&H+N (Necsulescu et al., 2015). BLESS (Baroni and Lenci, 2011), EVALution (Santus et al., 2015), and ROOT09 (Santus et al., 2016). These datasets were constructed with knowledge resources (e.g., WordNet, Wikipedia), crowd-sourcing, or both. We used noun pair instances of these datasets.[1] Table 1 displays the relations in each dataset used in our experiments. Note that we removed the two relations *Entails* and *MemberOf* with few instances from EVALution following Shwartz and Dagan (2016). For data splitting, we used the presplitted train/val/test sets from Shwartz and Dagan (2016) after removing all but the noun pairs from each set.

4.2 Corpus and Dependency Parsing

For path-based methods, we used the June 2017 Wikipedia dump as a corpus and extracted $(w_1, w_2, path)$ triples of noun pairs using the dependency parser of spaCy[2] to construct D. In this process, w_1 and w_2 were lemmatized with spaCy. We only used the dependency paths which occurred at least five times following the implementation of Shwartz and Dagan (2016).[3]

datasets	instances	instances with paths	proportion
K&H+N	57509	8866	15.4%
BLESS	14558	8775	60.3%
ROOT09	8602	6582	76.5%
EVALution	3240	3199	98.7%

Table 2: The number and proportion of instances whose dependency path is obtained from each dataset

Table 2 displays the number of instances and the proportion of the instances for which at least one dependency path was obtained.

4.3 Baseline

We conducted experiments with three neural path-based methods. The implementation details below follow those in Shwartz and Dagan (2016). We implemented all models using Chainer.[4]

Neural Path-Based Model (NPB). We implemented and trained the neural path-based model described in Section 2.2. We used the two-layer LSTM with 60-dimensional hidden units. An input vector was composed of embedding vectors of the lemma (50 dims), POS (4 dims), dependency label (5 dims), and dependency direction (1 dim). Regularization was applied by a dropout on each of the components embeddings (Iyyer et al., 2015; Kiperwasser and Goldberg, 2016).

LexNET. We implemented and trained the integrated model LexNET as described in Section 2.2. The LSTM details are the same as in the NPB model.

LexNET_h. This model, a variant of LexNET, has an additional hidden layer between the output layer and $\boldsymbol{v}_{(w_1,w_2)}$ of Equation (5). Because of this additional hidden layer, this model can take into account the interaction of the path information

[1] We focused only noun pairs to shorten the unsupervised learning time, though this restriction is not necessary for our methods and the unsupervised learning is still tractable.

[2] https://spacy.io

[3] https://github.com/vered1986/LexNET

[4] https://chainer.org

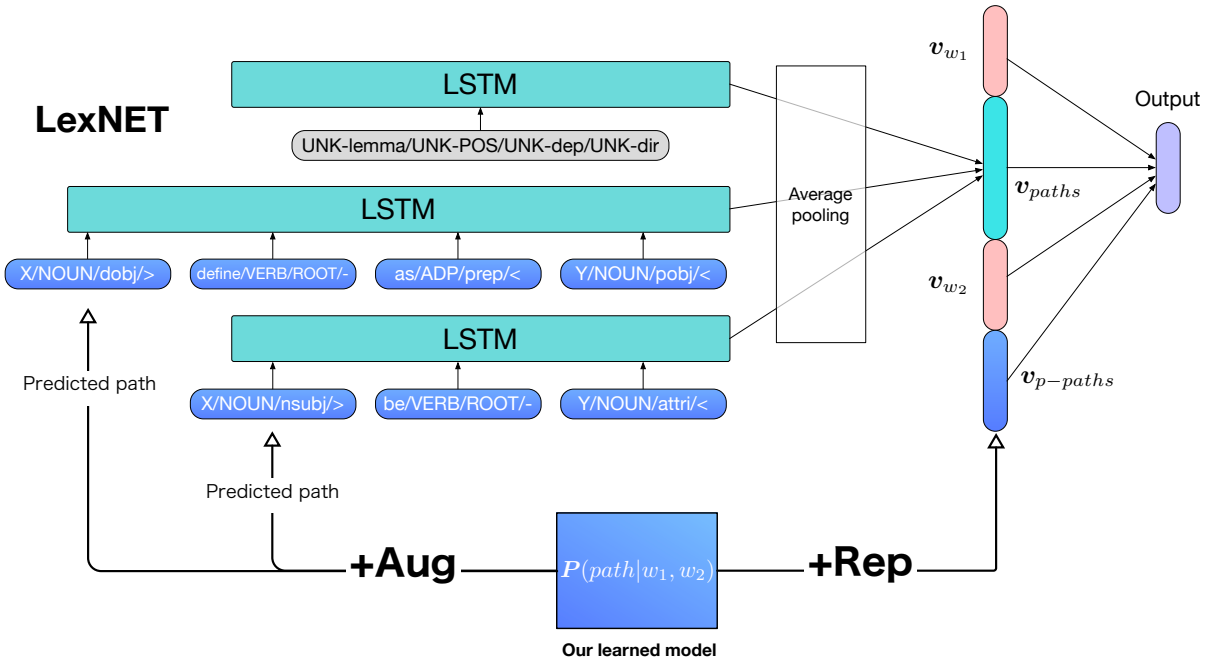


Figure 2: Illustration of +Aug and +Rep applied to LexNET. +Aug predicts plausible paths from two word embeddings, and these paths are fed into the LSTM path encoder. +Rep concatenates the pseudo-path representation $\boldsymbol{v}_{p-paths(w_1,w_2)}$ with the penultimate layer of LexNET

and distributional information of two word embeddings. The size of the additional hidden layer was set to 60.

Following Shwartz and Dagan (2016), we optimized each model using Adam (whose learning rate is 0.001) while tuning the dropout rate dr among $\{0.0, 0.2, 0.4\}$ on the validation set. The minibatch size was set to 100.

We initialized the lemma embeddings of LSTM and concatenated the word embeddings of LexNET with the pretrained 50-dimensional GloVe vector.[5] Training was stopped if performance on the validation set did not improve for seven epochs, and the best model for test evaluation was selected based on the score of the validation set.

4.4 Our Method

We implemented and trained our model of $P(path|w_1, w_2)$, described in Section 3.1, as follows. We used the most frequent 30,000 paths connecting nouns as the context paths for unsupervised learning. We initialized word embeddings with the same pretrained GloVe vector as the baseline models. For unsupervised learning data, we extracted $(w_1, w_2, path)$, whose w_1 and w_2 are included in the vocabulary of the GloVe vector, and whose $path$ is included in the context paths, from D. The number of these triples was 217,737,765.

We set the size of $\boldsymbol{h}_{(w_1,w_2)}$, $\tilde{\boldsymbol{h}}_{(w_1,w_2)}$, and $\boldsymbol{v}_{path}$ for context paths to 100. The negative sampling size n was set to 5. We trained our model for five epochs using Adam (whose learning rate is 0.001). The minibatch size was 100. To preserve the distributional regularity of the pretrained word embeddings, we did not update the input word embeddings during the unsupervised learning.

With our trained model, we applied the two methods described in Section 3.2 and 3.3 to the NPB and LexNET models as follows:

+Aug. We added the most plausible $2k$ paths to each $paths(w_1, w_2)$ as in Section 3.2. We tuned $k \in \{1, 3, 5\}$ on the validation set.

+Rep. We concatenated $\boldsymbol{v}_{p-paths(w_1,w_2)}$ in Equation (9) with the penultimate layer. To focus on the pure contribution of unsupervised learning, we did not update this component during supervised learning.

Figure 2 illustrates +Aug and +Rep applied to LexNET in the case where the two target words, w_1 and w_2, do not co-occur in the corpus.

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

Models	K&H+N	BLESS	ROOT09	EVALution
NPB	0.495	0.773	0.731	0.463
NPB+Aug	**0.897**	**0.842**	**0.778**	**0.489**

Table 3: Classification performance of the neural path-based model (NPB) and that with the path data augmentation (NPB+Aug).

Models	K&H+N	BLESS	ROOT09	EVALution
LexNET	0.969	0.922	0.776	0.539
LexNET_h	0.968	0.927	0.810	0.540
LexNET+Aug	**0.970**	0.927	0.806	0.545
LexNET+Rep	**0.970**	**0.944**	**0.832**	0.565
LexNET+Aug+Rep	0.969	0.942	0.820	**0.567**

Table 4: Classification performance of the integrated model, LexNET and LexNET_h, and those with our methods, +Aug and +Rep.

5 Result

In this section we examine how our methods improved the baseline models. Following the previous research (Shwartz and Dagan, 2016), the performance metrics were the "averaged" $F1$ of scikit-learn (Pedregosa et al., 2011), which computes the $F1$ for each relation, and reports their average weighted by the number of true instances for each relation.

5.1 Path-based Model and Path Data Augmentation

We examined whether or not our path data augmentation method +Aug contributes to the neural path-based method. The results are displayed in Table 3.

Applying our path data augmentation method improved the classification performance on each dataset. Especially for K&H+N, the large dataset where the three-fourths of word pairs had no paths, our method significantly improved the performance. This result shows that our path data augmentation effectively solves the missing path problem. Moreover, the model with our method outperforms the baseline on EVALution, in which nearly all word pairs co-occurred in the corpus. This indicates that the predicted paths provide useful information and enhance the path-based classification. We examine the paths that were predicted by our model of $P(path|w_1, w_2)$ in Section 6.1.

5.2 Integrated Model and Our Methods

We investigated how our methods using modeling $P(path|w_1, w_2)$ improved the baseline integrated model, LexNET. Table 4 displays the results.

Our proposed methods, +Aug and +Rep, improved the performance of LexNET on each dataset.[6] Moreover, the best score on each dataset was achieved by the model to which our methods were applied. These results show that our methods are also effective with the integrated models based on path information and distributional information.

The table also shows that LexNET+Rep outperforms LexNET_h, though the former has fewer parameters to be tuned during the supervised learning than the latter. This indicates that the word pair representations of our model capture information beyond the interaction of two word embeddings. We investigate the properties of our word pair representation in Section 6.2.

Finally, We found that applying both methods did not necessarily yield the best performance. A possible explanation for this is that applying both methods is redundant, as both +Aug and +Rep depend on the same model of $P(path|w_1, w_2)$.

6 Analysis

In this section, we investigate the properties of the predicted dependency paths and word pair representations of our model.

6.1 Predicted Dependency Paths

We extracted the word pairs of BLESS without co-occurring dependency paths and predicted the

[6]The improvement for K&H+N is smaller than those for the others. We think this owes to most instances of this dataset being correctly classified only by distributional information. This view is supported by Shwartz and Dagan (2016), in which LexNET hardly outperformed a distributional method for this dataset.

Word pair	Relation	Predicted paths
X = "jacket", Y = "commodity"	hypernym	**X/NOUN/nsubj/> be/VERB/ROOT/- shooter/NOUN/attr/< Y/NOUN/compound/<**
		X/NOUN/nsubj/> be/VERB/ROOT/- Y/NOUN/attr/< manufacture/VERB/acl/<
		red/ADJ/amod/< X/NOUN/nsubj/> be/VERB/ROOT/- Y/NOUN/attr/<
X = "goose", Y = "creature"	hypernym	**X/NOUN/nsubj/> be/VERB/ROOT/- species/NOUN/attr/< of/ADP/prep/< Y/NOUN/pobj/< of/ADP/prep/>**
		X/NOUN/nsubj/> be/VERB/ROOT/- specie/NOUN/attr/< of/ADP/prep/< Y/NOUN/pobj/< in/ADP/prep/>
		X/NOUN/pobj/> of/ADP/ROOT/- bird/NOUN/pobj/< Y/NOUN/conj/<
X = "owl", Y = "rump"	meronym	**X/NOUN/ROOT/- represent/VERB/relcl/< Y/NOUN/nsubj/<**
		X/NOUN/nsubj/> have/VERB/ROOT/- Y/NOUN/dobj/< be/VERB/relcl/>
		all/DET/det/< X/NOUN/nsubj/> have/VERB/ROOT/- Y/NOUN/dobj/<
X = "mug", Y = "plastic"	meronym	X/NOUN/pobj/> of/ADP/ROOT/- arm/NOUN/pobj/< Y/NOUN/conj/<
		the/DET/det/< X/NOUN/nsubjpass/> make/VERB/ROOT/- from/ADP/prep/< Y/NOUN/pobj/<
		X/NOUN/compound/> gun/NOUN/ROOT/- Y/NOUN/appos/<
X = "carrot", Y = "beans"	co-hyponym	**X/NOUN/compound/> leaf/NOUN/ROOT/- Y/NOUN/conj/<**
		X/NOUN/compound/> specie/NOUN/ROOT/- Y/NOUN/conj/<
		X/NOUN/dobj/> use/VERB/ROOT/- in/ADP/prep/< Y/NOUN/pobj/< of/ADP/prep/>
X = "cello", Y = "kazoo"	co-hyponym	**X/NOUN/dobj/> play/VERB/ROOT/- guitar/NOUN/dobj/< Y/NOUN/conj/<**
		X/NOUN/pobj/> for/ADP/ROOT/- piano/NOUN/pobj/< Y/NOUN/conj<
		X/NOUN/pobj/> on/ADP/ROOT/- drum/NOUN/pobj/< Y/NOUN/conj/<

Table 5: Predicted paths with our model for a word pair of each relation in BLESS.

plausible dependency paths of those pairs with our model of $P(path|w_1, w_2)$. The examples are displayed in Table 5 at the top three paths. We used the bold style for the paths that we believe to be indicative or representative for a given relationship.

Our model predicted plausible and indicative dependency paths for each relation, although the predicted paths also contain some implausible or unindicative ones. For hypernymy, our model predicted variants of the is-a path according to domains, such as *X is Y manufactured* in the clothing domain and *X is a species of Y* in the animal domain. For $(owl, rump)$, which is a meronymy pair, the top predicted path was *X that Y represent*. This is not plausible for $(owl, rump)$ but is indicative for meronymy, particularly member-of relations. Moreover, domain-independent paths which indicate meronymy, such as *all X have Y*, were predicted. For $(mug, plastic)$, one of the predicted paths, *X is made from Y*, is also a domain-independent indicative path for meronymy. For co-hypernymy, our model predicted domain-specific paths, which indicate that two nouns are of the same kind. For examples, given *X leaf and Y* and *X specie and Y* of $(carrot, beans)$, we can infer that both X and Y are plants or vegetables. Likewise, given *play X, guitar, and Y* of $(cello, kazoo)$, we can infer that both X and Y are musical instruments. These examples show that our path data augmentation is effective for the missing path problem and enhances path-based models.

6.2 Visualizing Word Pair Representations

We visualized the word pair representations $\boldsymbol{v}_{p-paths(w_1,w_2)}$ to examine their specific properties. In BLESS, every pair was annotated with 17 domain class labels. For each domain, we reduced the dimensionality of the representations using t-SNE (Maaten and Hinton, 2008) and plotted the data points of the hypernyms, co-hyponyms, and meronyms. We compared our representations with the concatenation of two word embeddings (pre-trained 50-dimensional GloVe). The examples are displayed in Figure 3.

We found that our representations (the top row in Figure 3) grouped the word pairs according to their semantic relation in some specific domains based only on unsupervised learning. This property is desirable for the lexical semantic relation detection task. In contrast to our representations,

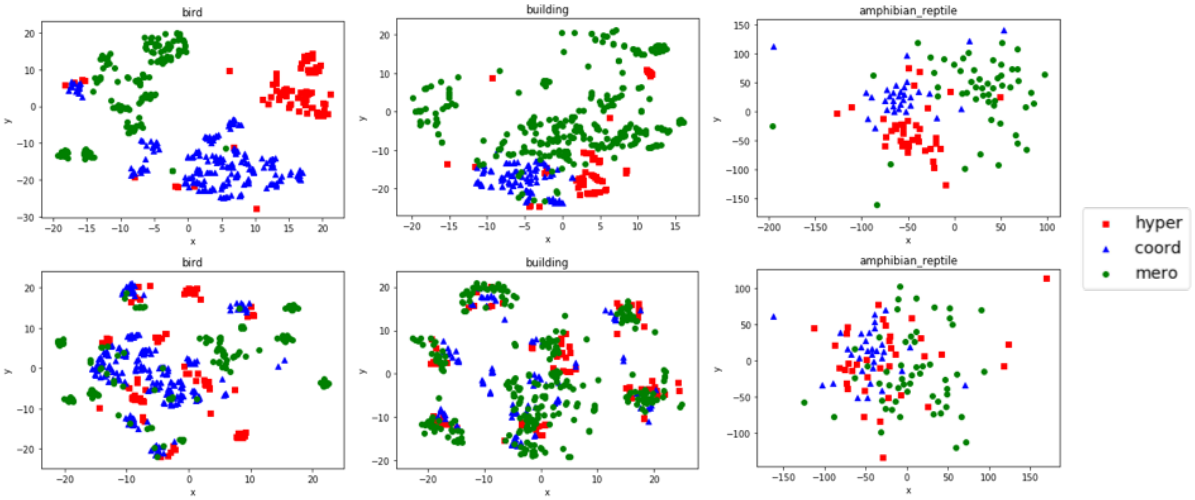


Figure 3: Visualization of the our word pair representations $\boldsymbol{v}_{p-paths(w_1,w_2)}$ (top row) and the concatenation of two word embeddings (bottom row) using t-SNE in some domains. The two axes of each plot, x and y, are the reduced dimensions using t-SNE.

the concatenation of word embeddings (the bottom row in Figure 3) has little or no such tendency in all domains. The data points of the concatenation of word embeddings are scattered or jumbled. This is because the concatenation of word embeddings cannot capture the relational information of word pairs but only the distributional information of each word (Levy et al., 2015).

This visualization further shows that our word pair representations can be used as pseudo-path representations to alleviate the missing path problem.

7 Conclusion

In this paper, we proposed the novel methods with modeling $P(path|w_1, w_2)$ to solve the missing path problem. Our neural model of $P(path|w_1, w_2)$ can be learned from a corpus in an unsupervised manner, and can generalize co-occurrences of word pairs and dependency paths. We demonstrated that this model can be applied in the two ways: (1) to augment path data by predicting plausible paths for a given word pair, and (2) to extract from word pairs useful features capturing co-occurring path information. Finally, our experiments demonstrated that our methods can improve upon the previous models and successfully solve the missing path problem.

In future work, we will explore unsupervised learning with a neural path encoder. Our model bears not only word pair representations but also dependency path representations as context vectors. Thus, we intend to apply these representations to various tasks, which path representations contribute to.

Acknowledgments

This work was supported by JSPS KAKENHI Grant numbers JP17H01831, JP15K12873.

References

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 23–32. http://www.aclweb.org/anthology/E12-1004.

Marco Baroni and Alessandro Lenci. 2011. How we blessed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on Geometrical Models of Natural Language Semantics*. Association for Computational Linguistics, pages 1–10. http://www.aclweb.org/anthology/W11-2501.

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

Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. 2010. Recognizing textual entailment: Rational, evaluation and approaches erratum. *Natural Language Engineering* 16(1):105–105. https://doi.org/10.1017/S1351324909990234.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, Mass.

Patrick Hanks. 2009. The impact of corpora on dictionaries. In Paul Baker, editor, *Contemporary Corpus Linguistics*, Continuum, London, Great Britain, pages 214–236.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *COLING 1992 Volume 2: The 15th International Conference on Computational Linguistics*. http://www.aclweb.org/anthology/C92-2082.

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

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1681–1691. https://doi.org/10.3115/v1/P15-1162.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association of Computational Linguistics* 4:313–327. http://www.aclweb.org/anthology/Q16-1023.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 302–308. https://doi.org/10.3115/v1/P14-2050.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 970–976. https://doi.org/10.3115/v1/N15-1098.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research* 9(Nov):2579–2605.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. Curran Associates Inc., USA, NIPS'13, pages 3111–3119. http://dl.acm.org/citation.cfm?id=2999792.2999959.

Silvia Necsulescu, Sara Mendes, David Jurgens, Núria Bel, and Roberto Navigli. 2015. Reading between the lines: Overcoming data sparsity for accurate classification of lexical relationships. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*. Association for Computational Linguistics, pages 182–192. https://doi.org/10.18653/v1/S15-1021.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research* 12(Oct):2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 1532–1543. https://doi.org/10.3115/v1/D14-1162.

Stephen Roller and Katrin Erk. 2016. Relations such as hypernymy: Identifying and exploiting hearst patterns in distributional vectors for lexical entailment. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 2163–2172. https://doi.org/10.18653/v1/D16-1234.

Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin City University and Association for Computational Linguistics, pages 1025–1036. http://www.aclweb.org/anthology/C14-1097.

Enrico Santus, Alessandro Lenci, Tin-Shing Chiu, Qin Lu, and Chu-Ren Huang. 2016. Nine features in a random forest to learn taxonomical semantic relations. In *LREC*. Portorož, Slovenia.

Enrico Santus, Frances Yung, Alessandro Lenci, and Chu-Ren Huang. 2015. Evalution 1.0: an evolving semantic dataset for training and evaluation of distributional semantic models. In *Proceedings of The 4th Workshop on Linked Data in Linguistics (LDL-2015)*. Association for Computational Linguistics, pages 64–69. https://doi.org/10.18653/v1/W15-4208.

Rico Sennrich and Barry Haddow. 2016. Linguistic Input Features Improve Neural Machine Translation. In *Proceedings of the First Conference on Machine Translation*. Association for Computational Linguistics, Berlin, Germany, pages 83–91. http://www.aclweb.org/anthology/W16-2209.pdf.

Vered Shwartz and Ido Dagan. 2016. Path-based vs. distributional information in recognizing lexical semantic relations. In *Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon (CogALex-V), in COLING*. Osaka, Japan. http://www.aclweb.org/anthology/W16-5304.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 2389–2398. https://doi.org/10.18653/v1/P16-1226.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. In *Advances in Neural Information Processing Systems 17*, MIT Press, Cambridge, MA, pages 1297–1304. http://books.nips.cc/papers/files/nips17/NIPS2004_0887.pdf.

Ekaterina Vylomova, Laura Rimell, Trevor Cohn, and Timothy Baldwin. 2016. Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1671–1682. https://doi.org/10.18653/v1/P16-1158.

Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller. 2014. Learning to distinguish hypernyms and co-hyponyms. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin City University and Association for Computational Linguistics, pages 2249–2259. http://www.aclweb.org/anthology/C14-1212.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1785–1794. http://aclweb.org/anthology/D15-1206.

Shuo Yang, Lei Zou, Zhongyuan Wang, Jun Yan, and Ji-Rong Wen. 2017. Efficiently answering technical questions-a knowledge graph approach. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*. pages 3111–3118.

Specialising Word Vectors for Lexical Entailment

Ivan Vulić[1] and **Nikola Mrkšić**[2]
[1] Language Technology Lab, University of Cambridge
[2] PolyAI
iv250@cam.ac.uk nikola@poly-ai.com

Abstract

We present LEAR (**L**exical **E**ntailment **A**ttract-**R**epel), a novel post-processing method that transforms any input word vector space to emphasise the asymmetric relation of *lexical entailment* (LE), also known as the IS-A or hyponymy-hypernymy relation. By injecting external linguistic constraints (e.g., WordNet links) into the initial vector space, the LE specialisation procedure brings true hyponymy-hypernymy pairs closer together in the transformed Euclidean space. The proposed asymmetric distance measure adjusts the norms of word vectors to reflect the actual WordNet-style hierarchy of concepts. Simultaneously, a joint objective enforces semantic similarity using the symmetric cosine distance, yielding a vector space specialised for both lexical relations at once. LEAR specialisation achieves state-of-the-art performance in the tasks of hypernymy directionality, hypernymy detection, and graded lexical entailment, demonstrating the effectiveness and robustness of the proposed asymmetric specialisation model.

1 Introduction

Word representation learning has become a research area of central importance in NLP, with its usefulness demonstrated across application areas such as parsing (Chen and Manning, 2014), machine translation (Zou et al., 2013), and many others (Turian et al., 2010; Collobert et al., 2011). Standard techniques for inducing word embeddings rely on the *distributional hypothesis* (Harris, 1954), using co-occurrence information from large textual corpora to learn meaningful word representations (Mikolov et al., 2013; Levy and Goldberg, 2014; Pennington et al., 2014; Bojanowski et al., 2017).

A major drawback of the distributional hypothesis is that it coalesces different relationships between words, such as synonymy and topical relatedness, into a single vector space. A popular solution is to go beyond stand-alone unsupervised learning and fine-tune distributional vector spaces by using external knowledge from human- or automatically-constructed knowledge bases. This is often done as a *post-processing* step, where distributional vectors are gradually refined to satisfy linguistic constraints extracted from lexical resources such as WordNet (Faruqui et al., 2015; Mrkšić et al., 2016), the Paraphrase Database (PPDB) (Wieting et al., 2015), or BabelNet (Mrkšić et al., 2017; Vulić et al., 2017a). One advantage of post-processing methods is that they treat the input vector space as a *black box*, making them applicable to any input space.

A key property of these methods is their ability to transform the vector space by *specialising* it for a particular relationship between words.[1] Prior work has predominantly focused on distinguishing between semantic similarity and conceptual relatedness (Faruqui et al., 2015; Mrkšić et al., 2017; Vulić et al., 2017b). In this paper, we introduce a novel post-processing model which specialises vector spaces for the *lexical entailment* (LE) relation.

Word-level lexical entailment is an *asymmetric* semantic relation (Collins and Quillian, 1972; Beckwith et al., 1991). It is a key principle determining the organisation of semantic networks into hierarchical structures such as semantic ontologies (Fellbaum, 1998). Automatic reasoning about LE supports tasks such as taxonomy creation (Snow et al., 2006; Navigli et al., 2011), natural language inference (Dagan et al., 2013; Bowman et al., 2015), text generation (Biran and McKeown, 2013), and metaphor detection (Mohler et al., 2013).

Our novel LE specialisation model, termed LEAR (**L**exical **E**ntailment **A**ttract-**R**epel), is inspired by ATTRACT-REPEL, a state-of-the-art general spe-

[1] Distinguishing between synonymy and antonymy has a positive impact on real-world language understanding tasks such as Dialogue State Tracking (Mrkšić et al., 2017).

Proceedings of NAACL-HLT 2018, pages 1134–1145
New Orleans, Louisiana, June 1 - 6, 2018.

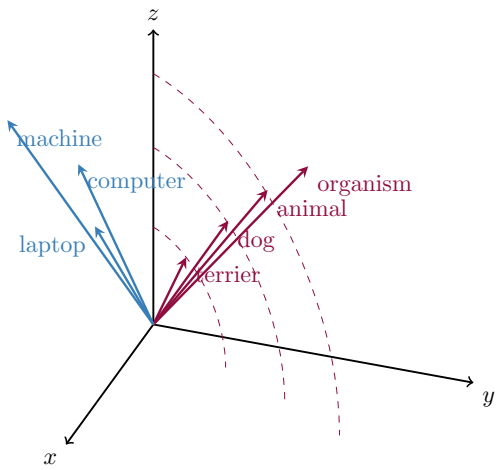


Figure 1: An illustration of LEAR specialisation. LEAR controls the arrangement of vectors in the transformed vector space by: **1)** emphasising symmetric similarity of LE pairs through cosine distance (by enforcing small angles between $\overrightarrow{terrier}$ and $\overrightarrow{dog}$ or $\overrightarrow{dog}$ and $\overrightarrow{animal}$); and **2)** by imposing an LE ordering using vector norms, adjusting them so that higher-level concepts have larger norms (e.g., $|\overrightarrow{animal}| > |\overrightarrow{dog}| > |\overrightarrow{terrier}|$).

cialisation framework (Mrkšić et al., 2017).[2] The key idea of LEAR, illustrated by Figure 1, is to pull desirable (ATTRACT) examples described by the constraints closer together, while at the same time pushing undesirable (REPEL) word pairs away from each other. Concurrently, LEAR (re-)arranges vector norms so that norm values in the Euclidean space reflect the hierarchical organisation of concepts according to the given LE constraints: put simply, higher-level concepts are assigned larger norms. Therefore, LEAR simultaneously captures the hierarchy of concepts (through vector norms) and their similarity (through their cosine distance). The two pivotal pieces of information are combined into an *asymmetric distance measure* which quantifies the LE strength in the specialised space.

After specialising four well-known input vector spaces with LEAR, we test them in three standard word-level LE tasks (Kiela et al., 2015b): **1)** hypernymy *directionality*; **2)** hypernymy *detection*; and **3)** *combined* hypernymy detection/directionality. Our specialised vectors yield notable improvements over the strongest baselines for each task, with each input space, demonstrating the effectiveness and robustness of LEAR specialisation.

[2] https://github.com/nmrksic/attract-repel

The employed asymmetric distance allows one to make graded assertions about hierarchical relationships between concepts in the specialised space. This property is evaluated using HyperLex, a recent *graded LE* dataset (Vulić et al., 2017). The LEAR-specialised vectors push state-of-the-art Spearman's correlation from 0.540 to 0.686 on the full dataset (2,616 word pairs), and from 0.512 to 0.705 on its noun subset (2,163 word pairs).

The code for the LEAR model is available from: `github.com/nmrksic/lear`.

2 Methodology

2.1 The ATTRACT-REPEL Framework

Let V be the vocabulary, A the set of ATTRACT word pairs (e.g., *intelligent* and *brilliant*), and R the set of REPEL word pairs (e.g., *vacant* and *occupied*). The ATTRACT-REPEL procedure operates over mini-batches of such pairs $\mathcal{B}_A$ and $\mathcal{B}_R$. For ease of notation, let each word pair (x_l, x_r) in these two sets correspond to a vector pair $(\mathbf{x}_l, \mathbf{x}_r)$, so that a mini-batch of k_1 word pairs is given by $\mathcal{B}_A = [(\mathbf{x}_l^1, \mathbf{x}_r^1), \ldots, (\mathbf{x}_l^{k_1}, \mathbf{x}_r^{k_1})]$ (similarly for $\mathcal{B}_R$, which consists of k_2 example pairs).

Next, the sets of pseudo-negative examples $T_A = [(\mathbf{t}_l^1, \mathbf{t}_r^1), \ldots, (\mathbf{t}_l^{k_1}, \mathbf{t}_r^{k_1})]$ and $T_R = [(\mathbf{t}_l^1, \mathbf{t}_r^1), \ldots, (\mathbf{t}_l^{k_2}, \mathbf{t}_r^{k_2})]$ are defined as pairs of *negative examples* for each ATTRACT and REPEL example pair in mini-batches $\mathcal{B}_A$ and $\mathcal{B}_R$. These negative examples are chosen from the word vectors present in $\mathcal{B}_A$ or $\mathcal{B}_R$ so that, for each ATTRACT pair $(\mathbf{x}_l, \mathbf{x}_r)$, the negative example pair $(\mathbf{t}_l, \mathbf{t}_r)$ is chosen so that $\mathbf{t}_l$ is the vector closest (in terms of cosine distance) to $\mathbf{x}_l$ and $\mathbf{t}_r$ is closest to $\mathbf{x}_r$. Similarly, for each REPEL pair $(\mathbf{x}_l, \mathbf{x}_r)$, the negative example pair $(\mathbf{t}_l, \mathbf{t}_r)$ is chosen from the remaining in-batch vectors so that $\mathbf{t}_l$ is the vector furthest away from $\mathbf{x}_l$ and $\mathbf{t}_r$ is furthest from $\mathbf{x}_r$.

The negative examples are used to: **a)** force ATTRACT pairs to be closer to each other than to their respective negative examples; and **b)** to force REPEL pairs to be further away from each other than from their negative examples. The first term of the cost function pulls ATTRACT pairs together:

$$Att(\mathcal{B}_A, T_A) = \sum_{i=1}^{k_1} \big[\, \tau\left(\delta_{att} + cos(\mathbf{x}_l^i, \mathbf{t}_l^i) - cos(\mathbf{x}_l^i, \mathbf{x}_r^i)\right) + \tau\left(\delta_{att} + cos(\mathbf{x}_r^i, \mathbf{t}_r^i) - cos(\mathbf{x}_l^i, \mathbf{x}_r^i)\right) \big] \quad (1)$$

where cos denotes cosine similarity, $\tau(x) = \max(0, x)$ is the hinge loss function and δ_{att} is the attract margin which determines how much closer these vectors should be to each other than to their respective negative examples. The second part of the cost function pushes REPEL word pairs away from each other:

$$Rep(\mathcal{B}_R, T_R) = \sum_{i=1}^{k_2} \big[\tau\left(\delta_{rep} + cos(\mathbf{x}_l^i, \mathbf{x}_r^i) - cos(\mathbf{x}_l^i, \mathbf{t}_l^i)\right) + \tau\left(\delta_{rep} + cos(\mathbf{x}_l^i, \mathbf{x}_r^i) - cos(\mathbf{x}_r^i, \mathbf{t}_r^i)\right) \big] \quad (2)$$

In addition to these two terms, an additional regularisation term is used to *preserve* the abundance of high-quality semantic content present in the distributional vector space, as long as this information does not contradict the injected linguistic constraints. If $V(\mathcal{B})$ is the set of all word vectors present in the given mini-batch, then:

$$Reg(\mathcal{B}_A, \mathcal{B}_R) = \sum_{\mathbf{x}_i \in V(\mathcal{B}_A \cup \mathcal{B}_R)} \lambda_{reg} \left\| \widehat{\mathbf{x}}_i - \mathbf{x}_i \right\|_2$$

where λ_{reg} is the L2 regularization constant and $\widehat{\mathbf{x}}_i$ denotes the original (distributional) word vector for word x_i. The full ATTRACT-REPEL cost function is given by the sum of all three terms.

2.2 LEAR: Encoding Lexical Entailment

In this section, the ATTRACT-REPEL framework is extended to model lexical entailment jointly with (symmetric) semantic similarity. To do this, the method uses an additional source of external lexical knowledge: let L be the set of *directed* lexical entailment constraints such as *(corgi, dog)*, *(dog, animal)*, or *(corgi, animal)*, with lower-level concepts on the left and higher-level ones on the right (the source of these constraints will be discussed in Section 3). The optimisation proceeds in the same way as before, considering a mini-batch of LE pairs $\mathcal{B}_L$ consisting of k_3 word pairs standing in the (directed) lexical entailment relation.

Unlike symmetric similarity, lexical entailment is an asymmetric relation which encodes a hierarchical ordering between concepts. Inferring the direction of the entailment relation between word vectors requires the use of an asymmetric distance function. We define three different ones, all of which use the word vector's norms to impose an ordering between high- and low-level concepts:

$$D_1(\mathbf{x}, \mathbf{y}) = |\mathbf{x}| - |\mathbf{y}| \quad (3)$$

$$D_2(\mathbf{x}, \mathbf{y}) = \frac{|\mathbf{x}| - |\mathbf{y}|}{|\mathbf{x}| + |\mathbf{y}|} \quad (4)$$

$$D_3(\mathbf{x}, \mathbf{y}) = \frac{|\mathbf{x}| - |\mathbf{y}|}{\max(|\mathbf{x}|, |\mathbf{y}|)} \quad (5)$$

The lexical entailment term (for the j-th asymmetric distance, $j \in 1, 2, 3$) is defined as:

$$LE_j(\mathcal{B}_L) = \sum_{i=1}^{k_3} D_j(\mathbf{x}_i, \mathbf{y}_i) \quad (6)$$

The first distance serves as the baseline: it uses the word vectors' norms to order the concepts, that is to decide which of the words is likely to be the higher-level concept. In this case, the magnitude of the difference between the two norms determines the 'intensity' of the LE relation. This is potentially problematic, as this distance does not impose a limit on the vectors' norms. The second and third metric take a more sophisticated approach, using the ratios of the differences between the two norms and either: **a)** the sum of the two norms; or **b)** the larger of the two norms. In doing that, these metrics ensure that the cost function only considers the norms' ratios. This means that the cost function no longer has the incentive to increase word vectors' norms past a certain point, as the magnitudes of norm ratios grow in size much faster than the linear relation defined by the first distance function.

To model the semantic and the LE relations jointly, the LEAR cost function jointly optimises the four terms of the expanded cost function:

$$C(\mathcal{B}_A, T_A, \mathcal{B}_R, T_R, \mathcal{B}_L, T_L) = Att(\mathcal{B}_S, T_S) + \ldots + Rep(\mathcal{B}_A, T_A) + Reg(\mathcal{B}_A, \mathcal{B}_R, \mathcal{B}_L) + \ldots + Att(\mathcal{B}_L, T_L) + LE_j(B_L)$$

LE Pairs as ATTRACT Constraints The combined cost function makes use of the batch of lexical constraints $\mathcal{B}_L$ twice: once in the defined asymmetric cost function LE_j, and once in the symmetric ATTRACT term $Att(\mathcal{B}_L, T_L)$. This means that words standing in the lexical entailment relation are forced to be similar both in terms of cosine distance (via the symmetric ATTRACT term) and in terms of the asymmetric LE distance from Eq. (6).

Decoding Lexical Entailment The defined cost function serves to encode semantic similarity and

LE relations in the same vector space. Whereas the similarity can be inferred from the standard cosine distance, the LEAR optimisation embeds lexical entailment as a combination of the symmetric ATTRACT term and the newly defined asymmetric LE_j cost function. Consequently, the metric used to determine whether two words stand in the LE relation must combine the two cost terms as well. We define the LE *decoding* metric as:

$$I_{LE}(\mathbf{x}, \mathbf{y}) = dcos(\mathbf{x}, \mathbf{y}) + D_j(\mathbf{x}, \mathbf{y}) \quad (7)$$

where $dcos(\mathbf{x}, \mathbf{y})$ denotes the cosine distance. This decoding function combines the symmetric and the asymmetric cost term, in line with the combination of the two used to perform LEAR specialisation. In the evaluation, we show that combining the two cost terms has a synergistic effect, with both terms contributing to stronger performance across all LE tasks used for evaluation.

3 Experimental Setup

Starting Distributional Vectors To test the robustness of LEAR specialisation, we experiment with a variety of well-known, publicly available English word vectors: **1)** Skip-Gram with Negative Sampling (SGNS) (Mikolov et al., 2013) trained on the Polyglot Wikipedia (Al-Rfou et al., 2013) by Levy and Goldberg (2014); **2)** GLOVE Common Crawl (Pennington et al., 2014); **3)** CONTEXT2VEC (Melamud et al., 2016), which replaces CBOW contexts with contexts based on bidirectional LSTMs (Hochreiter and Schmidhuber, 1997); and **4)** FASTTEXT (Bojanowski et al., 2017), a SGNS variant which builds word vectors as the sum of their constituent character n-gram vectors.[3]

Linguistic Constraints We use three groups of linguistic constraints in the LEAR specialisation model, covering three different relation types which are all beneficial to the specialisation process: directed **1)** *lexical entailment* (LE) *pairs*; **2)** *synonymy pairs*; and **3)** *antonymy pairs*. Synonyms are included as symmetric ATTRACT pairs (i.e., the $\mathcal{B}_A$ pairs) since they can be seen as defining a trivial symmetric IS-A relation (Rei and Briscoe, 2014; Vulić et al., 2017). For a similar reason, antonyms are clear REPEL constraints as they anti-correlate with the LE relation.[4] Synonymy and antonymy constraints are taken from prior work (Zhang et al., 2014; Ono et al., 2015): they are extracted from WordNet (Fellbaum, 1998) and Roget (Kipfer, 2009). In total, we work with 1,023,082 synonymy pairs (11.7 synonyms per word on average) and 380,873 antonymy pairs (6.5 per word).[5]

As in prior work (Nguyen et al., 2017; Nickel and Kiela, 2017), LE constraints are extracted from the WordNet hierarchy, relying on the transitivity of the LE relation. This means that we include both direct and indirect LE pairs in our set of constraints (e.g., (*pangasius, fish*), (*fish, animal*), and *(pangasius, animal)*). We retained only noun-noun and verb-verb pairs, while the rest were discarded: the final number of LE constraints is 1,545,630.[6]

Training Setup We adopt the original ATTRACT-REPEL model setup without any fine-tuning. Hyperparameter values are set to: $\delta_{att} = 0.6$, $\delta_{rep} = 0.0$, $\lambda_{reg} = 10^{-9}$ (Mrkšić et al., 2017). The models are trained for 5 epochs with the AdaGrad algorithm (Duchi et al., 2011), with batch sizes set to $k_1 = k_2 = k_3 = 128$ for faster convergence.

4 Results and Discussion

We test and analyse LEAR-specialised vector spaces in two standard word-level LE tasks used in prior work: hypernymy directionality and detection (Section 4.1) and graded LE (Section 4.2).

4.1 LE Directionality and Detection

The first evaluation uses three classification-style tasks with increased levels of difficulty. The tasks are evaluated on three datasets used extensively in the LE literature (Roller et al., 2014; Santus et al., 2014; Weeds et al., 2014; Shwartz et al., 2017; Nguyen et al., 2017), compiled into an integrated evaluation set by Kiela et al. (2015b).[7]

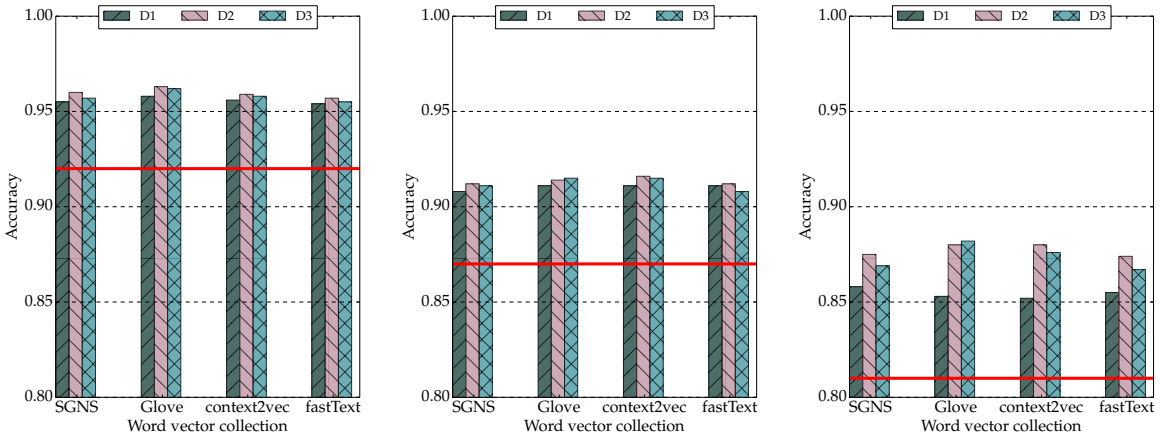
[3] All vectors are 300-dimensional except for the 600-dimensional CONTEXT2VEC vectors; for further details regarding the architectures and training setup of the used vector collections, we refer the reader to the original papers. We also experimented with dependency-based SGNS vectors (Levy and Goldberg, 2014), observing similar patterns in the results.

[4] In short, the question "*Is X a type of X?*" (synonymy) is trivially true, while the question "*Is ¬X a type of X?*" (antonymy) is trivially false.

[5] https://github.com/tticoin/AntonymDetection

[6] We also experimented with additional 30,491 LE constraints from the Paraphrase Database (PPDB) 2.0 (Pavlick et al., 2015). Adding them to the WordNet-based LE pairs makes no significant impact on the final performance. We also used synonymy and antonymy pairs from other sources, such as word pairs from PPDB used previously by Wieting et al. (2015), and BabelNet (Navigli and Ponzetto, 2012) used by Mrkšić et al. (2017), reaching the same conclusions.

[7] http://www.cl.cam.ac.uk/∼dk427/generality.html



(a) BLESS: Directionality **(b) WBLESS**: Detection **(c) BIBLESS**: Detect+direct

Figure 2: Summary of the results on three different word-level LE subtasks: (a) *directionality*; (b) *detection*; (c) *detection and directionality*. Vertical bars denote the results obtained by different input word vector spaces which are post-processed/specialised by our LEAR specialisation model using three variants of the asymmetric distance (D_1, D_2, D_3), see Section 2. Thick horizontal red lines refer to the best reported scores on each subtask for these datasets; the baseline scores are taken from Nguyen et al. (2017).

The first task, LE directionality, is conducted on 1,337 LE pairs originating from the BLESS evaluation set (Baroni and Lenci, 2011). Given a true LE pair, the task is to predict the correct hypernym. With LEAR-specialised vectors this is achieved by simply comparing the vector norms of each concept in a pair: the one with the larger norm is the hypernym (see Figure 1).

The second task, LE detection, involves a binary classification on the WBLESS dataset (Weeds et al., 2014) which comprises 1,668 word pairs standing in a variety of relations (LE, meronymy-holonymy, co-hyponymy, reversed LE, no relation). The model has to detect a true LE pair, that is, to distinguish between the pairs where the statement *X is a (type of) Y* is true from all other pairs. With LEAR vectors, this classification is based on the asymmetric distance score: if the score is above a certain threshold, we classify the pair as "true LE", otherwise as "other". While Kiela et al. (2015b) manually define the threshold value, we follow the approach of Nguyen et al. (2017) and cross-validate: in each of the 1,000 iterations, 2% of the pairs are sampled for threshold tuning, and the remaining 98% are used for testing. The reported numbers are therefore average accuracy scores.[8]

[8]We have conducted more LE directionality and detection experiments on other datasets such as EVALution (Santus et al., 2015), the $N_1 \vDash N_2$ dataset of Baroni et al. (2012), and the dataset of Lenci and Benotto (2012) with similar performances and findings. We do not report all these results for brevity and clarity of presentation.

The final task, LE detection *and* directionality, concerns a three-way classification on BIBLESS, a relabeled version of WBLESS. The task is now to distinguish both LE pairs ($\rightarrow 1$) and reversed LE pairs ($\rightarrow -1$) from other relations ($\rightarrow 0$), and then additionally select the correct hypernym in each detected LE pair. We apply the same test protocol as in the LE detection task.

Results and Analysis The original paper of Kiela et al. (2015b) reports the following best scores on each task: 0.88 (BLESS), 0.75 (WBLESS), 0.57 (BIBLESS). These scores were recently surpassed by Nguyen et al. (2017), who, instead of post-processing, combine WordNet-based constraints with an SGNS-style objective into a joint model. They report the best scores to date: 0.92 (BLESS), 0.87 (WBLESS), and 0.81 (BIBLESS).

The performance of the four LEAR-specialised word vector collections is shown in Figure 2 (together with the strongest baseline scores for each of the three tasks). The comparative analysis confirms the increased complexity of subsequent tasks. LEAR specialisation of *each* of the starting vector spaces consistently outperformed *all* baseline scores across *all* three tasks. The extent of the improvements is correlated with task difficulty: it is lowest for the easiest directionality task ($0.92 \rightarrow 0.96$), and highest for the most difficult detection plus directionality task ($0.81 \rightarrow 0.88$).

The results show that the two LEAR variants which do not rely on absolute norm values and

	Norm		Norm		Norm
terrier	0.87	laptop	0.60	cabriolet	0.74
dog	2.64	computer	2.96	car	3.59
mammal	8.57	machine	6.15	vehicle	7.78
vertebrate	10.96	device	12.09	transport	8.01
animal	11.91	artifact	17.71	instrumentality	14.56
organism	20.08	object	23.55	–	–

Table 1: L2 norms for selected concepts from the WordNet hierarchy. Input: FASTTEXT; LEAR: D2.

perform a normalisation step in the asymmetric distance (D2 and D3) have an edge over the D1 variant which operates with unbounded norms. The difference in performance between D2/D3 and D1 is even more pronounced in the graded LE task (see Section 4.2). This shows that the use of unbounded vector norms diminishes the importance of the symmetric cosine distance in the combined asymmetric distance. Conversely, the synergistic combination used in D2/D3 does not suffer from this issue.

The high scores achieved with each of the four word vector collections show that LEAR is not dependent on any particular word representation architecture. Moreover, the extent of the performance improvements in each task suggests that LEAR is able to reconstruct the concept hierarchy coded in the input linguistic constraints.

Moreover, we have conducted a small experiment to verify that the LEAR method can generalise beyond what is directly coded in pairwise external constraints. A simple WordNet lookup baseline yields accuracy scores of 0.82 and 0.80 on the directionality and detection tasks, respectively. This baseline is outperformed by LEAR: its scores are 0.96 and 0.92 on the two tasks when relying on the same set of WordNet constraints.

Importance of Vector Norms To verify that the knowledge concerning the position in the semantic hierarchy actually arises from vector norms, we also manually inspect the norms after LEAR specialisation. A few examples are provided in Table 1. They indicate a desirable pattern in the norm values which imposes a hierarchical ordering on the concepts. Note that the original distributional SGNS model (Mikolov et al., 2013) does not normalise vectors to unit length after training. However, these norms are not at all correlated with the desired hierarchical ordering, and are therefore useless for LE-related applications: the non-specialised distributional SGNS model scores 0.44, 0.48, and 0.34 on the three tasks, respectively.

4.2 Graded Lexical Entailment

Asymmetric distances in the LEAR-specialised space quantify the degree of lexical entailment between any two concepts. This means that they can be used to make fine-grained assertions regarding the hierarchical relationships between concepts. We test this property on HyperLex (Vulić et al., 2017), a gold standard dataset for evaluating how well word representation models capture graded LE, grounded in the notions of *concept (proto)typicality* (Rosch, 1973; Medin et al., 1984) and *category vagueness* (Kamp and Partee, 1995; Hampton, 2007) from cognitive science. HyperLex contains 2,616 word pairs (2,163 noun pairs and 453 verb pairs) scored by human raters in the $[0, 6]$ interval following the question *"To what degree is X a (type of) Y?"*[9]

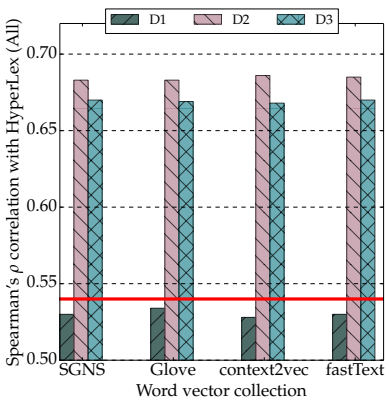
As shown by the high inter-annotator agreement on HyperLex (0.85), humans are able to consistently reason about graded LE.[10] However, current state-of-the-art representation architectures are far from this ceiling. For instance, Vulić et al. (2017) evaluate a plethora of architectures and report a high-score of only 0.320 (see the summary table in Figure 3). Two recent representation models (Nickel and Kiela, 2017; Nguyen et al., 2017) focused on the LE relation in particular (and employing the same set of WordNet-based constraints as LEAR) report the highest score of 0.540 (on the entire dataset) and 0.512 (on the noun subset).

Results and Analysis We scored all HyperLex pairs using the combined asymmetric distance described by Equation (7), and then computed Spearman's rank correlation with the ground-truth ranking. Our results, together with the strongest baseline scores, are summarised in Figure 3.

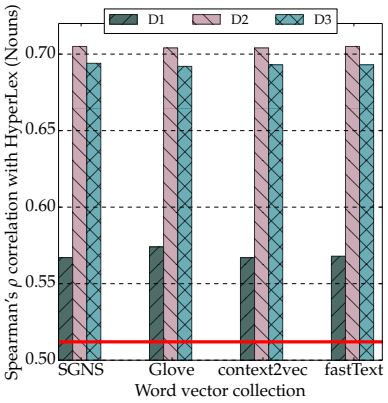
The summary table in Figure 3(c) shows the HyperLex performance of several prominent LE models. We provide only a quick outline of these models here; further details can be found in the original papers. FREQ-RATIO exploits the fact that more general concepts tend to occur more frequently in textual corpora. SGNS (COS) uses non-specialised

[9]From another perspective, one might say that graded LE provides finer-grained human judgements on a continuous scale rather than simplifying the judgements into binary discrete decisions. For instance, the HyperLex score for the pair *(girl, person)* is 5.91/6, the score for *(guest, person)* is 4.33, while the score for the reversed pair *(person, guest)* is 1.73.

[10]For further details concerning HyperLex, we refer the reader to the resource paper (Vulić et al., 2017). The dataset is available at: http://people.ds.cam.ac.uk/iv250/hyperlex.html



(a) HyperLex: **All**



(b) HyperLex: **Nouns**

	All
FREQ-RATIO	0.279
SGNS (COS)	0.205
SLQS-SIM	0.228
VISUAL	0.209
WN-BEST	0.234
WORD2GAUSS	0.206
SIM-SPEC	0.320
ORDER-EMB	0.191
POINCARÉ (nouns)	0.512
HYPERVEC	0.540
Best LEAR	**0.686**

(c) Summary

Figure 3: Results on the graded LE task defined by HyperLex. Following Nickel and Kiela (2017), we use Spearman's rank correlation scores on: **a)** the entire dataset (2,616 noun and verb pairs); and **b)** its noun subset (2,163 pairs). The summary table shows the performance of other well-known architectures on the full HyperLex dataset, compared to the best results achieved using LEAR specialisation.

SGNS vectors and quantifies the LE strength using the symmetric cosine distance between vectors. A comparison of these models to the best-performing LEAR vectors shows the extent of the improvements achieved using the specialisation approach.

LEAR-specialised vectors also outperform SLQS-SIM (Santus et al., 2014) and VISUAL (Kiela et al., 2015b), two LE detection models similar in spirit to LEAR. These models combine symmetric semantic similarity (through cosine distance) with an asymmetric measure of lexical generality obtained either from text (SLQS-SIM) or visual data (VISUAL). The results on HyperLex indicate that the two generality-based measures are too coarse-grained for graded LE judgements. These models were originally constructed to tackle LE directionality and detection tasks (see Section 4.1), but their performance is surpassed by LEAR on those tasks as well. The VISUAL model outperforms SLQS-SIM. However, its numbers on BLESS (0.88), WBLESS (0.75), and BIBLESS (0.57) are far from the top-performing LEAR vectors (0.96, 0.92, 0.88).[11]

WN-BEST denotes the best result with asymmetric similarity measures which use the WordNet structure as their starting point (Wu and Palmer, 1994; Pedersen et al., 2004). This model can be observed as a model that directly looks up the full WordNet structure to reason about graded lexical entailment. The reported results from Figure 3(c) suggest it is more effective to quantify the LE relation strength by using WordNet as the source of constraints for specialisation models such as HYPERVEC or LEAR.

WORD2GAUSS (Vilnis and McCallum, 2015) represents words as multivariate K-dimensional Gaussians rather than points in the embedding space: it is therefore naturally asymmetric and was used in LE tasks before, but its performance on HyperLex indicates that it cannot effectively capture the subtleties required to model graded LE. However, note that the comparison is not strictly fair as WORD2GAUSS does not leverage any external knowledge. An interesting line for future research is to embed external knowledge within this representation framework.

Most importantly, LEAR outperforms three recent (and conceptually different) architectures: ORDER-EMB (Vendrov et al., 2016), POINCARÉ (Nickel and Kiela, 2017), and HYPERVEC (Nguyen et al., 2017). Like LEAR, all of these models complement distributional knowledge with external linguistic constraints extracted from WordNet. Each model uses a different strategy to exploit the hierarchical relationships encoded in these constraints (their approaches are discussed in Section 5).[12] However, LEAR, as the first LE-oriented post-processor, is able to utilise the constraints more effectively than its competitors. Another advantage of LEAR is its applicability to any input

[11] We note that SLQS and VISUAL do not leverage any external knowledge from WordNet, but the VISUAL model leverages external visual information about concepts.

[12] As discussed previously by Vulić et al. (2017), the off-the-shelf ORDER-EMB vectors were trained for the binary ungraded LE detection task: this limits their expressiveness in the graded LE task.

	WBLESS	BIBLESS	HL-A	HL-N
LEAR variant				
SYM-ONLY	0.687	0.679	0.469	0.429
ASYM-ONLY	0.867	0.824	0.529	0.565
FULL	**0.912**	**0.875**	**0.686**	**0.705**

Table 2: Analysing the importance of the synergy in the FULL LEAR model on the final performance on WBLESS, BLESS, HyperLex-All (HL-A) and HyperLex-Nouns (HL-N). Input: FASTTEXT. D2.

vector space.

Figures 3(a) and 3(b) indicate that the two LEAR variants which rely on norm ratios (D2 and D3), rather than on absolute (unbounded) norm differences (D1), achieve stronger performance on HyperLex. The highest correlation scores are again achieved by D2 with all input vector spaces.

4.3 Further Discussion

Why Symmetric + Asymmetric? In another experiment, we analyse the contributions of both LE-related terms in the LEAR combined objective function (see Section 2.2). We compare three variants of LEAR: **1)** a symmetric variant which does not arrange vector norms using the $LE_j(\mathcal{B}_L)$ term (SYM-ONLY); **2)** a variant which arranges norms, but does not use LE constraints as additional symmetric ATTRACT constraints (ASYM-ONLY); and **3)** the full LEAR model, which uses both cost terms (FULL). The results with one input space (similar results are achieved with others) are shown in Table 2. This table shows that, while the stand-alone ASYM-ONLY term seems more beneficial than the SYM-ONLY one, using the two terms jointly yields the strongest performance across all LE tasks.

LE and Semantic Similarity We also test whether the asymmetric LE term harms the (norm-independent) cosine distances used to represent semantic similarity. The LEAR model is compared to the original ATTRACT-REPEL model making use of the same set of linguistic constraints. Two true semantic similarity datasets are used for evaluation: SimLex-999 (Hill et al., 2015) and SimVerb-3500 (Gerz et al., 2016). There is no significant difference in performance between the two models, both of which yield similar results on SimLex (Spearman's rank correlation of ≈ 0.71) and SimVerb (≈ 0.70). This proves that cosine distances remain preserved during the optimization of the asymmetric objective performed by the joint LEAR model.

5 Related Work

Vector Space Specialisation A standard approach to incorporating external information into vector spaces is to pull the representations of similar words closer together. Some models integrate such constraints into the training procedure: they modify the prior or the regularisation (Yu and Dredze, 2014; Xu et al., 2014; Bian et al., 2014; Kiela et al., 2015a), or use a variant of the SGNS-style objective (Liu et al., 2015; Osborne et al., 2016; Nguyen et al., 2017). Another class of models, popularly termed *retrofitting*, fine-tune distributional vector spaces by injecting lexical knowledge from semantic databases such as WordNet or the Paraphrase Database (Faruqui et al., 2015; Jauhar et al., 2015; Wieting et al., 2015; Nguyen et al., 2016; Mrkšić et al., 2016; Mrkšić et al., 2017).

LEAR falls into the latter category. However, while previous post-processing methods have focused almost exclusively on specialising vector spaces to emphasise semantic similarity (i.e., to distinguish between similarity and relatedness by explicitly pulling synonyms closer and pushing antonyms further apart), this paper proposed a principled methodology for specialising vector spaces for asymmetric hierarchical relations (of which *lexical entailment* is an instance). Its starting point is the state-of-the-art similarity specialisation framework of Mrkšić et al. (2017), which we extend to support the inclusion of hierarchical asymmetric relationships between words.

Word Vectors and Lexical Entailment Since the hierarchical LE relation is one of the fundamental building blocks of semantic taxonomies and hierarchical concept categorisations (Beckwith et al., 1991; Fellbaum, 1998), a significant amount of research in semantics has been invested into its automatic detection and classification. Early work relied on asymmetric directional measures (Weeds et al., 2004; Clarke, 2009; Kotlerman et al., 2010; Lenci and Benotto, 2012, i.a.) which were based on the distributional inclusion hypothesis (Geffet and Dagan, 2005) or the distributional informativeness or generality hypothesis (Herbelot and Ganesalingam, 2013; Santus et al., 2014). However, these approaches have recently been superseded by methods based on word embeddings. These methods build dense real-valued vectors for capturing the LE relation, either directly in the LE-focused space (Vilnis and McCallum, 2015; Vendrov et al.,

2016; Henderson and Popa, 2016; Nickel and Kiela, 2017; Nguyen et al., 2017) or by using the vectors as features for supervised LE detection models (Tuan et al., 2016; Shwartz et al., 2016; Nguyen et al., 2017; Glavaš and Ponzetto, 2017).

Several LE models embed useful hierarchical relations from external resources such as WordNet into LE-focused vector spaces, with solutions coming in different flavours. The model of Yu et al. (2015) is a dynamic distance-margin model optimised for the LE detection task using hierarchical WordNet constraints. This model was extended by Tuan et al. (2016) to make use of contextual sentential information. A major drawback of both models is their inability to make directionality judgements. Further, their performance has recently been surpassed by the HYPERVEC model of Nguyen et al. (2017). This model combines WordNet constraints with the SGNS distributional objective into a joint model. As such, the model is tied to the SGNS objective and any change of the distributional modelling paradigm implies a change of the entire HYPERVEC model. This makes their model less versatile than the proposed LEAR framework. Moreover, the results achieved using LEAR specialisation achieve substantially better performance across all LE tasks used for evaluation.

Another model similar in spirit to LEAR is the ORDER-EMB model of Vendrov et al. (2016), which encodes hierarchical structure by imposing a partial order in the embedding space: higher-level concepts get assigned higher per-coordinate values in a d-dimensional vector space. The model minimises the violation of the per-coordinate orderings during training by relying on hierarchical WordNet constraints between word pairs. Finally, the POINCARÉ model of Nickel and Kiela (2017) makes use of hyperbolic spaces to learn general-purpose LE embeddings based on n-dimensional Poincaré balls which encode both hierarchy and semantic similarity, again using the WordNet constraints. A similar model in hyperbolic spaces was proposed by Chamberlain et al. (2017). In this paper, we demonstrate that LE-specialised word embeddings with stronger performance can be induced using a simpler model operating in more intuitively interpretable Euclidean vector spaces.

6 Conclusion and Future Work

This paper proposed LEAR, a vector space specialisation procedure which simultaneously injects symmetric and asymmetric constraints into existing vector spaces, performing joint specialisation for two properties: *lexical entailment* and *semantic similarity*. Since the former is not symmetric, LEAR uses an asymmetric cost function which encodes the hierarchy between concepts by manipulating the norms of word vectors, assigning higher norms to higher-level concepts. Specialising the vector space for both relations has a synergistic effect: LEAR-specialised vectors attain state-of-the-art performance in judging semantic similarity and set new high scores across four different lexical entailment tasks. The code for the LEAR model is available from: `github.com/nmrksic/lear`.

In future work, we plan to apply a similar methodology to other asymmetric relations (e.g., *meronymy*), as well as to investigate fine-grained models which can account for differing path lengths from the WordNet hierarchy. We will also extend the model to reason over words unseen in input lexical resources, similar to the recent post-specialisation model oriented towards specialisation of unseen words for similarity (Vulić et al., 2018). We also plan to test the usefulness of LE-specialised vectors in downstream natural language understanding tasks. Porting the model to other languages and enabling cross-lingual applications such as cross-lingual lexical entailment (Upadhyay et al., 2018) is another future research direction.

Acknowledgments

We thank the three anonymous reviewers for their insightful comments and suggestions. We are also grateful to the TakeLab research group at the University of Zagreb for offering support to computationally intensive experiments in our hour of need. This work is supported by the ERC Consolidator Grant LEXICAL: Lexical Acquisition Across Languages (no 648909).

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual NLP. In *Proceedings of CoNLL*, pages 183–192.

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. In *Proceedings of EACL*, pages 23–32.

Marco Baroni and Alessandro Lenci. 2011. How we

BLESSed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop*, pages 1–10.

Richard Beckwith, Christiane Fellbaum, Derek Gross, and George A. Miller. 1991. WordNet: A lexical database organized on psycholinguistic principles. *Lexical acquisition: Exploiting on-line resources to build a lexicon*, pages 211–231.

Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Knowledge-powered deep learning for word embedding. In *Proceedings of ECML-PKDD*, pages 132–148.

Or Biran and Kathleen McKeown. 2013. Classifying taxonomic relations between pairs of Wikipedia articles. In *Proceedings of IJCNLP*, pages 788–794.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the ACL*, 5:135–146.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of EMNLP*, pages 632–642.

Benjamin Paul Chamberlain, James Clough, and Marc Peter Deisenroth. 2017. Neural embeddings of graphs in hyperbolic space. *CoRR*, abs/1705.10359.

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

Daoud Clarke. 2009. Context-theoretic semantics for natural language: An overview. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics (GEMS)*, pages 112–119.

Allan M. Collins and Ross M. Quillian. 1972. Experiments on semantic memory and language comprehension. *Cognition in Learning and Memory*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. Recognizing textual entailment: Models and applications. *Synthesis Lectures on Human Language Technologies*, 6(4):1–220.

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

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of NAACL-HLT*, pages 1606–1615.

Christiane Fellbaum. 1998. *WordNet*.

Maayan Geffet and Ido Dagan. 2005. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of ACL*, pages 107–114.

Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. 2016. SimVerb-3500: A large-scale evaluation set of verb similarity. In *Proceedings of EMNLP*, pages 2173–2182.

Goran Glavaš and Simone Paolo Ponzetto. 2017. Dual tensor model for detecting asymmetric lexico-semantic relations. In *Proceedings of EMNLP*, pages 1758–1768.

James A. Hampton. 2007. Typicality, graded membership, and vagueness. *Cognitive Science*, 31(3):355–384.

Zellig S. Harris. 1954. Distributional structure. *Word*, 10(23):146–162.

James Henderson and Diana Popa. 2016. A vector space for distributional semantics for entailment. In *Proceedings of ACL*, pages 2052–2062.

Aurélie Herbelot and Mohan Ganesalingam. 2013. Measuring semantic content in distributional vectors. In *Proceedings of ACL*, pages 440–445.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. SimLex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Sujay Kumar Jauhar, Chris Dyer, and Eduard H. Hovy. 2015. Ontologically grounded multi-sense representation learning for semantic vector space models. In *Proceedings of NAACL*, pages 683–693.

Hans Kamp and Barbara Partee. 1995. Prototype theory and compositionality. *Cognition*, 57(2):129–191.

Douwe Kiela, Felix Hill, and Stephen Clark. 2015a. Specializing word embeddings for similarity or relatedness. In *Proceedings of EMNLP*, pages 2044–2048.

Douwe Kiela, Laura Rimell, Ivan Vulić, and Stephen Clark. 2015b. Exploiting image generality for lexical entailment detection. In *Proceedings of ACL*, pages 119–124.

Barbara Ann Kipfer. 2009. *Roget's 21st Century Thesaurus (3rd Edition)*. Philip Lief Group.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering*, 16(4):359–389.

Alessandro Lenci and Giulia Benotto. 2012. Identifying hypernyms in distributional semantic spaces. In *Proceedings of *SEM*, pages 75–79.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of ACL*, pages 302–308.

Quan Liu, Hui Jiang, Si Wei, Zhen-Hua Ling, and Yu Hu. 2015. Learning semantic word embeddings based on ordinal knowledge constraints. In *Proceedings of ACL*, pages 1501–1511.

Douglas L. Medin, Mark W. Altom, and Timothy D. Murphy. 1984. Given versus induced category representations: Use of prototype and exemplar information in classification. *Journal of Experimental Psychology*, 10(3):333–352.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. Context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of CoNLL*, pages 51–61.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119.

Michael Mohler, David Bracewell, Marc Tomlinson, and David Hinote. 2013. Semantic signatures for example-based linguistic metaphor detection. In *Proceedings of the First Workshop on Metaphor in NLP*, pages 27–35.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of ACL*, pages 1777–1788.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina Maria Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *Proceedings of NAACL-HLT*.

Nikola Mrkšić, Ivan Vulić, Diarmuid Ó Séaghdha, Ira Leviant, Roi Reichart, Milica Gašić, Anna Korhonen, and Steve Young. 2017. Semantic specialisation of distributional word vector spaces using monolingual and cross-lingual constraints. *Transactions of the ACL*, 5:309–324.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Roberto Navigli, Paola Velardi, and Stefano Faralli. 2011. A graph-based algorithm for inducing lexical taxonomies from scratch. In *Proceedings of IJCAI*, pages 1872–1877.

Kim Anh Nguyen, Maximilian Köper, Sabine Schulte im Walde, and Ngoc Thang Vu. 2017. Hierarchical embeddings for hypernymy detection and directionality. In *Proceedings of EMNLP*, pages 233–243.

Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. 2016. Integrating distributional lexical contrast into word embeddings for antonym-synonym distinction. In *Proceedings of ACL*, pages 454–459.

Maximilian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations. In *Proceedings of NIPS*.

Masataka Ono, Makoto Miwa, and Yutaka Sasaki. 2015. Word embedding-based antonym detection using thesauri and distributional information. In *Proceedings of NAACL-HLT*, pages 984–989.

Dominique Osborne, Shashi Narayan, and Shay Cohen. 2016. Encoding prior knowledge with eigenword embeddings. *Transactions of the ACL*, 4:417–430.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of ACL*, pages 425–430.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. WordNet::Similarity - Measuring the relatedness of oncepts. In *Proceedings of AAAI*, pages 1024–1025.

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

Marek Rei and Ted Briscoe. 2014. Looking for hyponyms in vector space. In *Proceedings of CoNLL*, pages 68–77.

Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. In *Proceedings of COLING*, pages 1025–1036.

Eleanor H. Rosch. 1973. Natural categories. *Cognitive Psychology*, 4(3):328–350.

Enrico Santus, Alessandro Lenci, Qin Lu, and Sabine Schulte im Walde. 2014. Chasing hypernyms in vector spaces with entropy. In *Proceedings of EACL*, pages 38–42.

Enrico Santus, Frances Yung, Alessandro Lenci, and Chu-Ren Huang. 2015. EVALution 1.0: an evolving semantic dataset for training and evaluation of distributional semantic models. In *Proceedings of the 4th Workshop on Linked Data in Linguistics: Resources and Applications*, pages 64–69.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In *Proceedings of ACL*, pages 2389–2398.

Vered Shwartz, Enrico Santus, and Dominik Schlechtweg. 2017. Hypernyms under siege: Linguistically-motivated artillery for hypernymy detection. In *Proceedings of EACL*, pages 65–75.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of ACL*, pages 801–808.

Luu Anh Tuan, Yi Tay, Siu Cheung Hui, and See Kiong Ng. 2016. Learning term embeddings for taxonomic relation identification using dynamic weighting neural network. In *Proceedings of EMNLP*, pages 403–413.

Joseph P. Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of ACL*, pages 384–394.

Shyam Upadhyay, Yogarshi Vyas, Marine Carpuat, and Dan Roth. 2018. Robust cross-lingual hypernymy detection using dependency context. In *Proceedings of NAACL-HLT*.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2016. Order-embeddings of images and language. In *Proceedings of ICLR (Conference Track)*.

Luke Vilnis and Andrew McCallum. 2015. Word representations via Gaussian embedding. In *Proceedings of ICLR (Conference Track)*.

Ivan Vulić, Daniela Gerz, Douwe Kiela, Felix Hill, and Anna Korhonen. 2017. Hyperlex: A large-scale evaluation of graded lexical entailment. *Computational Linguistics*.

Ivan Vulić, Goran Glavaš, Nikola Mrkšić, and Anna Korhonen. 2018. Post-specialisation: Retrofitting vectors of words unseen in lexical resources. In *Proceedings of NAACL-HLT*.

Ivan Vulić, Nikola Mrkšić, and Anna Korhonen. 2017a. Cross-lingual induction and transfer of verb classes based on word vector space specialisation. In *Proceedings of EMNLP*, pages 2536–2548.

Ivan Vulić, Nikola Mrkšić, Roi Reichart, Diarmuid Ó Séaghdha, Steve Young, and Anna Korhonen. 2017b. Morph-fitting: Fine-tuning word vector spaces with simple language-specific rules. In *Proceedings of ACL*, pages 56–68.

Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller. 2014. Learning to distinguish hypernyms and co-hyponyms. In *Proceedings of COLING*, pages 2249–2259.

Julie Weeds, David Weir, and Diana McCarthy. 2004. Characterising measures of lexical distributional similarity. In *Proceedings of COLING*, pages 1015–1021.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. From paraphrase database to compositional paraphrase model and back. *Transactions of the ACL*, 3:345–358.

Zhibiao Wu and Martha Palmer. 1994. Verb semantics and lexical selection. In *Proceedings of ACL*, pages 133–138.

Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. 2014. RC-NET: A general framework for incorporating knowledge into word representations. In *Proceedings of CIKM*, pages 1219–1228.

Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *Proceedings of ACL*, pages 545–550.

Zheng Yu, Haixun Wang, Xuemin Lin, and Min Wang. 2015. Learning term embeddings for hypernymy identification. In *Proceedings of IJCAI*, pages 1390–1397.

Jingwei Zhang, Jeremy Salwen, Michael Glass, and Alfio Gliozzo. 2014. Word semantic representations using bayesian probabilistic tensor factorization. In *Proceedings of EMNLP*, pages 1522–1531.

Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of EMNLP*, pages 1393–1398.

Cross-lingual Abstract Meaning Representation Parsing

Marco Damonte **Shay B. Cohen**
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB, UK
m.damonte@sms.ed.ac.uk
scohen@inf.ed.ac.uk

Abstract

Abstract Meaning Representation (AMR) research has mostly focused on English. We show that it is possible to use AMR annotations for English as a semantic representation for sentences written in other languages. We exploit an AMR parser for English and parallel corpora to learn AMR parsers for Italian, Spanish, German and Chinese. Qualitative analysis show that the new parsers overcome structural differences between the languages. We further propose a method to evaluate the parsers that does not require gold standard data in the target languages. This method highly correlates with the gold standard evaluation, obtaining a (Pearson) correlation of 0.95.

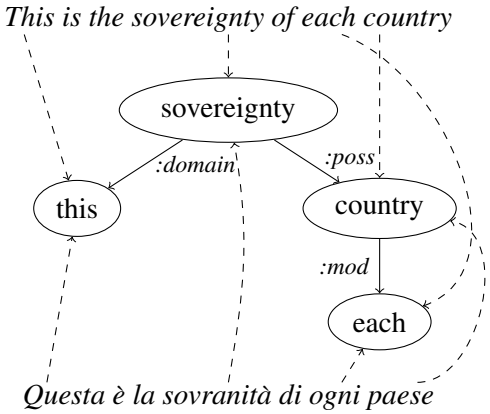


Figure 1: AMR alignments for a English sentence and its Italian translation.

1 Introduction

Abstract Meaning Representation (AMR) parsing is the process of converting natural language sentences into their corresponding AMR representations (Banarescu et al., 2013). An AMR is a graph with nodes representing the concepts of the sentence and edges representing the semantic relations between them. Most available AMR datasets large enough to train statistical models consist of pairs of English sentences and AMR graphs.

The cross-lingual properties of AMR across languages has been the subject of preliminary discussions. The AMR guidelines state that AMR is not an interlingua (Banarescu et al., 2013) and Bojar (2014) categorizes different kinds of divergences in the annotation between English AMRs and Czech AMRs. Xue et al. (2014) show that structurally aligning English AMRs with Czech and Chinese AMRs is not always possible but that refined annotation guidelines suffice to resolve some of these cases. We extend this line of research by exploring whether divergences among languages can be overcome, i.e., we investigate whether it is possible to maintain the AMR annotated for English as a semantic representation for sentences written in other languages, as in Figure 1.

We implement AMR parsers for Italian, Spanish, German and Chinese using annotation projection, where existing annotations are projected from a source language (English) to a target language through a parallel corpus (e.g., Yarowsky et al., 2001; Hwa et al., 2005; Padó and Lapata, 2009; Evang and Bos, 2016). By evaluating the parsers and manually analyzing their output, we show that the parsers are able to recover the AMR structures even when there exist structural differences between the languages, i.e., although AMR is not an interlingua it can act as one. This method also provides a quick way to prototype multilingual AMR parsers, assuming that Part-of-speech (POS) taggers, Named Entity Recognition (NER) taggers and dependency parsers are available for the target languages. We also propose an alternative approach, where Machine Translation (MT) is used to translate the input sentences into English so that an available English AMR parser can

Proceedings of NAACL-HLT 2018, pages 1146–1155
New Orleans, Louisiana, June 1 - 6, 2018.

be employed. This method is an even quicker solution which only requires translation models between the target languages and English.

Due to the lack of gold standard in the target languages, we exploit the English data to evaluate the parsers for the target languages. Henceforth, we will use the term target parser to indicate a parser for a target language. We achieve this by first learning the target parser from the gold standard English parser, and then inverting this process to learn a new English parser from the target parser. We then evaluate the resulting English parser against the gold standard. We call this "full-cycle" evaluation.

Similarly to Evang and Bos (2016), we also directly evaluate the target parser on "silver" data, obtained by parsing the English side of a parallel corpus.

In order to assess the reliability of these evaluation methods, we collected gold standard datasets for Italian, Spanish, German and Chinese by acquiring professional translations of the AMR gold standard data to these languages. We hypothesize that the full-cycle score can be used as a more reliable proxy than the silver score for evaluating the target parser. We provide evidence to this claim by comparing the three evaluation procedures (silver, full-cycle, and gold) across languages and parsers.

Our main contributions are:

- We provide evidence that AMR annotations can be successfully shared across languages.
- We propose two ways to rapidly implement non-English AMR parsers.
- We propose a novel method to evaluate non-English AMR parsers when gold annotations in the target languages are missing. This method highly correlates with gold standard evaluation, obtaining a Pearson correlation coefficient of 0.95.
- We release human translations of an AMR dataset (LDC2015E86) to Italian, Spanish, German and Chinese.

2 Cross-lingual AMR parsing

AMR is a semantic representation heavily biased towards English, where labels for nodes and edges are either English words or Propbank frames (Kingsbury and Palmer, 2002). The goal of AMR is to abstract away from the syntactic realization of the original sentences while maintaining its underlying meaning. As a consequence, different phrasings of one sentence are expected to provide identical AMR representations. This canonicalization does not always hold across languages: two sentences that express the same meaning in two different languages are not guaranteed to produce identical AMR structures (Bojar, 2014; Xue et al., 2014). However, Xue et al. (2014) show that in many cases the unlabeled AMRs are in fact shared across languages. We are encouraged by this finding and argue that it should be possible to develop algorithms that account for some of these differences when they arise. We therefore introduce a new problem, which we call cross-lingual AMR parsing: given a sentence in any language, the goal is to recover the AMR graph that was originally devised for its English translation. This task is harder than traditional AMR parsing as it requires to recover English labels as well as to deal with structural differences between languages, usually referred as translation divergence. We propose two initial solutions to this problem: by annotation projection and by machine translation.

2.1 Method 1: Annotation Projection

AMR is not grounded in the input sentence, therefore there is no need to change the AMR annotation when projecting to another language. We think of English labels for the graph nodes as ones from an independent language, which incidentally looks similar to English. However, in order to train state-of-the-art AMR parsers, we also need to project the alignments between AMR nodes and words in the sentence (henceforth called AMR alignments). We use word alignments, similarly to other annotation projection work, to project the AMR alignments to the target languages.

Our approach depends on an underlying assumption that we make: if a source word is word-aligned to a target word and it is AMR aligned with an AMR node, then the target word is also aligned to that AMR node. More formally, let $S = s_1 \dots s_{|s|}$ be the source language sentence and $T = t_1 \dots t_{|t|}$ be the target language sentence; $A_s(\cdot)$ be the AMR alignment mapping word tokens in S to the set of AMR nodes that are triggered by it; $A_t(\cdot)$ be the same function for T; v be a node in the AMR graph; and finally, $W(\cdot)$ be an alignment that maps a word in S to a subset of words in T. Then, the AMR projection assump-

tion is:

$$\forall i, j, v \;\; t_j \in W(s_i) \wedge v \in A_s(s_i) \Rightarrow v \in A_t(t_j)$$

In the example of Figure 1, *Questa* is word-aligned with *This* and therefore AMR-aligned with the node *this*, and the same logic applies to the other aligned words. The words *is*, *the* and *of* do not generate any AMR nodes, so we ignore their word alignments. We apply this method to project existing AMR annotations to other languages, which are then used to train the target parsers.

2.2 Method 2: Machine Translation

We invoke an MT system to translate the input sentence into English so that we can use an available English parser to obtain its AMR graph. Naturally, the quality of the output graph depends on the quality of the translations. If the automatic translation is close to the reference translation, then the predicted AMR graph will be close to the reference AMR graph. It is therefore evident that this method is not informative in terms of the cross-lingual properties of AMR. However, its simplicity makes it a compelling engineering solution for parsing other languages.

2.3 Evaluation

We now turn to the problem of evaluation. Let us assume that we trained a parser for a target language, for example using the annotation projection method discussed in Section 2.1. In line with rapid development of new parsers, we assume that the only gold AMR dataset available is the one released for English.

SILVER We can generate a silver test set by running an automatic (English) AMR parser on the English side of a parallel corpus and use the output AMRs as references. However, the silver test set is affected by mistakes made by the English AMR parser, therefore it may not be reliable.

FULL-CYCLE In order to perform the evaluation on a gold test set, we propose full-cycle evaluation: after learning the target parser from the English parser, we invert this process to learn a new English parser from the target parser, in the same way that we learned the target parser from the English parser. The resulting English parser is then evaluated against the (English) AMR gold standard. We hypothesize that the score of the new English parser can be used as a proxy to the score of the target parser.

GOLD To show whether the evaluation methods proposed can be used reliably, we also generated gold test AMR datasets for four target languages (Italian, Spanish, German and Chinese). In order to do so, we collected professional translations for the English sentences in the AMR test set.[1] We were then able to create pairs of human-produced sentences with human-produced AMR graphs.

A diagram summarizing the different evaluation stages is shown in Figure 2. In the case of MT-based systems, the full-cycle corresponds to first translating from English to the target language and then back to English (back-translation), and only then parsing the sentences with the English AMR parser. At the end of this process, a noisy version of the original sentence will be returned and its parsed graph will be a noisy version of the graph parsed from the original sentence.

3 Experiments

We run experiments on four languages: Italian, Spanish, German and Chinese. We use Europarl (Koehn, 2005) as the parallel corpus for Italian, Spanish and German, containing around 1.9M sentences for each language pair. For Chinese, we use the first 2M sentences from the United Nations Parallel Corpus (Ziemski et al., 2016). For each target language we extract two parallel datasets of 20,000/2,000/2,000 (train/dev/test) sentences for the two step of the annotation projection (English $\rightarrow$ target and target $\rightarrow$ English). These are used to train the AMR parsers. The projection approach also requires training the word alignments, for which we use all the remaining sentences from the parallel corpora (Europarl for Spanish/German/Italian and UN Parallel Corpus for Chinese). These are also the sentences we use to train the MT models. The gold AMR dataset is LDC2015E86, containing 16,833 training sentences, 1,368 development sentences, and 1,371 testing sentences.

Word alignments were generated using fast_align (Dyer et al., 2013), while AMR alignments were generated with JAMR (Flanigan et al., 2014). AMREager (Damonte et al., 2017) was chosen as the pre-existing English AMR parser. AMREager is an open-source AMR parser that

[1] These datasets are currently available upon request from the authors.

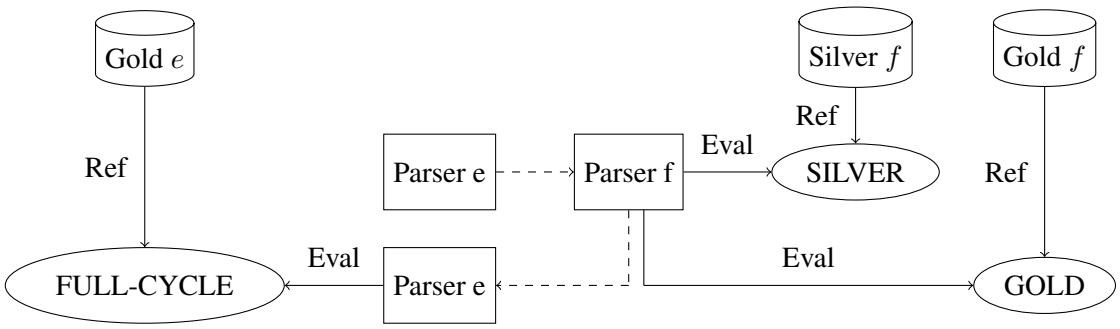


Figure 2: Description of SILVER, FULL-CYCLE and GOLD evaluations. e stands for English and f stands for the target (foreign) language. Dashed lines represent the process of transferring learning across languages (e.g. with annotation projection). SILVER uses a parsed parallel corpus as reference ("Ref"), FULL-CYCLE uses the English gold standard (Gold e) and GOLD uses the target language gold standard we collected (Silver f).

needs only minor modifications for re-use with other languages.[2] It requires tokenization, POS tagging, NER tagging and dependency parsing, which for English, German and Chinese are provided by CoreNLP (Manning et al., 2014). We use Freeling (Carreras et al., 2004) for Spanish, as CoreNLP does not provide dependency parsing for this language. Italian is not supported in CoreNLP: we use Tint (Aprosio and Moretti, 2016), a CoreNLP-compatible NLP pipeline for Italian.

In order to experiment with the approach of Section 2.2, we experimented with translations from Google Translate.[3] As Google Translate has access to a much larger training corpus, we also trained baseline MT models using Moses (Koehn et al., 2007) and Nematus (Sennrich et al., 2017), with the same training data we use for the projection method and default hyper-parameters.

Smatch (Cai and Knight, 2013) is used to evaluate AMR parsers. It looks for the best alignment between the predicted AMR and the reference AMR and it then computes precision, recall and F_1 of their edges. The original English parser achieves 65% Smatch score on the test split of LDC2015E86. Full-cycle and gold evaluations use the same dataset, while silver evaluation is performed on the split of the parallel corpora we reserved for testing. Results are shown in Table 1. The Google Translate system outperforms all other systems, but is not directly comparable to them, as it has the unfair advantage of being trained on a much larger dataset. Due to noisy JAMR alignments and silver training data involved in the annotation projection approach, the MT-based systems give in general better parsing results. The BLEU scores of all translation systems are shown in Table 2.

	System	Silver	Gold	Cycle
IT	Projection	45	43	45
	Moses	51	52	51
	Nematus	49	43	41
	GT	52	58	59
ES	Projection	44	42	44
	Moses	53	53	51
	Nematus	51	43	42
	GT	56	60	60
DE	Projection	45	39	43
	Moses	50	49	49
	Nematus	47	38	39
	GT	54	57	59
ZH	Projection	45	35	32
	Moses	57	42	48
	Nematus	57	39	40
	GT	64	50	55

Table 1: Silver, gold and full-cycle Smatch scores for projection-based and MT-based systems.

There are several sources of noise in the annotation projection method, which affect the parsing results: 1) the parsers are trained on silver data obtained by an automatic parser for English; 2) the projection uses noisy word alignments; 3) the AMR alignments on the source side are also noisy; 4) translation divergences exist between the languages, making it sometimes difficult to project the annotation without loss of information.

[2]The multilingual adaptation of AMREager is available at http://www.github.com/mdtux89/amr-eager-multilingual. A demo is available at http://cohort.inf.ed.ac.uk/amreager.html.

[3]https://translate.google.com/toolkit.

Model	Moses	Nematus	GT
EN-IT	23.83	21.27	61.31
IT-EN	23.74	19.77	42.20
EN-ES	29.00	26.14	78.14
ES-EN	27.66	21.63	50.78
EN-DE	15.47	15.74	63.48
DE-EN	21.50	14.96	41.78
EN-ZH	9.19	8.67	26.75
ZH-EN	10.81	10.37	22.21

Table 2: BLEU scores for Moses, Nematus and Google Translate (GT) on the (out-of-domain) LDC2015E86 test set

4 Qualitative Analysis

Figure 3 shows examples of output parses[4] for all languages, including the AMR alignments by-product of the parsing process, that we use to discuss the mistakes made by the parsers.

In the Italian example, the only evident error is that *Infine* (*Lastly*) should be ignored. In the Spanish example, the word *medida* (*measure*) is wrongly ignored: it should be used to generate a child of the node *impact-01*. Some of the *:ARG* roles are also not correct. In the German example, *meines* (*my*) should reflect the fact that the speaker is talking about his own country. Finally, in the Chinese example, there are several mistakes including yet another concept identification mistake: *intend-01* is erroneously triggered.

Most mistakes involve concept identification. In particular, relevant words are often erroneously ignored by the parser. This is directly related to the problem of noisy word alignments in annotation projection: the parser learns what words are likely to trigger a node (or a set of nodes) in the AMR by looking at their AMR alignments (which are induced by the word alignments). If an important word consistently remains unaligned, the parser will erroneously learn to discard it. More accurate alignments are therefore crucial in order to achieve better parsing results. We computed the percentage of words in the training data that are learned to be non-content-bearing in each parser and we found that the Chinese parser, which is our least accurate parser, is the one that most suffer from this, with 33% non-content-bearing words. On the other hand, in the German parser, which is the highest scoring, only 26% of the words are non-content-bearing, which is the lowest percentage amongst all parsers.

[4]In this section, all parsed graphs were generated with the projection-based system of Section 2.1.

4.1 Translational Divergence

In order to investigate the hypothesis that AMR can be shared across these languages, we now look at translational divergence and discuss how it affects parsing, following the classification used in previous work (Dorr et al., 2002; Dorr, 1994), which identifies classes of divergences for several languages. Sulem et al. (2015) also follow the same categorization for French.

Figure 4 shows six sentences displaying these divergences. The aim of this analysis is to assess how the parsers deal with the different kind of translational divergences, regardless of the overall quality of the output.

Categorical. This divergence happens when two languages use different POS tags to express the same meaning. For example, the English sentence *I am jealous of you* is translated into Spanish as *Tengo envidia de ti* (*I have jealousy of you*). The English adjective *jealous* is translated in the Spanish noun *envidia*. In Figure 4a we note that the categorical divergence does not create problems since the parsers correctly recognized that *envidia* (*jealousy*/*envy*) should be used as the predicate, regardless of its POS.

Conflational. This divergence happens when verbs expressed in a language with a single word can be expressed with more words in another language. Two subtypes are distinguished: *manner* and *light verb*. Manner refers to a manner verb that is mapped to a motion verb plus a manner-bearing word. For example, *We will answer* is translated in the Italian sentence *Noi daremo una riposta* (*We will give an answer*), where *to answer* is translated as *daremo una risposta* (*will give an answer*). Figure 4b shows that the Italian parser generates a sensible output for this sentence by creating a single node labeled *answer-01* for the expression *dare una riposta*.

In a light verb conflational divergence, a verb is mapped to a light verb plus an additional meaning unit, such as when *I fear* is translated as *Io ho paura* (*I have fear*) in Italian: *to fear* is mapped to the light verb *ho* (*have*) plus the noun *paura* (*fear*). Figure 4e shows that also this divergence is dealt properly by the Italian parser: *ho paura* correctly triggers the root *fear-01*.

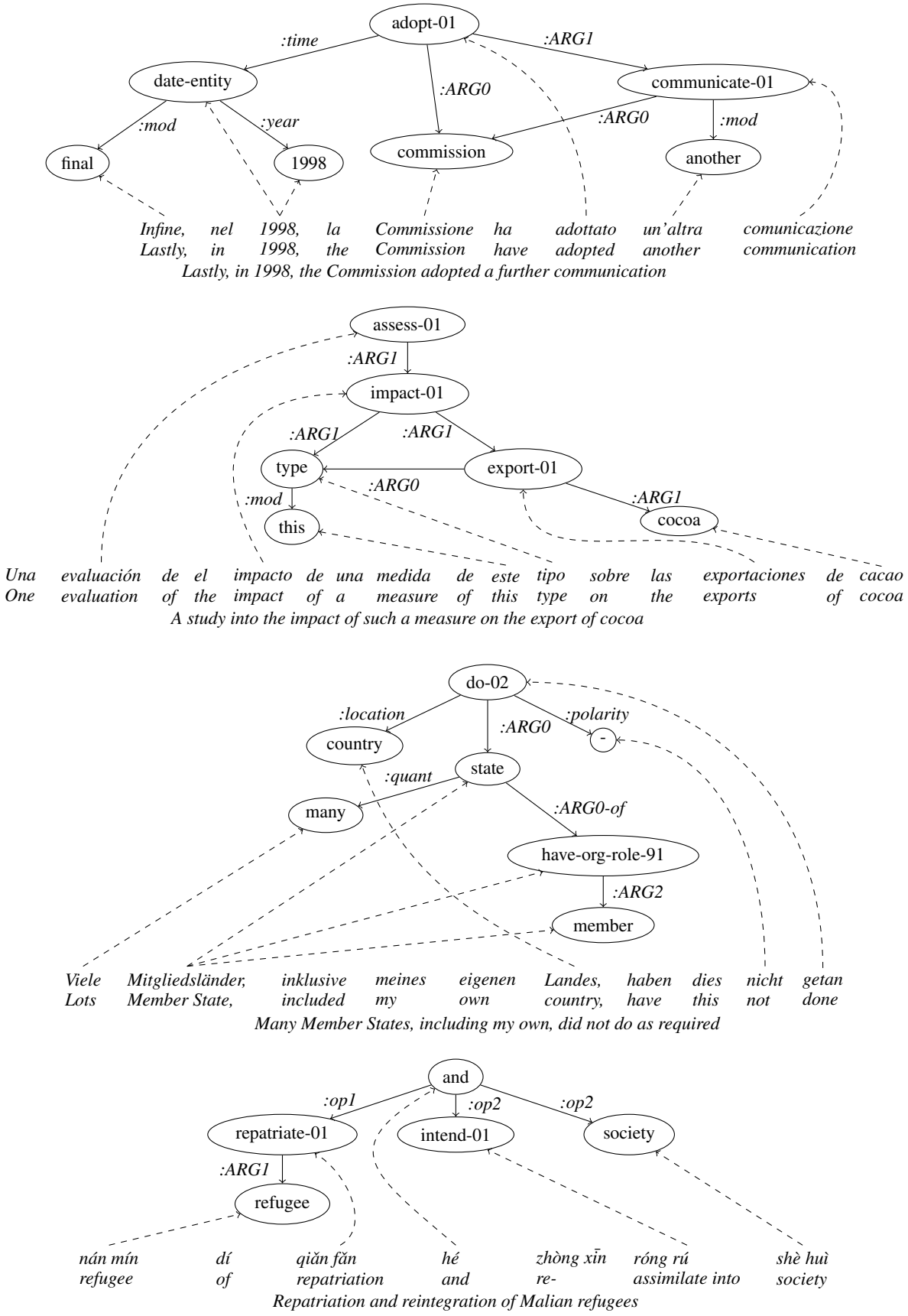


Figure 3: Parsed AMR graph and alignments (dashed lines) for an Italian sentence, a Spanish sentence, a German sentences and a Chinese sentence.

Structural. This divergence happens when verb arguments result in different syntactic configurations, for example, due to an additional PP attachment. When translating *He entered the house* with *Lui è entrato nella casa* (*He entered in the house*), the Italian translation has an additional *in* preposition. Also this parsed graph, in Figure 4c, is structurally correct. The missing node *he* is due to pronoun-dropping, which is frequent in Italian.

Head swapping. This divergence occurs when the direction of the dependency between two words is inverted. For example, *I like eating*, where *like* is head of *eating*, becomes *Ich esse gern* (*I eat likingly*) in German, where the dependency is inverted. Unlike all other examples, in this case, the German parser does not cope well with this divergence: it is unable to recognize *like-01* as the main concept in the sentence, as shown in Figure 4d.

Thematic. Finally, the parse of Figure 4f has to deal with a thematic divergence, which happens when the semantic roles of a predicate are inverted. In the sentence *I like grapes*, translated to Spanish as *Me gustan uvas*, *I* is the subject in English while *Me* is the object in Spanish. Even though we note an erroneous reentrant edge between *grape* and *I*, the thematic divergence does not create problems: the parser correctly recognizes the *:ARG0* relationship between *like-01* and *I* and the *:ARG1* relationship between *like-01* and *grape*. In this case, the edge labels are important, as this type of divergence is concerned with the semantic roles.

5 Discussion

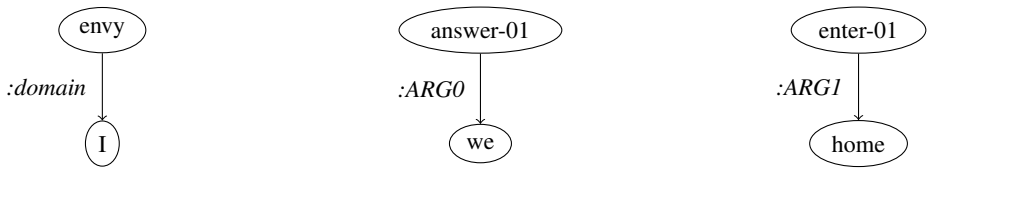
Can AMR be shared across these languages? As mentioned in Section 2.2, the MT-based systems are not helpful in answering this question and we instead focus on the projection-based parsers. Qualitative analysis showed that the parsers are able to overcome translational divergence and that concept identification must be more accurate in order to provide good parsing results. We therefore argue that the suboptimal performance of the parsers in terms of Smatch scores is due to the many sources of noise in the annotation projection approach rather than instability of AMR across languages. We provide strong evidence that cross-lingual AMR parsing is indeed feasible and hope that the release of the gold standard test sets will motivate further work in this direction.

Are silver and full-cycle evaluations reliable? We computed the Pearson correlation coefficients for the Smatch scores of Table 1 to determine how well silver and full-cycle correlate with gold evaluation. Full-cycle correlates better than silver: the Pearson coefficient is 0.95 for full-cycle and 0.47 for silver. Figure 5 shows linear regression lines. Unlike silver, full-cycle uses the same dataset as gold evaluation and it does not contain parsing mistakes, which makes it more reliable than silver. Interestingly, if we ignore the scores obtained for Chinese, the correlation between silver and gold dramatically increases, perhaps indicating that Europarl is more suitable than the UN corpus for this task: the Pearson coefficient becomes 0.97 for full-cycle and 0.87 for silver. A good proxy for gold evaluation should rank different systems similarly. We hence computed the Kendall-tau score (Kendall, 1945), a measure for similarity between permutations, of the rankings extracted from Table 1. The results further confirm that full-cycle approximate gold better than silver does: the score is 0.40 for silver and 0.82 for full-cycle. Full cycle introduces additional noise but it is not as expensive as gold and is more reliable than silver.

6 Related Work

AMR parsing for languages other than English has made only a few steps forward. In previous work (Li et al., 2016; Xue et al., 2014; Bojar, 2014), nodes of the target graph were labeled with either English words or with words in the target language. We instead use the AMR annotation used for English for the target language as well, without translating any word. To the best of our knowledge, the only previous work that attempts to automatically parse AMR graphs for non-English sentences is by Vanderwende et al. (2015). Sentences in several languages (French, German, Spanish and Japanese) are parsed into a logical representation, which is then converted to AMR using a small set of rules. A comparison with this work is difficult, as the authors do not report results for the parsers (due to the lack of an annotated corpus) or release their code.

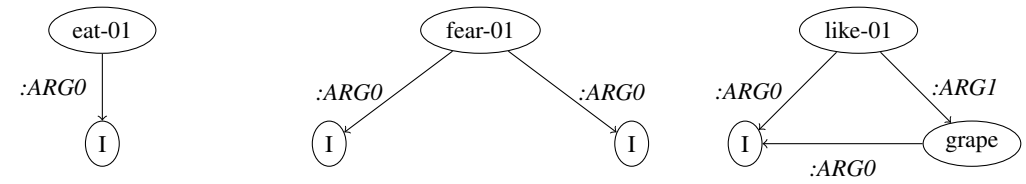
Besides AMR, other semantic parsing frameworks for non-English languages have been investigated (Hoffman, 1992; Cinková et al., 2009; Gesmundo et al., 2009; Evang and Bos, 2016). Evang and Bos (2016) is the most closely related to our



(a) ES: Tengo envidia de ti
(I am jealous of you)

(b) IT: Noi daremo una risposta
(We will answer)

(c) IT: Lui è entrato nella casa
(He entered the house)



(d) DE: Ich esse gern
(I like eating)

(e) IT: Io ho paura
(I fear)

(f) ES: Me gustan uvas
(I like grapes)

Figure 4: Parsing examples in several languages involving common translational divergence phenomena: (a) contains a categorical divergence, (b) and (e) conflational divergences, (c) a structural divergence, (d) an head swapping and (f) a thematic divergence.

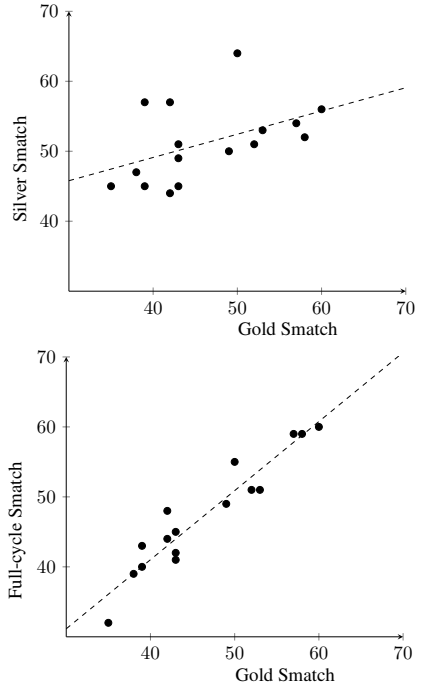


Figure 5: Linear regression lines for silver and full-cycle.

work as it uses a projection mechanism similar to ours for CCG. A crucial difference is that, in order to project CCG parse trees to the target languages, they only make use of literal translation. Previous work has also focused on assessing the stability across languages of semantic frameworks such as AMR (Xue et al., 2014; Bojar, 2014), UCCA (Sulem et al., 2015) and Propbank (Van der Plas et al., 2010).

Cross-lingual techniques can cope with the lack of labeled data on languages when this data is available in at least one language, usually English. The annotation projection method, which we follow in this work, is one way to address this problem. It was introduced for POS tagging, base noun phrase bracketing, NER tagging, and inflectional morphological analysis (Yarowsky et al., 2001) but it has also been used for dependency parsing (Hwa et al., 2005), role labeling (Padó and Lapata, 2009; Akbik et al., 2015) and semantic parsing (Evang and Bos, 2016). Another common thread of cross-lingual work is model transfer, where parameters are shared across languages (Zeman and Resnik, 2008; Cohen and Smith, 2009; Cohen et al., 2011; McDonald et al., 2011; Søgaard, 2011).

7 Conclusions

We introduced the problem of parsing AMR structures, annotated for English, from sentences written in other languages as a way to test the cross-lingual properties of AMR. We provided evidence that AMR can be indeed shared across the lan-

guages tested and that it is possible to overcome translational divergences. We further proposed a novel way to evaluate the target parsers that does not require manual annotations of the target language. The full-cycle procedure is not limited to AMR parsing and could be used for other cross-lingual problems in NLP. The results of the projection-based AMR parsers indicate that there is a vast room for improvements, especially in terms of generating better alignments. We encourage further work in this direction by releasing professional translations of the AMR test set into four languages.

Acknowledgments

The authors would like to thank the three anonymous reviewers and Sameer Bansal, Gozde Gul Sahin, Sorcha Gilroy, Ida Szubert, Esma Balkır, Nikos Papasarantopoulos, Joana Ribeiro, Shashi Narayan, Toms Bergmanis, Clara Vania, Yang Liu and Adam Lopez for their helpful comments. This research was supported by a grant from Bloomberg and by the H2020 project SUMMA, under grant agreement 688139.

References

Alan Akbik, Laura Chiticariu, Marina Danilevsky, Yunyao Li, Shivakumar Vaithyanathan, and Huaiyu Zhu. 2015. Generating high quality proposition banks for multilingual semantic role labeling. In *Proceedings of ACL*.

Alessio Palmero Aprosio and Giovanni Moretti. 2016. Italy goes to stanford: a collection of corenlp modules for italian. *arXiv preprint arXiv:1609.06204* .

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. *Linguistic Annotation Workshop* .

Zdenka Urešová Jan Hajic Ondrej Bojar. 2014. Comparing czech and english amrs. In *Workshop on Lexical and Grammatical Resources for Language Processing*.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. *Proceedings of ACL* .

Xavier Carreras, Isaac Chao, Lluís Padró, and Muntsa Padró. 2004. Freeling: An open-source suite of language analyzers. In *Proceedings of LREC*.

Silvie Cinková, Josef Toman, Jan Hajic, Kristỳna Cermáková, Václav Klimeš, Lucie Mladová, Jana Šindlerová, Kristỳna Tomšu, and Zdenek Zabokrtskỳ. 2009. Tectogrammatical annotation of the wall street. *The Prague Bulletin of Mathematical Linguistics* .

Shay B. Cohen, Dipanjan Das, and Noah A. Smith. 2011. Unsupervised structure prediction with non-parallel multilingual guidance. In *Proceedings of EMNLP*.

Shay B Cohen and Noah A Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of NAACL-HLT*.

Marco Damonte, Shay B Cohen, and Giorgio Satta. 2017. An incremental parser for abstract meaning representation. In *Proceedings of EACL*.

Bonnie J Dorr. 1994. Machine translation divergences: A formal description and proposed solution. *Computational Linguistics* 20(4):597–633.

Bonnie J Dorr, Lisa Pearl, Rebecca Hwa, and Nizar Habash. 2002. Improved word-level alignment: Injecting knowledge about mt divergences. Technical report, DTIC Document.

Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of NAACL-HLT*.

Kilian Evang and Johan Bos. 2016. Cross-lingual learning of an open-domain semantic parser. In *Proceedings of COLING*.

Jeffrey Flanigan, Sam Thomson, Jaime G Carbonell, Chris Dyer, and Noah A Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. *Proceedings of ACL* .

Andrea Gesmundo, James Henderson, Paola Merlo, and Ivan Titov. 2009. A latent variable model of synchronous syntactic-semantic parsing for multiple languages. In *Proceedings of CoNLL*.

Beryl Hoffman. 1992. A ccg approach to free word order languages. In *Proceedings of ACL*.

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

Maurice G Kendall. 1945. The treatment of ties in ranking problems. *Biometrika* 33(3):239–251.

Paul Kingsbury and Martha Palmer. 2002. From treebank to propbank. *Proceedings of LREC* .

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*. volume 5, pages 79–86.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL*.

Bin Li, Yuan Wen, Lijun Bu, Weiguang Qu, and Nianwen Xue. 2016. Annotating the little prince with chinese amrs. *Linguistic Annotation Workshop* .

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of ACL*.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of EMNLP*.

Sebastian Padó and Mirella Lapata. 2009. Cross-lingual annotation projection for semantic roles. *Journal of Artificial Intelligence Research* 36(1):307–340.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. 2017. Nematus: a toolkit for neural machine translation. In *Proceedings of EACL*.

Anders Søgaard. 2011. Data point selection for cross-language adaptation of dependency parsers. In *Proceedings of ACL-HLT*.

Elior Sulem, Omri Abend, and Ari Rappoport. 2015. Conceptual annotations preserve structure across translations: A french-english case study. In *Workshop on Semantics-Driven Statistical Machine Translation*.

Lonneke Van der Plas, Tanja Samardžić, and Paola Merlo. 2010. Cross-lingual validity of propbank in the manual annotation of French. In *Linguistic Annotation Workshop*.

Lucy Vanderwende, Arul Menezes, and Chris Quirk. 2015. An amr parser for english, french, german, spanish and japanese and a new amr-annotated corpus. In *Proceedings of NAACL-HLT*.

Nianwen Xue, Ondrej Bojar, Jan Hajic, Martha Palmer, Zdenka Uresova, and Xiuhong Zhang. 2014. Not an interlingua, but close: Comparison of english amrs to chinese and czech. In *Proceedings of LREC*.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of NAACL-HLT*.

Daniel Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In *Proceedings of IJCNLP*.

Michal Ziemski, Marcin Junczys-Dowmunt, and Bruno Pouliquen. 2016. The united nations parallel corpus v1. 0. In *Proceedings of LREC*.

Sentences with Gapping: Parsing and Reconstructing Elided Predicates

Sebastian Schuster **Joakim Nivre**‡ **Christopher D. Manning**

Departments of Linguistics and Computer Science, Stanford University
{sebschu,manning}@stanford.edu
‡Department of Linguistics and Philology, Uppsala University
joakim.nivre@lingfil.uu.se

Abstract

Sentences with gapping, such as *Paul likes coffee and Mary tea*, lack an overt predicate to indicate the relation between two or more arguments. Surface syntax representations of such sentences are often produced poorly by parsers, and even if correct, not well suited to downstream natural language understanding tasks such as relation extraction that are typically designed to extract information from sentences with canonical clause structure. In this paper, we present two methods for parsing to a Universal Dependencies graph representation that explicitly encodes the elided material with additional nodes and edges. We find that both methods can reconstruct elided material from dependency trees with high accuracy when the parser correctly predicts the existence of a gap. We further demonstrate that one of our methods can be applied to other languages based on a case study on Swedish.

1 Introduction

Sentences with gapping (Ross, 1970) such as *Paul likes coffee and Mary tea* are characterized by having one or more conjuncts that contain multiple arguments or modifiers of an elided predicate. In this example, the predicate *likes* is elided for the relation *Mary likes tea*. While these sentences appear relatively infrequently in most written texts, they are often used to convey a lot of factual information that is highly relevant for language understanding (NLU) tasks such as open information extraction and semantic parsing. For example, consider the following sentence from the WSJ portion of the Penn Treebank (Marcus et al., 1993).

(1) Unemployment has reached 27.6% in Azerbaijan, 25.7% in Tadzhikistan, 22.8% in Uzbekistan, 18.8% in Turkmenia, 18% in Armenia and 16.3% in Kirgizia, [...]

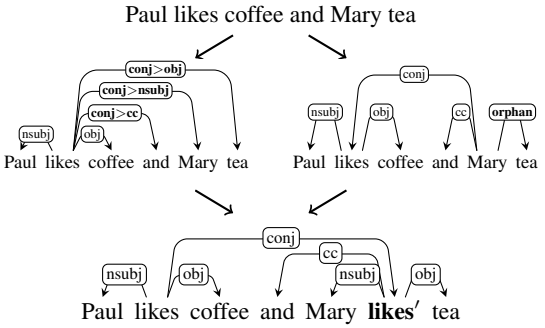


Figure 1: Overview of our two approaches. Both methods first parse a sentence with gapping to one of two different dependency tree representations and then reconstruct the elided predicate from this tree.

To extract the information about unemployment rates in the various countries, an NLU system has to identify that the percentages indicate unemployment rates and the locational modifiers indicate the corresponding country. Given only this sentence, or this sentence and a strict surface syntax representation that does not indicate elided predicates, this is a challenging task. However, given a dependency graph that reconstructs the elided predicate for each conjunct, the problem becomes much easier and methods developed to extract information from dependency trees of clauses with canonical structures are much more likely to extract the correct information from a gapped clause.

While gapping constructions receive a lot of attention in the theoretical syntax literature (e.g., Ross 1970; Jackendoff 1971; Steedman 1990; Coppock 2001; Osborne 2006; Johnson 2014; Toosarvandani 2016; Kubota and Levine 2016), they have been almost entirely neglected by the NLP community so far. The Penn Treebank explicitly annotates gapping constructions, by co-indexing arguments in the clause with a predicate and the clause with the gap, but these co-indices are not included in the standard parsing metrics

Proceedings of NAACL-HLT 2018, pages 1156–1168
New Orleans, Louisiana, June 1 - 6, 2018.

and almost all parsers ignore them.[1] Despite the sophisticated analysis of gapping within CCG (Steedman, 1990), sentences with gapping were deemed too difficult to represent within the CCGBank (Hockenmaier and Steedman, 2007). Similarly the treebanks for the Semantic Dependencies Shared Task (Oepen et al., 2015) exclude all sentences from the Wall Street Journal that contain gapping. Finally, while the tectogrammatical layer of the Prague Dependency Treebank (Bejček et al., 2013) as well as the enhanced Universal Dependencies (UD) representation (Nivre et al., 2016) provide an analysis with reconstructed nodes for gapping constructions, there exist no methods to automatically parse to these representations.

Here, we provide the first careful analysis of parsing of gapping constructions, and we present two methods for reconstructing elided predicates in sentences with gapping within the UD framework. As illustrated in Figure 1, we first parse to a dependency tree and then reconstruct the elided material. The methods differ in how much information is encoded in the dependency tree. The first method adapts an existing procedure for parsing sentences with elided function words (Seeker et al., 2012), which uses composite labels that can be deterministically turned into dependency graphs in most cases. The second method is a novel procedure that relies on the parser only to identify a gap, and then employs an unsupervised method to reconstruct the elided predicates and reattach the arguments to the reconstructed predicate. We find that both methods can reconstruct elided predicates with very high accuracy from gold standard dependency trees. When applied to the output of a parser, which often fails to identify gapping, our methods achieve a sentence-level accuracy of 32% and 34%, significantly outperforming the recently proposed constituent parser by Kummerfeld and Klein (2017).

2 Background

2.1 Gapping constructions

Gapping constructions in English come in many forms that can be broadly classified as follows.

(2) *Single predicate gaps*:
John **bought** books, and Mary ___ flowers.

(3) *Contiguous predicate-argument gap (including ACCs)*:
Eve **gave flowers** to Al and Sue ___ to Paul.
Eve **gave** a CD to Al and ___ roses to Sue.

(4) *Non-contiguous predicate-argument gap*:
Arizona **elected** Goldwater **Senator**, and Pennsylvania ___ Schwelker ___.
(Jackendoff, 1971)

(5) *Verb cluster gap*:
I **want to try to begin to write a novel** and
... Mary ___ a play.
... Mary ___ to write a play.
... Mary ___ to begin to write a play.
... Mary ___ to try to begin to write a play.
(Ross, 1970)

The defining characteristic of gapping constructions is that there is a clause that lacks a predicate (the *gap*) but still contains two or more arguments or modifiers of the elided predicate (the *remnants* or *orphans*). In most cases, the remnants have a corresponding argument or modifier (the *correspondent*) in the clause with the overt predicate.

These types of gapping also make up the majority of attested constructions in other languages. However, Wyngaerd (2007) notes that Dutch permits gaps in relative clauses, and Farudi (2013) notes that Farsi permits gaps in finite embedded clauses even if the overt predicate is not embedded.[2]

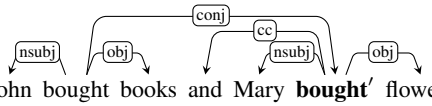
2.2 Target representation

We work within the UD framework, which aims to provide cross-linguistically consistent dependency annotations that are useful for NLP tasks. UD defines two types of representation: the *basic* UD representation which is a strict surface syntax dependency tree and the *enhanced* UD representation (Schuster and Manning, 2016) which may be a graph instead of a tree and may contain additional nodes. The analysis of gapping in the enhanced representation makes use of copy nodes for elided predicates and additional edges for elided arguments, which we both try to automatically reconstruct in this paper. In the simple case in which only one predicate was elided, there is exactly one

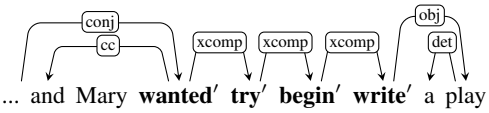
[1] To the best of our knowledge, the parser by Kummerfeld and Klein (2017) is the only parser that tries to output the co-indexing of constituents in clauses with gapping but they lack an explicit evaluation of their co-indexing prediction accuracy.

[2] See Johnson (2014) or Schuster et al. (2017) for a more comprehensive overview of cross-linguistically attested gapping constructions.

copy node for the elided predicate, which leads to a structure that is identical to the structure of the same sentence without a gap.[3]

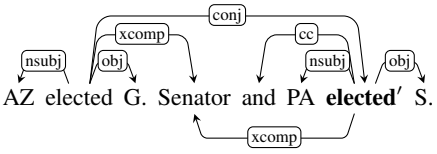


If a clause contains a more complex gap, the enhanced representation contains copies for all content words that are required to attach the remnants.



The motivation behind this analysis is that the semantically empty markers *to* are not needed for interpreting the sentence and minimizing the number of copy nodes leads to less complex graphs.

Finally, if a core argument was elided along with the predicate, we introduce additional dependencies between the copy nodes and the shared arguments, as for example, the open clausal complement (`xcomp`) dependency between the copy node and *Senator* in the following example.

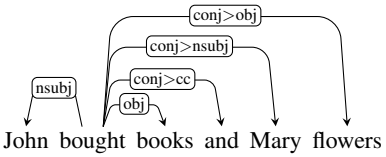


The rationale for not copying all arguments is again to keep the graph simple, while still encoding all relations between content words. Arguments can be arbitrarily complex and it seems misguided to copy entire subtrees of arguments which, e.g., could contain multiple adverbial clauses. Note that linking to existing nodes would not work in the case of verb clusters because they do not satisfy the subtree constraint.

3 Methods

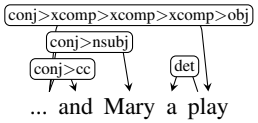
3.1 Composite relations

Our first method adapts one of the procedures by Seeker et al. (2012), which represents gaps in dependency trees by attaching dependents of an elided predicate with composite relations. These relations represent the dependency path that would have existed if nothing had been elided. For example, in the following sentence, the verb *bought*, which would have been attached to the head of the first conjunct with a `conj` relation, was elided from the second conjunct and hence all nodes that would have depended on the elided verb, are attached to the first conjunct using a composite relation consisting of `conj` and the type of argument.



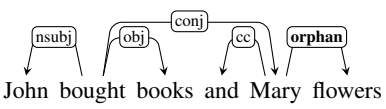
The major advantage of this approach is that the dependency tree contains information about the types of arguments and so it should be straightforward to turn dependency trees of this form into enhanced UD graphs. For most dependency trees, one can obtain the enhanced UD graph by splitting the composite relations into its atomic parts and inserting copy nodes at the splitting points.[4]

At the same time, this approach comes with the drawback of drastically increasing the label space. For sentences with more complex gaps as in (5), one has to use composite relations that consist of more than two atomic relations and theoretically, the number of composite relations is unbounded:



3.2 Orphan procedure

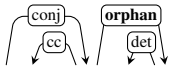
Our second method also uses a two-step approach to resolve gaps, but compared to the previous method, it puts less work on the parser. We first parse sentences to the basic UD v2 representation, which analyzes gapping constructions as follows. One remnant is promoted to be the head of the clause and all other remnants are attached to the promoted phrase. For example, in this sentence, the subject of the second clause, *Mary*, is the head of the clause and the other remnant, *flowers*, is attached to *Mary* with the special `orphan` relation:



[3] To enhance the readability of our examples, we place the copy node in the sentence where the elided predicate would have been pronounced. However, as linear order typically does not matter for extracting information with dependency patterns, our procedures only try to recover the structure of canonical sentences but not their linear order.

[4] Note that this representation does not indicate conjunct boundaries, and for sentences with multiple gapped conjuncts, it is thus unclear how many copy nodes are required.

This analysis can also be used for more complex gaps, as in the example with a gap that consists of a chain of non-finite embedded verbs in (5).



When parsing to this representation, the parser only has to identify that there is a gap but does not have to recover the elided material or determine the type of remnants. As a second step, we use an unsupervised procedure to determine which nodes to copy and how and where to attach the remnants. In developing this procedure, we made use of the fact that in the vast majority of cases, all arguments and modifiers that are expressed in gapped conjunct are also expressed in the full conjunct. The problem of determining which nodes to copy and which relations to use can thus be reduced to the problem of aligning arguments in the gapped conjunct to arguments in the full conjunct. We apply the following procedure to all sentences that contain at least one `orphan` relation.

1. Create a list F of arguments of the head of the full conjunct by considering all core argument dependents of the conjunct's head as well as clausal and nominal non-core dependents, and adverbial modifiers.
2. Create a list G of arguments in the gapped conjunct that contains the head of the gapped conjunct and all its `orphan` dependents.
3. Find the highest-scoring monotonic alignment of arguments in G to arguments in F.
4. Copy the head of the full conjunct and attach the copy node c to the head of the full conjunct with the original relation of the head of the gapped conjunct (usually `conj`).
5. For each argument $g \in G$ that has been aligned to $f \in F$, attach g to c with the same relation as the parent relation of f, e.g., if f is attached to the head of the full conjunct with an `nsubj` relation, also attach g to c with an `nsubj` relation. Attach arguments $g' \in G$ that were not aligned to any token in F to c using the general `dep` relation.
6. For each copy node c, add dependencies to all core arguments of the original node which do not have a corresponding remnant in the gapped clause. For example, if the full conjunct contains a subject, an object, and an oblique modifier but the clause with the gap, only a subject and an oblique modifier, add an object dependency between the copy node and the object in the full conjunct.

A crucial step is the third step, determining the highest-scoring alignment. This can be done straightforwardly with the sequence alignment algorithm by Needleman and Wunsch (1970) if one defines a similarity function $sim(g, f)$ that returns a similarity score between the arguments g and f. We defined sim based on the intuitions that often, parallel arguments are of the same syntactic category, that they are introduced by the same function words (e.g., the same preposition), and that they are closely related in meaning. The first intuition can be captured by penalizing mismatching POS tags, and the other two by computing the distance between argument embeddings. We compute these embeddings by averaging over the 100-dim. pretrained GloVe (Pennington et al., 2014) embeddings for each token in the argument. Given the POS tags t_g and t_f and the argument embeddings v_g and v_f, sim is defined as follows.[5]

$$sim(g, f) = -\|v_g - v_f\|_2 + \mathbb{1}\left[t_g = t_f\right] \times pos\_mismatch\_penalty$$

We set $pos\_mismatch\_penalty$, a parameter that penalizes mismatching POS tags, to -2.[6]

This procedure can be used for almost all sentences with gapping constructions. However, if parts of an argument were elided along with the main predicate, it can become necessary to copy multiple nodes. We therefore consider the alignment not only between complete arguments in the full clause and the gapped clause but also between partial arguments in the full clause and the complete arguments in the gapped clause. For example, for the sentence "*Mary wants to write a play and Sue a book*" the complete arguments of the full clause are {*Mary*, *to write a play*} and the arguments of the gapped clause are {*Sue*, *a book*}. In this case, we also consider the partial arguments {*Mary*, *a play*} and if the arguments of the gapped

[5] As suggested by one of the reviewers, we also ran a post-hoc experiment with a simpler similarity score function without the embedding distance term, which only takes into account whether the POS tags match. We found that quantitatively, the embeddings do not lead to significant better scores on the test set according to our metrics but qualitatively, they lead to better results for the examples with verb cluster gaps.

[6] We optimized this parameter on the training set by trying integer values from -1 to -15.

	EWT			GAPPING			EWT + GAPPING (COMBINED)		
	Train	**Dev**	**Test**	**Train**	**Dev**	**Test**	**Train**	**Dev**	**Test**
sentences	12,543	2,002	2,077	164	79	79	12,707	2,081	2,156
tokens	204,585	25,148	25,096	4,698	2,383	2,175	209,283	27,531	27,271
sentences with gapping	21 0.15%	1 0.05%	1 0.05%	164 100%	79 100%	79 100%	185 1.46%	80 3.84%	80 3.71%
copy nodes	22	2	1	201	96	102	223	98	103
unique composite relations	16	6	2	41	29	31	46	29	31

Table 1: Treebank statistics. The *copy nodes* row lists the number of copy nodes and the *unique composite relations* row lists the number of unique composite relations in the treebanks annotated according to the COMPOSITE analysis. The percentages are relative to the total number of sentences.

Gap type	**Frequency**
Single predicate	172
Contiguous predicate-argument	140
Non-contiguous predicate-argument	9
Verb cluster	24

Table 2: Distribution of gap types in our corpus. The classification is according to the four types of gaps that we discussed in Section 2.1.

conjunct align better to the partial arguments, we use this alignment. However, now that the token *write* is part of the dependency path between *want* and *play*, we also have to make a copy of *write* to reconstruct the UD graph of the gapped clause.

4 Experiments

Both methods rely on a dependency parser followed by a post-processing step. We evaluated the individual steps and the end-to-end performance.

4.1 Data

We used the UD English Web Treebank v2.1 (henceforth EWT; Silveira et al., 2014; Nivre et al., 2017) for training and evaluating parsers. As the treebank is relatively small and therefore only contains very few sentences with gapping, we also extracted gapping constructions from the WSJ and Brown portions of the PTB (Marcus et al., 1993) and the GENIA corpus (Ohta et al., 2002). Further, we copied sentences from the Wikipedia page on gapping[7] and from published papers on gapping. The sentences in the EWT already contain annotations with the `orphan` relation and copy nodes for the enhanced representation, and we manually added both of these annotations for the remaining examples. The composite relations can be automatically obtained from the enhanced representation by removing the copy nodes and concatenating the dependency labels, which we did to build the training and test corpus for the composite relation procedure. Table 1 shows properties of the data splits of the original treebank, the additional sentences with gapping, and their combination; Table 2 shows the number of sentences in our corpus for each of the gap types.

[7] `https://en.wikipedia.org/wiki/Gapping`, accessed on Aug 24, 2017.

4.2 Parsing experiments

Parser We used the parser by Dozat and Manning (2017) for parsing to the two different intermediate dependency representations. This parser is a graph-based parser (McDonald et al., 2005) that uses a biLSTM to compute token representations and then uses a multi-layer perceptron with biaffine attention to compute arc and label scores.

Setup We trained the parser on the COMBINED training corpus with gold tokenization, and predicted fine-grained and universal part-of-speech tags, for which we used the tagger by Dozat et al. (2017). We trained the tagger on the COMBINED training corpus. As pre-trained embeddings, we used the word2vec (Mikolov et al., 2013) embeddings that were provided for the CoNLL 2017 Shared Task (Zeman et al., 2017), and we used the same hyperparameters as Dozat et al. (2017).

Evaluation We evaluated the parseability of the two dependency representations using labeled and unlabeled attachment scores (LAS and UAS). Further, to specifically evaluate how well parsers are able to parse gapping constructions according to the two annotation schemes, we also computed the LAS and UAS just for the head tokens of remnants (LAS_g and UAS_g). For all our metrics, we excluded punctuation tokens. To determine sta-

		EWT		GAPPING	
		UAS	LAS	UAS	LAS
Dev	ORPHAN	**90.57**	87.32	**89.34**	**85.69****
	COMPOSITE	90.46	**87.37**	88.86	84.21
Test	ORPHAN	90.42	87.06	**87.44**	**83.97****
	COMPOSITE	**90.54**	**87.33**	86.51	81.69

Table 3: Labeled (LAS) and unlabeled attachment score (UAS) of parsers trained and evaluated on the UD representation (ORPHAN) and the composite relations representation (COMPOSITE) on the development and test sets of the EWT and the GAPPING treebank. ** indicates that results differ significantly at $p < 0.01$.

	Development		**Test**	
	UAS_g	LAS_g	UAS_g	LAS_g
ORPHAN	**72.36**	**64.73*****	**72.56***	**65.79*****
COMPOSITE	68.36	49.45	62.41	46.24

Table 4: Labeled (LAS_g) and unlabeled attachment score (UAS_g) of head tokens of remnants for parsers trained and evaluated on the UD representation (ORPHAN) and the composite relations representation (COMPOSITE) on the development and test sets of the COMBINED treebank. Results that differ significantly are marked with * ($p < 0.05$) or *** ($p < 0.001$).

tistical significance of pairwise comparisons, we performed two-tailed approximate randomization tests (Noreen, 1989; Yeh, 2000) with an adapted version of the `sigf` package (Padó, 2006).

Results Table 3 shows the overall parsing results on the development and test sets of the two treebanks. There was no significant difference between the parser that was trained on the UD representation (ORPHAN) and the parser trained on the composite representation (COMPOSITE) when tested on the EWT data sets, which is not surprising considering that there is just one sentence with gapping each in the development and the test split. When evaluated on the GAPPING datasets, the ORPHAN parser performs significantly better ($p < 0.01$) in terms of labeled attachment score, which suggests that the parser trained on the COMPOSITE representation is indeed struggling with the greatly increased label space. This is further confirmed by the attachment scores of the head tokens of remnants (Table 4). The labeled attachment score of remnants is significantly higher for the ORPHAN parser than for the COMPOSITE parser. Further, the unlabeled attachment score on the test set is also higher for the ORPHAN parser, which suggests that the COMPOSITE parser is sometimes struggling with finding the right attachment for the multiple long-distance composite dependencies.

4.3 Recovery experiments

Our second set of experiments concerns the recovery of the elided material and the reattachment of the orphans. We conducted two experiments: an oracle experiment that used gold standard dependency trees and an end-to-end experiment that used the output of the parser as input. For all experiments, we used the COMBINED treebank.

Evaluation Here, we evaluated dependency graphs and therefore used the labeled and unlabeled precision and recall metrics. However, as our two procedures are only changing the attachment of orphans, we only computed these metrics for copy nodes and their dependents. Further, we excluded punctuation and coordinating conjunctions as their attachment is usually trivial and including them would inflate scores. Lastly, we computed the sentence-level accuracy for all sentences with gapping. For this metric, we considered a sentence to be correct if all copy nodes and their dependents of a sentence were attached to the correct head with the correct label.

Oracle results The top part of Table 5 shows the results for the oracle experiment. Both methods are able to reconstruct the elided material and the canonical clause structure from gold dependency trees with high accuracy. This was expected for the COMPOSITE procedure, which can make use of the composite relations in the dependency trees, but less so for the ORPHAN procedure which has to recover the structure and the types of relations. The two methods work equally well in terms of all metrics except for the sentence-level accuracy, which is significantly higher for the COMPOSITE procedure. This difference is caused by a difference in the types of mistakes. All errors of the COMPOSITE procedure are of a structural nature and stem from copying the wrong number of nodes while the dependency labels are always correct because they are part of the dependency tree. The majority of errors of the ORPHAN procedure stem from incorrect dependency labels, and these mistakes are scattered across more examples, which leads to the lower sentence-level accuracy.

End-to-end results The middle part of Table 5 shows the results for the end-to-end experiment. The performance of both methods is considerably lower than in the oracle experiment, which is pri-

		Development					Test				
		UP	UR	LP	LR	SAcc.	UP	UR	LP	LR	SAcc.
oracle	COMPOSITE	91.32	88.20	**91.32**	**88.20**	**91.14****	90.71	86.81	**90.71**	86.81	**86.08***
	ORPHAN	**94.08**	**93.79**	87.54	87.27	72.15	**92.02**	**92.02**	87.12	**87.12**	72.15
end-to-end	COMPOSITE	70.48	**49.69***	**65.64**	**46.27***	**31.65**	67.39	**47.55**	61.74	**43.56**	31.65
	ORPHAN	**71.73**	42.55	65.45	38.82	30.38	**78.92****	44.78	**68.11**	38.65	**34.18**
	K&K 2017	-	-	-	-	0.00	-	-	-	-	0.00

Table 5: Labeled and unlabeled precision and recall as well as sentence-level accuracy of the two gapping reconstructions methods and the K&K parser on the development and test set of the COMBINED treebank. Results that differ significantly from the other result within the same section are marked with * ($p < 0.05$) or ** ($p < 0.01$).

marily driven by the much lower recall. Both methods assume that the parser detects the existence of a gap and if the parser fails to do so, neither method attempts to reconstruct the elided material. In general, precision tends to be a bit higher for the ORPHAN procedure whereas recall tends to be a bit higher for the COMPOSITE method but overall and in terms of sentence-level accuracy both methods seem to perform equally well.

Error analysis For both methods, the primary issue is low recall, which is a result of parsing errors. When the parser correctly predicts the `orphan` relation, the main sources of error for the ORPHAN procedure are missing correspondents for remnants (e.g., *[for good]* has no correspondent in *They had left the company, many for good*) or that the types of argument of the remnant and its correspondent differ (e.g., in *She was convicted of selling unregistered securities in Florida and of unlawful phone calls in Ohio*, *[of selling unregistered securities]* is an adverbial clause whereas *[of unlawful phone calls]* is an oblique modifier).

Apart from the cases where the COMPOSITE procedure leads to an incorrect structure, the remaining errors are all caused by the parser predicting the wrong composite relation.

4.4 Comparison to Kummerfeld and Klein

Kummerfeld and Klein (henceforth K&K; 2017) recently proposed a one-endpoint-crossing graph parser that is able to directly parse to PTB-style trees with traces. They also briefly discuss gapping constructions and their parser tries to output the co-indexing that is used for gapping constructions in the PTB. The EWT and all the sentences that we took from the WSJ, Brown, and GENIA treebanks already come with constituency tree annotations, and we manually annotated the remaining sentences according to the PTB guidelines (Bies et al., 1995). This allowed us to train the K&K parser with exactly the same set of sentences that we used in our previous experiments. As this parser outputs constituency trees, we could not compute dependency graph metrics for this method. For the sentence-level accuracy, we considered an example to be correct if a) each argument in the gapped conjunct was the child of a single constituent node, which in return was the sibling of the full clause/verb phrase, and b) the co-indexing of each argument in the gapped conjunct was correct. For example, the following bracketing would be considered correct despite the incorrect internal structure of the first conjunct:

$[_S[_S[_{NP\text{-}1}$ Al $]$ likes $[_{NP\text{-}2}$ coffee $]]$ and $[_S[_{NP=1}$ Sue $][_{NP=2}$ tea $]]]$

The last row of Table 5 shows the results of the K&K parser. The parser failed to output the correct constituency structure or co-indexing for every single example in the development and test sets. The parser struggled in particular with outputting the correct co-indices: For 32.5% of the test sentences with gapping, the bracketing of the gapped clause was correct but one or more of the co-indices were missing from the output.

Overall these results suggest that our dependency-based approach is much more reliable at identifying gapping constructions than the parser by K&K, which, in their defense, was optimized to output traces for other phenomena. Our method is also faster and took only seconds to parse the test set, while the K&K parser took several hours.

5 Resolving gaps in other languages

One of the appeals of the ORPHAN procedure is that it can be easily applied to other languages even if there exist no annotated enhanced dependency graphs.[8] On the one hand, this is because

[8]There is no theoretical reason that would prevent one from using the COMPOSITE procedure for other languages

our method does not make use of lexical information, and on the other hand, this is because we developed our method on top of the UD annotation scheme, which has already been applied to many languages and for which many treebanks exist.

Currently, all treebanks but the English one lack copy nodes for gapping constructions and many of them incorrectly use the `orphan` relation (Droganova and Zeman, 2017) and therefore we could not evaluate our method on a large variety of languages. In order to demonstrate that our method can be applied to other languages, we therefore did a case study on the Swedish UD treebank. The Swedish UD treebank is an automatic conversion from a section of the Talbanken (Einarsson, 1976) with extensive manual corrections. While the treebank is overall of high quality, we noticed conversion errors that led to incorrect uses of the `orphan` relation in 11 of the 29 sentences with `orphan` relations, which we excluded from our evaluation. We applied our gapping resolution procedure without any modifications to the remaining 18 sentences. We used the Swedish word2vec embeddings that were prepared for the CoNLL 2017 Shared Task. Our method correctly predicts the insertion of 29 copy nodes and is able to predict the correct structure of the enhanced representation in all cases, including complex ones with elided verb clusters such as the example in Figure 2. It also predicts the correct dependency label for 108/110 relations, leading to a labeled precision and labeled recall of 98.18%, which are both higher than the English numbers despite the fact that we optimized our procedure for English. The main reason for the higher performance seems to be that many of the Swedish examples come from informational texts from public organizations, which are more likely to be written to be clear and unambiguous. Further, the Swedish data does not contain challenging examples from the linguistic literature.

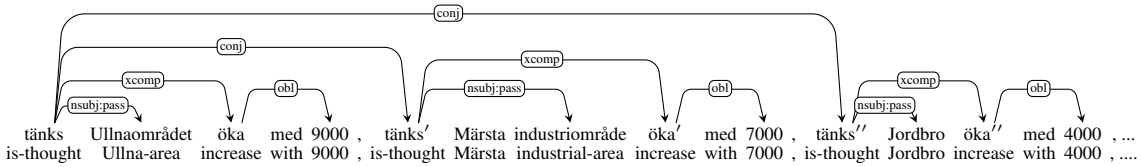
As Swedish is a Germanic language like English and thus shares many structural properties, we cannot conclude that our method is applicable to any language based on just this experiment. However, given that our method does not rely on language-specific structural patterns, we expect it to work well for a wide range of languages.

but given that UD treebanks are annotated with `orphan` relations, using the the COMPOSITE procedure would require additional manual annotations in practice.

6 Related work

Gapping constructions have been little studied in NLP, but several approaches (e.g., Dukes and Habash 2011; Simkó and Vincze 2017) parse to dependency trees with empty nodes. Seeker et al. (2012) compared three ways of parsing with empty heads: adding a transition that inserts empty nodes, using composite relation labels for nodes that depend on an elided node, and pre-inserting empties before parsing. These papers all focus on recovering nodes for elided function words such as auxiliaries; none of them attempt to recover and resolve the content word elisions of gapping. Ficler and Goldberg (2016) modified PTB annotations of argument-cluster coordinations (ACCs), i.e., gapping constructions with two post-verbal orphan phrases, which make up a subset of the gapping constructions in the PTB. While the modified annotation style leads to higher parsing accuracy of ACCs, it is specific to ACCs and does not generalize to other gapping constructions. Moreover, they did not reconstruct gapped ACC clauses. Traditional grammar-based chart parsers (Kay, 1980; Klein and Manning, 2001) did handle empty nodes and so could in principle provide a parse of gapping sentences though additional mechanisms would be needed for reconstruction. In practice, though, dealing with gapping in a grammar-based framework is not straightforward and can lead to a combinatorial explosion that slows down parsing in general, as has been noted for the English Resource Grammar (Flickinger, 2017, p.c.) and for an HPSG implementation for Norwegian (Haugereid, 2017). The grammar-based parser built with augmented transition networks (Woods, 1970) provided an extension in the form of the SYSCONJ operation (Woods, 1973) to parse some gapping constructions, but also this approach lacked explicit reconstruction mechanisms and provided only limited coverage.

There also exists a long line of work on post-processing surface-syntax constituency trees to recover traces in the PTB (Johnson, 2002; Levy and Manning, 2004; Campbell, 2004; Gabbard et al., 2006), pre-processing sentences such that they contain tokens for traces before parsing (Dienes and Dubey, 2003b), or directly parsing sentences to either PTB-style trees with empty elements or pre-processed trees that can be deterministically converted to PTB-style trees (Collins,



'The Ullna area is expected to grow by 9,000 (new workplaces), the Märsta industrial area by 7,000, Jordbro by 4,000, ...'

Figure 2: Dependency graph for part of the sentence `sv-ud-train-1102` as output by the ORPHAN procedure. The system correctly predicts the copy nodes for the matrix and the embedded verb, and correctly attaches the arguments to the copy nodes.

1997; Dienes and Dubey, 2003a; Schmid, 2006; Cai et al., 2011; Hayashi and Nagata, 2016; Kato and Matsubara, 2016; Kummerfeld and Klein, 2017). However, all of these works are primarily concerned with recovering traces for phenomena such as Wh-movement or control and raising constructions and, with the exception of Kummerfeld and Klein (2017), none of these works attempt to output the co-indexing that is used for analyzing gapping constructions. And again, none of these works try to reconstruct elided material.

Lastly, several methods have been proposed for resolving other forms of ellipsis, including VP ellipsis (Hardt, 1997; Nielsen, 2004; Lappin, 2005; McShane and Babkin, 2016) and sluicing (Anand and Hardt, 2016) but none of these methods consider gapping constructions.

7 Conclusion

We presented two methods to recover elided predicates in sentences with gapping. Our experiments suggest that both methods work equally well in a realistic end-to-end setting. While in general, recall is still low, the oracle experiments suggest that both methods can recover elided predicates from correct dependency trees, which suggests that as parsers become more and more accurate, the gap recovery accuracy should also increase.

We also demonstrated that our method can be used to automatically add the enhanced UD representation to UD treebanks in other languages than English. Apart from being useful in a parsing pipeline, we therefore also expect our method to be useful for building enhanced UD treebanks.

Reproducibility

All data, pre-trained models, system outputs as well as a package for running the enhancement procedure are available from `https://github.com/sebschu/naacl-gapping`.

Acknowledgments

We thank the anonymous reviewers for their thoughtful feedback. Also thanks to Vera Gribanova and Boris Harizanov for continuous feedback throughout this project, and to Matthew Lamm for help with annotating the data. This work was supported in part by gifts from Google, Inc. and IPSoft, Inc. The first author is also supported by a Goodan Family Graduate Fellowship.

References

Pranav Anand and Daniel Hardt. 2016. Antecedent selection for sluicing: Structure and content. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*. pages 1234–1243. `https://doi.org/10.18653/v1/D16-1131`.

Eduard Bejček, Eva Hajičová, Jan Hajič, Pavlína Jínová, Václava Kettnerová, Veronika Kolářová, Marie Mikulová, Jiří Mírovský, Anna Nedoluzhko, Jarmila Panevová, Lucie Poláková, Magda Ševčíková, Jan Štěpánek, and Šárka Zikánová. 2013. Prague Dependency Treebank 3.0. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University. `http://hdl.handle.net/11858/00-097C-0000-0023-1AAF-3`.

Ann Bies, Mark Ferguson, Karen Katz, Robert MacIntyre, Victoria Tredinnick, Grace Kim, Mary Ann Marcinkiewicz, and Britta Schasberger. 1995. Bracketing guidelines for Treebank II style Penn Treebank project. Technical report, University of Pennsylvania.

Shu Cai, David Chiang, and Yoav Goldberg. 2011. Language-independent parsing with empty elements. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL 2011)*. pages 212–216. `http://www.aclweb.org/anthology/P11-2037`.

Richard Campbell. 2004. Using linguistic principles to recover empty categories. In *Proceedings of the*

42nd Annual Meeting on Association for Computational Linguistics (ACL 2004). https://doi.org/10.3115/1218955.1219037.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL 1997)*. pages 16–23. https://doi.org/10.3115/976909.979620.

Elizabeth Coppock. 2001. Gapping: In defense of deletion. In Mary Andronis, Christopher Ball, Heidi Elston, and Sylvain Neuvel, editors, *Papers from the 37th Meeting of the Chicago Linguistic Society*. Chicago Linguistic Society, Chicago, pages 133–148.

Péter Dienes and Amit Dubey. 2003a. Antecedent recovery: Experiments with a trace tagger. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP 2003)*. pages 33–40. http://www.aclweb.org/anthology/W03-1005.

Péter Dienes and Amit Dubey. 2003b. Deep syntactic processing by combining shallow methods. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics (ACL 2003)*. pages 431–438. https://doi.org/10.3115/1075096.1075151.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*. pages 1–8. https://openreview.net/pdf?id=Hk95PK9le.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the CoNLL 2017 Shared Task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. pages 20–30. https://doi.org/10.18653/v1/K17-3002.

Kira Droganova and Daniel Zeman. 2017. Elliptic constructions: Spotting patterns in UD treebanks. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*. pages 48–57. http://www.aclweb.org/anthology/W17-0406.

Kais Dukes and Nizar Habash. 2011. One-step statistical parsing of hybrid dependency-constituency syntactic representations. In *Proceedings of the 12th International Conference on Parsing Technologies*. pages 92–103. http://www.aclweb.org/anthology/W11-2912.

Jan Einarsson. 1976. Talbankens skriftspråkskonkordans. Institutionen för nordiska språk, Lunds universitet.

Annahita Farudi. 2013. *Gapping in Farsi: A Crosslinguistic Investigation*. Ph.D. thesis, University of Massachusetts Amherst. http://scholarworks.umass.edu/dissertations/AAI3556244.

Jessica Ficler and Yoav Goldberg. 2016. Improved parsing for argument-clusters coordination. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*. pages 72–76. https://doi.org/10.18653/v1/P16-2012.

Ryan Gabbard, Mitchell Marcus, and Seth Kulick. 2006. Fully parsing the Penn Treebank. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL (NAACL 2006)*. pages 184–191. https://doi.org/10.3115/1220835.1220859.

Daniel Hardt. 1997. An empirical approach to VP ellipsis. *Computational Linguistics* 23(4):525–541. http://www.aclweb.org/anthology/J97-4002.

Petter Haugereid. 2017. An incremental approach to gapping and conjunction reduction. In *Proceedings of the 24th International Conference on Head-Driven Phrase Structure Grammar*. pages 179–198. http://cslipublications.stanford.edu/HPSG/2017/hpsg2017-haugereid.pdf.

Katsuhiko Hayashi and Masaaki Nagata. 2016. Empty element recovery by spinal parser operations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*. pages 95–100. https://doi.org/10.18653/v1/P16-2016.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics* 33(3):355–396. http://www.aclweb.org/anthology/J07-3004.

Ray S. Jackendoff. 1971. Gapping and related rules. *Linguistic Inquiry* 2(1):21–35.

Kyle Johnson. 2014. Gapping. Unpublished manuscript, University of Massachusetts at Amherst. http://people.umass.edu/kbj/homepage/Content/gapping.pdf.

Mark Johnson. 2002. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*. pages 136–143. https://doi.org/10.3115/1073083.1073107.

Yoshihide Kato and Shigeki Matsubara. 2016. Transition-based left-corner parsing for identifying PTB-style nonlocal dependencies. In *Proceedings of the 54th Annual Meeting of*

the Association for Computational Linguistics (ACL 2016). pages 930–940. https://doi.org/10.18653/v1/P16-1088.

Martin Kay. 1980. Algorithm schemata and data structures in syntactic processing. Technical report, Xerox PARC.

Dan Klein and Christopher Manning. 2001. Parsing with treebank grammars: empirical bounds, theoretical models, and the structure of the Penn Treebank. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics (ACL 2001)*. pages 338–345. https://doi.org/10.3115/1073012.1073056.

Yusuke Kubota and Robert Levine. 2016. Gapping as hypothetical reasoning. *Natural Language and Linguistic Theory* 34(1):107–156. https://doi.org/10.1007/s11049-015-9298-4.

Jonathan K. Kummerfeld and Dan Klein. 2017. Parsing with traces: An $O(n^4)$ algorithm and a structural representation. *Transactions of the Association for Computational Linguistics* 5:441–454. https://www.transacl.org/ojs/index.php/tacl/article/view/1170.

Shalom Lappin. 2005. A sequenced model of anaphora and ellipsis resolution. In António Branco, Tony McEnery, and Ruslan Mitkov, editors, *Anaphora Processing: Linguistic, Cognitive and Computational Modelling*, John Benjamins Publishing, Amsterdam, pages 3–16. https://doi.org/10.1075/cilt.263.03lap.

Roger Levy and Christopher D. Manning. 2004. Deep dependencies from context-free statistical parsers. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (ACL 2004)*. pages 327–334. https://doi.org/10.3115/1218955.1218997.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* 19(2):313–330. http://www.aclweb.org/anthology/J93-2004.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT-EMNLP 2005)*. pages 523–530. https://doi.org/10.3115/1220575.1220641.

Marjorie McShane and Petr Babkin. 2016. Detection and resolution of verb phrase ellipsis. *Linguistic Issues in Language Technology (LiLT)* 13(1):1–34.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26 (NIPS 2013)*. pages 3111–3119.

Saul B. Needleman and Christian D. Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology* 48(3):443–453. https://doi.org/10.1016/0022-2836(70)90057-4.

Leif Arda Nielsen. 2004. Verb phrase ellipsis detection using automatically parsed text. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*. 1, pages 1093–1099. https://doi.org/10.3115/1220355.1220512.

Joakim Nivre, Željko Agić, Lars Ahrenberg, Lene Antonsen, Maria Jesus Aranzabe, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Eckhard Bick, Victoria Bobicev, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Aljoscha Burchardt, Marie Candito, Gauthier Caron, Gülşen Cebiroğlu Eryiğit, Giuseppe G. A. Celano, Savas Cetin, Fabricio Chalub, Jinho Choi, Silvie Cinková, Çağrı Çöltekin, Miriam Connor, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Ali Elkahky, Tomaž Erjavec, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Nizar Habash, Jan Hajič, Jan Hajič jr., Linh Hà Mỹ, Kim Harris, Dag Haug, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Radu Ion, Elena Irimia, Tomáš Jelínek, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Hiroshi Kanayama, Jenna Kanerva, Tolga Kayadelen, Václava Kettnerová, Jesse Kirchner, Natalia Kotsyba, Simon Krek, Veronika Laippala, Lorenzo Lambertino, Tatiana Lando, John Lee, Phương Lê Hồng, Alessandro Lenci, Saran Lertpradit, Herman Leung, Cheuk Ying Li, Josie Li, Keying Li, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher D. Manning, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Gustavo Mendonça, Niko Miekka, Anna Missilä, Cătălin Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Shinsuke Mori, Bohdan Moskalevskyi, Kadri Muischnek, Kaili Müürisep, Pinkey Nainwani, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Lương Nguyễn Thị, Huyền Nguyễn Thị Minh, Vitaly Nikolaev, Hanna Nurmi, Stina Ojala, Petya Osenova, Robert Östling, Lilja Øvrelid, Elena Pascual,

Marco Passarotti, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Emily Pitler, Barbara Plank, Martin Popel, Lauma Pretkalniņa, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Vinit Ravishankar, Livy Real, Siva Reddy, Georg Rehm, Larissa Rinaldi, Laura Rituma, Mykhailo Romanenko, Rudolf Rosa, Davide Rovati, Benoît Sagot, Shadi Saleh, Tanja Samardžić, Manuela Sanguinetti, Baiba Saulīte, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Antonio Stella, Milan Straka, Jana Strnadová, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Takaaki Tanaka, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Sowmya Vajjala, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Eric Villemonte de la Clergerie, Veronika Vincze, Lars Wallin, Jonathan North Washington, Mats Wirén, Tak-sum Wong, Zhuoran Yu, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, and Hanzhi Zhu. 2017. Universal Dependencies 2.1. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University. http://hdl.handle.net/11234/1-2515.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. pages 1659–1666. http://www.lrec-conf.org/proceedings/lrec2016/pdf/348_Paper.pdf.

Eric W. Noreen. 1989. *Computer-Intensive Methods for Testing Hypotheses*. John Wiley & Sons, New York, NY.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinkova, Dan Flickinger, Jan Hajic, and Zdenka Uresova. 2015. SemEval 2015 Task 18: Broad-coverage semantic dependency parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. pages 915–926. https://doi.org/10.18653/v1/S15-2153.

Tomoko Ohta, Yuka Tateisi, and Jin-Dong Kim. 2002. The GENIA corpus: An annotated research abstract corpus in molecular biology domain. *Proceedings of the Second International Conference on Human Language Technology Research (HLT 2002)* pages 82–86. https://doi.org/10.3115/1289189.1289260.

Timothy Osborne. 2006. Gapping vs. non-gapping coordination. *Linguistische Berichte* 207:307–337.

Sebastian Padó. 2006. *User's guide to sigf: Significance testing by approximate randomisation*. https://nlpado.de/~sebastian/software/sigf.shtml.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*. pages 1532–1543. https://doi.org/10.3115/v1/D14-1162.

John Robert Ross. 1970. Gapping and the order of constituents. In Manfred Bierwisch and Karl Erich Heidolph, editors, *Progress in Linguistics*, De Gruyter, The Hague, pages 249–259.

Helmut Schmid. 2006. Trace prediction and recovery with unlexicalized PCFGs and slash features. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL (ACL 2006)*. pages 177–184. https://doi.org/10.3115/1220175.1220198.

Sebastian Schuster, Matthew Lamm, and Christopher D. Manning. 2017. Gapping constructions in Universal Dependencies v2. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*. pages 123–132. http://www.aclweb.org/anthology/W17-0416.

Sebastian Schuster and Christopher D. Manning. 2016. Enhanced English Universal Dependencies: An improved representation for natural language understanding tasks. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. pages 2371–2378. http://www.lrec-conf.org/proceedings/lrec2016/pdf/779_Paper.pdf.

Wolfgang Seeker, Richárd Farkas, Bernd Bohnet, Helmut Schmid, and Jonas Kuhn. 2012. Data-driven dependency parsing with empty heads. In *Proceedings of COLING 2012*. pages 1081–1090. http://www.aclweb.org/anthology/C12-2105.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Christopher D Manning. 2014. A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2014)*. pages 2897–2904. http://www.lrec-conf.org/proceedings/lrec2014/pdf/1089_Paper.pdf.

Katalin Ilona Simkó and Veronika Vincze. 2017. Hungarian copula constructions in dependency syntax and parsing. In *Proceedings of the Fourth International Conference on Dependency Linguistics (Depling 2017)*. pages 240–247. http://www.aclweb.org/anthology/W17-6527.

Mark Steedman. 1990. Gapping as constituent coordination. *Linguistics and Philosophy* 13(2):207–263. https://doi.org/10.1007/BF00630734.

Maziar Toosarvandani. 2016. Embedding the antecedent in gapping: Low coordination and the role of parallelism. *Linguistic Inquiry* 47(2):381–390. https://doi.org/10.1162/LING_a_00216.

William A. Woods. 1970. Transition network grammars for natural language analysis. *Communications of the ACM* 13(10):591–606.

William A. Woods. 1973. An experimental parsing system for transition network grammars. In Randall Rustin, editor, *Natural Language Processing*, Algorithmics Press, New York, pages 111–154.

G. Vanden Wyngaerd. 2007. Gapping constituents. Unpublished manuscript, FWO/K.U. Brussel. https://lirias.kuleuven.be/bitstream/123456789/408979/1/09HRPL&L02.pdf.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the The 18th International Conference on Computational Linguistics (COLING 2000)*. pages 947–953. http://www.aclweb.org/anthology/C00-2137.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajič, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gökrmak, Anna Nedoluzhko, Silvie Cinková, Jan Hajič jr., Jaroslava Hlaváčová, Václava Kettnerová, Zdenka Urešová, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria de Paiva, Kira Droganova, Hěctor Martínez Alonso, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadova, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonça, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 2017. CoNLL 2017 Shared Task: Multilingual parsing from raw text to Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. pages 1–19. https://doi.org/10.18653/v1/K17-3001.

A Structured Syntax-Semantics Interface for English-AMR Alignment

Ida Szubert **Adam Lopez**
School of Informatics
University of Edinburgh
Edinburgh, Scotland, UK
{k.i.szubert@sms, alopez@inf}.ed.ac.uk

Nathan Schneider
Linguistics and Computer Science
Georgetown University
Washington, DC, USA
nathan.schneider@georgetown.edu

Abstract

Abstract Meaning Representation (AMR) annotations are often assumed to closely mirror dependency syntax, but AMR explicitly does not require this, and the assumption has never been tested. To test it, we devise an expressive framework to align AMR graphs to dependency graphs, which we use to annotate 200 AMRs. Our annotation explains how 97% of AMR edges are evoked by words or syntax. Previously existing AMR alignment frameworks did not allow for mapping AMR onto syntax, and as a consequence they explained at most 23%. While we find that there are indeed many cases where AMR annotations closely mirror syntax, there are also pervasive differences. We use our annotations to test a baseline AMR-to-syntax aligner, finding that this task is more difficult than AMR-to-string alignment; and to pinpoint errors in an AMR parser. We make our data and code freely available for further research on AMR parsing and generation, and the relationship of AMR to syntax.

1 Introduction

Abstract Meaning Representation (AMR; Banarescu et al., 2013) is a popular framework for annotating whole sentence meaning. An AMR annotation is a directed, usually acyclic graph in which nodes represent entities and events, and edges represent relations between them, as on the right in figure 1.[1]

AMR annotations include no explicit mapping between elements of an AMR and the corresponding elements of the sentence that evoke them, and this presents a challenge to developers of machine learning systems that parse sentences to AMR or generate sentences from AMR, since they must first infer this mapping in the training data (e.g. Flanigan et al., 2014; Wang et al., 2015; Artzi et al., 2015; Flanigan et al., 2016; Pourdamghani et al., 2016; Misra and Artzi, 2016; Damonte et al., 2017; Peng et al., 2017, inter alia).[2]

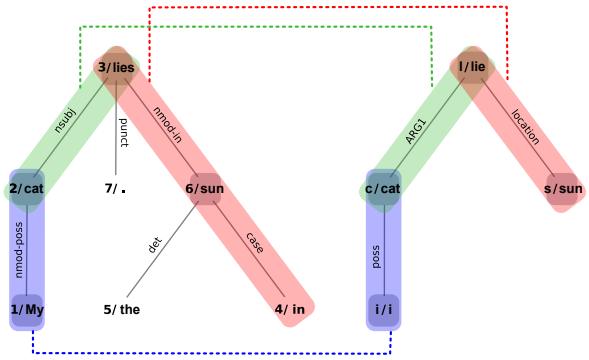


Figure 1: "My cat lies in the sun." An alignment between the dependency parse (*left*) and AMR (*right*). Nodes participating in lexical alignments are marked with boxes, but the links between them are not displayed. Structural alignments are colour-coded and linked by dotted lines. Sense numbers for concepts that are PropBank frames are omitted for brevity.

This AMR alignment problem was first formalized by Flanigan et al. (2014), who mapped AMR nodes or connected subgraphs to words or sequences of words under the assumption of a one-to-one mapping—we call this **JAMR alignment**. Pourdamghani et al. (2014) then re-formalized it so that any AMR node or edge can map to any word without a one-to-one assumption—we call this **ISI alignment**. In ISI alignments, edges often align to syntactic function words: for example, `:location` aligns to *in* in figure 1. So edge alignments allow ISI to explain more of the AMR structure than JAMR, but in a limited way: only 23% of AMR edges are aligned in the ISI corpus. This may be be-

[1] For clarity of presentation, we have constructed the sentences and AMRs shown in figures—except for figure 3, which is a simplified version of a sentence in the corpus.

[2] Some recent neural AMR sytems require minimal or no explicit alignments (Konstas et al., 2017; van Noord and Bos, 2017). But they implicitly learn them in the form of soft attention, and we believe that a clearer understanding of alignment will benefit modeling and error analysis even in these systems.

Proceedings of NAACL-HLT 2018, pages 1169–1180
New Orleans, Louisiana, June 1 - 6, 2018.

cause edges are often evoked by syntactic structure rather than words: for instance, the :ARG1 edge in figure 1 is evoked by the fact that *cat* is the subject of *lies* and not by any particular word.

Although it seems sensible to assume that all of the nodes and edges of an AMR are evoked by the words and syntax of a sentence, the existing alignment schemes do not allow for expressing that relationship. We therefore propose a framework expressive enough to align AMR to syntax (§2) and use it to align a corpus of 200 AMRs to dependency parses. We analyse our corpus and show that the addition of syntactic alignments allows us account for 97% of the AMR content.

Syntactic-semantic mappings are often assumed by AMR parsing models (e.g. Wang et al., 2015; Artzi et al., 2015; Damonte et al., 2017), which is understandable since these mappings are well-studied in linguistic theory. But AMR explicitly avoids theoretical commitment to a syntax-semantics mapping: Banarescu et al. (2013) state that "AMR is agnostic about how we might want to derive meanings from strings." If we are going to build such an assumption into our models, we should test it empirically, which we can do by analysing our corpus. We observe some pervasive structural differences between AMR and dependency syntax (§3), despite the fact that a majority of AMR edges map easily onto dependency edges.

Since syntactic alignment can largely explain AMRs, we also develop a baseline rule-based aligner for it, and show that this new task is much more difficult than lexical alignment (§4). We also show how our data can be used to analyze errors made by an AMR parser (§5). We make our annotated data and aligner freely available for further research.[3]

2 Aligning AMR to dependency syntax

Our syntactic representation is dependency grammar, which represents the sentence as a rooted, directed graph where nodes are words and edges are grammatical relations between them (Kruijff, 2006). We use Universal Dependencies (UD), a cross-lingual dependency annotation scheme, as implemented in Stanford CoreNLP (Manning et al., 2014). Within the UD framework, we use *enhanced* dependencies (Schuster and Manning, 2016), in which dependents can have more than one head, resulting in **dependency graphs** (DGs).[4]

Our alignment guidelines generalize ideas present in the existing frameworks. We want to allow many-to-many alignments, which we motivate by the observation that some phenomena cause an AMR graph to have one structure expressing the same information as multiple DG structures, and vice versa. For instance, in figure 2 the AMR subgraph representing Cruella de Vil aligns to two subgraphs in the dependency graph because of pronominal coreference. In the other direction, in figure 3 the *capabilities* node aligns to both capable nodes in the AMR, which is a result of the AMR treating conjoined adjectival modifiers as a case of ellipsis. The alignments we propose hold between subgraphs of any size. By aligning subgraphs we gain expressiveness needed to point out correspondences between semantic and syntactic structure. If AMR and DG were very similar in how they represent information, such correspondences would probably hold between subgraphs consisting of a single edge, as in figure 1 $cat \xrightarrow{\textbf{\textit{nmod:poss}}} my \sim \texttt{cat} \xrightarrow{\texttt{poss}} \texttt{I}$. However, AMR by design abstracts away from syntax and it should not be assumed that all mappings will be so clean. For example, the same figure has $lies \xrightarrow{\textbf{\textit{nmod-in}}} sun \xrightarrow{\textbf{\textit{case}}} in \sim \texttt{lies} \xrightarrow{\texttt{location}} \texttt{sun}$. Moreover, AMR represents the meaning of particular words or phrases with elaborate structures, the result of which might be that the same information is expressed by a single word and a complex AMR subgraph, as in figure 3 where AMR represents *general* as $\texttt{person} \xrightarrow{\texttt{ARG0-of}} \texttt{have-org-role} \xrightarrow{\texttt{ARG2}} \texttt{general}$.

2.1 Overview

An alignment is a link between subgraphs in an AMR and a DG which represent equivalent information. Given a sentence's DG and AMR we define an **alignment** as a mapping between an AMR subgraph and a DG subgraph. **Lexical alignments** (§2.2) hold between pairs of nodes, and nodes from either graph may participate in multiple lexical alignments. **Structural alignments** (§2.3) hold between pairs of connected subgraphs where at least one of the subgraphs contains an edge.

[3] https://github.com/ida-szubert/amr_ud

[4] We chose UD because it emphasises shallow and semantically motivated annotation, by the virtue of which it can be expected to align relatively straightforwardly to a semantic annotation such as AMR. Aligning AMR with different versions of dependency grammar (e.g. Prague) or different syntactic frameworks (e.g. CCG, TAG) would be an interesting extension of our work.

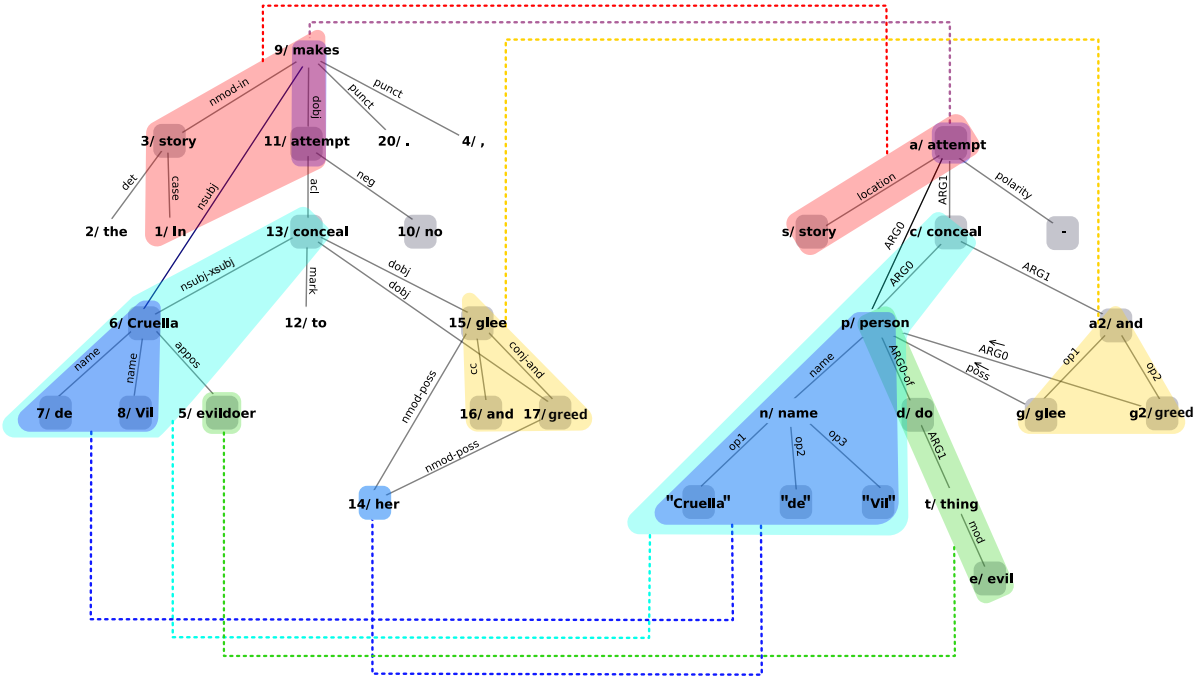


Figure 2: "In the story, evildoer Cruella de Vil makes no attempt to conceal her glee and greed." For legibility in this and following figures only a subset of the structural alignments are shown.

In the following two sections we discuss the types of alignments that our framework allows. More detailed guidelines regarding how to align particular linguistic constructions can be found in appendix A.

2.2 Lexical alignments

A lexical alignment should hold between a word and an AMR concept if the latter is judged to express the lexical meaning of the former. Node labels usually reflect their lexically aligned word or its lemma, including derivational morphology (e.g. *thirsty* ~ `thirst-01`). Thus, **string similarity** is a useful heuristic for lexical alignment.[5]

Most AMR nodes align lexically to a single word. Cases of one-to-many alignments include **coreference**, when an entity is mentioned multiple times in the sentence, and **multiword expressions** such as a verb-particle constructions (*pay off* ~ `pay-off-02`) and fixed grammatical expressions (*instead of* ~ `instead-of-91`). Occasionally an AMR node does not lexically align to any DG node. This is true for constants indicating sentence mood such as `imperative`, implicit uses of *and* to group list items, inferred concept nodes such as **entity types**, `name` in named entities, and `-91` frames like `have-org-role-91`.

Most words are lexically aligned to a single AMR node, if they are aligned at all. A word may align to multiple AMR nodes if it is **duplicated** in the AMR due to ellipsis or distributive coordination (*capabilities* aligns to `c2 / capable` and `c3 / capable` in figure 3), or if it is **morphologically decomposed** in the AMR (*evildoer* aligns to `evil` and `do-02` in figure 2). Many words are not lexically aligned to any AMR node, including **punctuation** tokens, **articles**, **copulas**, nonmodal **auxiliaries**, expletive subjects, infinitival *to*, complementizer *that*, and relative pronouns.

2.3 Structural alignments

Structural alignments primarily reflect compositional **grammatical constructions**, be they syntactic or morphological. Note that the structural alignments build upon the lexical ones. Structural alignments hold between two subgraphs, at least one of which is larger than a single node. If a subgraph includes any edges, it automatically includes nodes adjacent to those edges. Structural alignments need not be disjoint: an edge can appear in two or more distinct alignments. Nodes and edges in both AMR and DG may be unaligned.

2.3.1 Constraints on structural alignments

The ability to align subgraphs to subgraphs gives considerable flexibility in how the annotation task

[5] Exceptions include: pronouns with noun antecedents in the sentence; the `-` indicating negative polarity, which lexically aligns to *no*, *not*, and negative prefixes; modal auxiliaries, e.g., *can* ~ `possible`; normalized dates and values such as *February* ~ `2` in a `date-entity`; and `amr-unknown`, which aligns to wh-words.

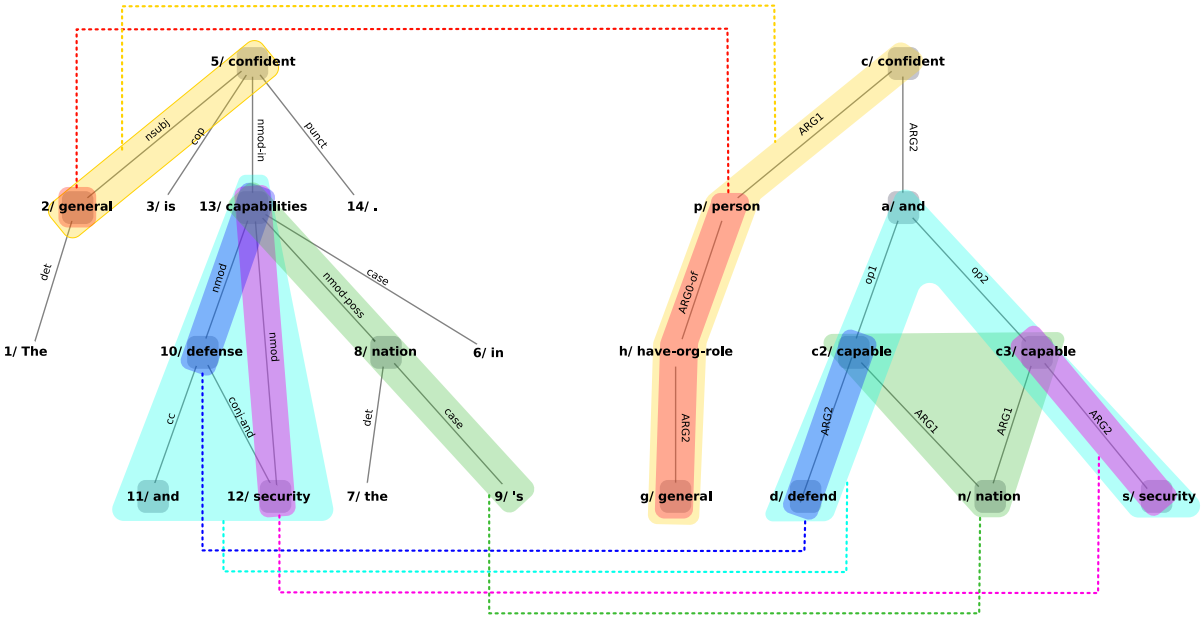


Figure 3: "The general is confident in the nation's defense and security capabilities."

can be interpreted. We establish the following principles to guide the specification of alignment:

Connectedness Principle. In an alignment $d \sim a$, d must be a connected subgraph of the DG, and a must be a connected subgraph of the AMR.

Minimality Principle. If two alignments, $d \sim a$ and $d' \sim a'$, have no dependency or AMR edges in common, then their union $d \cup d' \sim a \cup a'$ is redundant, even if it is valid. Individual alignments should be as small as possible; we believe compositionality is best captured by keeping structures minimal. Therefore, in figure 1 there is no alignment between subgraphs spanning *My*, *cat*, *lies* and `i`, `cat`, `lie`. Such subgraphs do express equivalent information, but the alignment between them decomposes neatly into smaller alignments and we record only those.

Subsumption Principle. This principle expresses the fact that our alignments are hierarchical. Structural alignments need to be consistent with lexical alignments: for subgraph a to be aligned to subgraph d, all nodes lexically aligned to nodes in a must be included in d, and vice versa. Moreover, structural alignments need to be consistent with other structural alignments. A structural alignment $d \sim a$ is valid only if, for every connected AMR subgraph $a_< \subset a$ which is aligned to a DG subgraph, $d' \sim a_<$, we also have that d' is a subgraph of d—and vice versa for every $d_< \subset d$.

Further, if a contains a node n which is not lexically aligned but which is part of a structurally aligned subgraph a' such that $d' \sim a'$, it needs to be the case that $a' \subset a \wedge d' \subset d$ or $a' \supset a \wedge d' \supset d$. (And vice versa for nodes in d.) For example, $\textit{conceal} \xrightarrow{\textbf{\textit{nsubj-xsubj}}} \textit{Cruella} \sim \texttt{conceal} \xrightarrow{\texttt{ARG0}} \texttt{person} \xrightarrow{\texttt{name}} \texttt{name} \xrightarrow{\texttt{op1}} \texttt{Cruella}$ is not a valid alignment, because the AMR side contains nodes `person` and `name`, which are not lexically aligned but which are both parts of a structural alignment marked in blue.

Coordination Principle. If an alignment contains a dependency edge between two conjuncts, or between a conjunct and a coordinating conjunction, then it must also include all conjuncts and the conjunction. This preserves the integrity of coordinate structures in alignments. For example, in figure 2 there is no alignment $\textit{glee} \xrightarrow{\textbf{\textit{cc}}} \textit{and} \sim \texttt{and} \xrightarrow{\texttt{op1}} \texttt{glee}$; only the larger structure which includes the `greed` nodes is aligned.

Named Entity Principle. Any structural alignment containing an AMR `name` node or any of the strings under it must contain the full subgraph rooted in the `name` plus the node above it specifying the entity type. This means that for example, in figure 2 there is no alignment $\textit{conceal} \xrightarrow{\textbf{\textit{nsubj-xsubj}}} \textit{Cruella} \sim \texttt{conceal} \xrightarrow{\texttt{ARG0}} \texttt{person} \xrightarrow{\texttt{name}} \texttt{name} \xrightarrow{\texttt{op1}} \texttt{"Cruella"}$. Such an alignment would also be stopped by the Subsumption Principle provided that the blue alignment of the whole name was present. The Named Entity Principle is superfluous, but is provided to explicitly describe the treatment of such constructions.

2.3.2 Typology of structural alignments

The smallest structure which can participate in a structural alignment is a single node, provided that it is aligned to a subgraph containing at least one edge. A DG node may align to an AMR subgraph if the word is morphologically decomposed or otherwise analyzed in the AMR (e.g. in figure 2, *evil-doer* ~ person$\xrightarrow{\text{ARG0-of}}$do-02$\xrightarrow{\text{ARG1}}$thing$\xrightarrow{\text{mod}}$evil). Examples of DG structures whose meaning is expressed in a single AMR node include light verb constructions, phrasal verbs, and various other multiword expressions (e.g. in figure 2, *makes*$\xrightarrow{\textbf{\textit{dobj}}}$*attempt* ~ attempt-01).

Conceptually the simplest case of structural alignment is one edge to one edge, as in the blue and green alignments in figure 1. For such an alignment to be possible, two requirements must be satisfied: nodes which are endpoints of those edges need to be aligned one-to-one; and the AMR relation and the syntactic dependency must map cleanly in terms of the relationship they express.

A one edge to multiple edges alignment arises when either of those requirements is not met. To see what happens in absence of one-to-one endpoint alignments let's look at the relation between *confident* and *general* in figure 3. The DG *general* node is aligned to an AMR subgraph: *general* ~ person$\xrightarrow{\text{ARG0-of}}$have-org-role$\xrightarrow{\text{ARG2}}$general. All alignments which involve the *general* node on the DG side need to include its aligned subgraph on the AMR side. It necessarily follows that the AMR subgraphs in those alignments will contain more edges that the DG ones; in this case the yellow subgraph in DG has 1 edge, and in AMR 3 edges. As for the second requirement, it is possible for one graph to use multiple edges to express a relationship when the other graph needs only one. This is the case for *lie*$\xrightarrow{\textbf{\textit{nmod-in}}}$*sun*$\xrightarrow{\textbf{\textit{case}}}$*in* ~ lie$\xrightarrow{\text{location}}$sun in figure 1. An example which combines both the node- and edge-related issues is marked in red in figure 2.

Finally, we also allow for many edges to many edges alignments. This may seem counterintuitive considering the assumption that we want to capture mappings between relations expressed in DG and AMR, and that we want to align minimal subgraphs. There are cases where an alignment is actually capturing a single relation, but we need to treat a subgraph as an endpoint of the edge both in DG and AMR. For instance, consider in figure 2 the relationship that holds between Cruella de Vil and concealing, expressed syntactically as an ***nsubj-xsubj*** edge and semantically as an ARG0 edge. One of the entities involved in that relationship, Cruella, is represented by a 2-edge DG subgraph and a 4-edge AMR subgraph. Consequently, the alignment covering the DG and AMR edges that relate Cruella to concealing must link subgraphs consisting respectively of 3 and 5 edges. A more difficult case of many edges to many edges alignment arises when relationships between nodes are expressed so differently in the DG and AMR that given an edge in one graph it is not possible to find in the other graph a subgraph that would convey the same information without also including some other information. Coordination has this property: e.g. in figure 2 the ***conj-and*** dependency between *glee* and *greed* has no counterpart in the AMR. There is no edge between AMR nodes aligned to those words, and the smallest AMR subgraph which contains them also contains and, which is itself lexically aligned. We cannot align *glee*$\xrightarrow{\textbf{\textit{conj-and}}}$*greed* ~ glee$\xleftarrow{\text{op1}}and\xrightarrow{\text{op2}}$greed because of the rule that all lexically aligned nodes in one subgraph must be aligned to nodes in the other subgraph. Therefore we need to extend the DG side to *and*$\xleftarrow{\textbf{\textit{cc}}}$*glee*$\xrightarrow{\textbf{\textit{conj-and}}}$*greed*.

3 Manually aligned corpus

We annotated a corpus of 200 AMR-sentence pairs (3813 aligned structures) using the guidelines of §2 and appendix A.[6]

Data selection. To create the corpus we drew a total of 200 AMR-sentence pairs: 135 from the training split of the AMR Annotation Release 1.0 (Knight et al., 2014), 55 from the training split of *The Little Prince* Corpus v1.6,[7] and 10 sentences from the Adam part of the CHILDES Brown corpus (Brown, 1973), for which AMRs were produced by an experienced annotator. Seventy items were selected to illustrate particular linguistic phenomena.[8] The remaining 130 were selected at random.

[6] We followed the precedent of previous AMR-to-sentence alignment corpora (see §4.2) in including 200 sentences in our gold standard, though ours was a different sample.

[7] https://amr.isi.edu/download/amr-bank-struct-v1.6.txt

[8] Namely: relative clauses, reflexive and non-reflexive pronominal anaphora, subject and object control, raising, exceptional case marking, coordination, wh-questions, do-support questions, ellipsis, expletives, modal verbs, light verbs, comparison constructions, and quantification.

Preprocessing. Dependency parses were obtained using Stanford CoreNLP neural network parser[9] (Chen and Manning, 2014) and manually corrected. The final parses conform to the enhanced UD guidelines,[10] except they lack enhancements for ellipsis.

Inter-annotator agreement. The corpus was created by one annotator. To assess inter-annotator agreement, a second annotator deeply familiar with UD and AMR annotated a random sample of sentences accounting for 10% of alignments in the corpus. The overall inter-annotator F_1-score was 88%, with 96% agreement on lexical alignments and 80% on structural alignments. We take this as an indication that our richly structured alignment framework as laid out in §2 is reasonably well-defined for annotators.

3.1 Coverage

To assess our attempt to explain as much of the AMR as possible, we computed the proportion of AMR nodes and edges that participate in at least one alignment. Overall, 99.3% of nodes and 97.2% of edges in AMRs are aligned. We found that 81.5% of AMR graphs have full coverage, 18.5% have at least one unaligned edge, and 7.5% have one unaligned node (none had more than one; all unaligned nodes express mood or discourse-related information: `interrogative`, `and`, and `say`). We conclude that nearly all information in an AMR is evoked by lexical items or syntactic structure.

We expected coverage of DG to be lower because punctuation and many function words are unaligned in our guidelines (§2.2). Indeed, only 71.4% of words and 65.2% of dependency edges are aligned.

3.2 Syntactic-semantic similarity

The similarity of AMR to syntax in examples like figure 1 invites the assumption of a close mapping, which often seems to be made in AMR parsers (Wang et al., 2015; Artzi et al., 2015; Misra and Artzi, 2016; Damonte et al., 2017) and aligners (Chu and Kurohashi, 2016; Chen and Palmer, 2017).[11] Such an attitude reflects decades of work in the syntax-semantics interface (Partee, 2014) and the utility of dependency syntax for other forms of semantics (e.g., Oepen et al., 2014; Reddy et al., 2016; Stanovsky et al., 2016; White et al., 2016; Zhang et al., 2017; Hershcovich et al., 2017). However, this assumption has not been empirically tested, and as Bender et al. (2015) observe, it is an assumption not guaranteed by the AMR annotation style. Having aligned a corpus of AMR-DG pairs, we are in a position to provide empirical evidence.

simple configurations			*complex configurations*		
max config.	**# sents**	**avg. words**	**max config.**	**# sents**	**avg. words**
1:1	18	8.7	2:2	21	12.9
1:2	16	13.1	2:3	14	16.0
3:1	12	13.4	3:2	13	16.8
2:0	6	5.8	3:4	12	20.3
1:3	5	13.2	3:3	10	19.1
other	9	15.2	other	64	20.9
total:	66	11.6		134	18.0

Table 1: Number of sentences whose highest alignment configurations is max config.

Are AMRs and dependency graphs structurally similar? We approach the question by analyzing the sizes of subgraphs used to align the two representations of the sentence.

We define the size of a subgraph as the number of edges it contains. If a structure consists of a single node, we say its size is 0. The **configuration** of an alignment is then the pair of sizes for its AMR and DG sides; for example, an alignment with 1 AMR edge and 2 DG edges has configuration 1:2. We call an alignment configuration *simple* if at least one of the subgraphs is a single edge, indicating that there is a single relation which the alignment captures. *Complex* configurations cover multiple relations. By principle of minimality we infer that some structural difference between the graphs prevented those relations from aligning individually.

One measure of similarity between AMR and DG graphs is the configuration of the most complex subgraph alignment between them. Configuration a:b is higher than c:d if $a+b>c+d$. However, all configurations involving 0 are lower than those which do not. A maximum of 1:1 means the graphs have only node-to-node, node-to-edge, and edge-to-edge alignments, rendering the graphs isomorphic (ignoring edge directions and unaligned nodes). In

[9] The corpus is annotated with UD v1; a release of the dataset converted to UD v2 is planned for the future. We used the pretrained dependency parsing model provided in CoreNLP with `depparse.extradependencies` set to `MAXIMAL`, and used collapsed CCprocessed dependencies.

[10] `http://universaldependencies.org/u/overview/enhanced-syntax.html`

[11] In particular, Chen and Palmer (2017) align dependency paths to AMR edges. However, their evaluation only considers node-to-node alignment, and their code and data are not available for comparison at the time of this writing.

named entities		coordination		semantic decomposition		quantities & dates		other		overall	
2:0	112	2:2	30	1:0	32	2:1	15	0:0	1946	0:0	1946
3:1	44	3:4	14	2:0	14	3:0	5	1:1	1002	1:1	1046
4:2	7	3:3	13	2:1	11	1:0	4	1:2	220	1:2	244
1:1	6	4:3	5	4:1	6	3:2	3	1:0	42	2:0	127
5:2	4	3:2	5	3:1	6	8:2	1	2:2	42	2:2	83
other	20	other	50	other	15	other	0	other	13	other	361
total:	193		117		84		28		3385		**3807**

Table 2: Frequency of alignment configurations for named entities, coordination, semantically decomposed words, quantities and dates, and other phenomena.

general, if the maximum alignment configuration is a simple one, the graphs could be made isomorphic by collapsing the larger side of the alignment (e.g., in figure 2, the AMR side of the alignment *evil-doer* ~ person$\xrightarrow{\text{ARG0-of}}do\xrightarrow{\text{ARG1}}$thing$\xrightarrow{\text{mod}}$evil could be collapsed into a node).

In contrast, complex configurations imply serious structural dissimilarity, as in figure 3, where the cyan alignment has configuration 4:4.

The numbers in table 1 show that ≈33% of the sentences are *simple*.

Table 2 provides a detailed breakdown of alignment configurations in the corpus. Phenomena which often trigger complex configurations include coordination, named entities, semantically decomposed words, attachment of negation, and preposition-based concepts encoding location, time, and quantity.[12]

We observe, comparing tables 1 and 2, that while simple configurations are most frequent in the corpus, the majority of sentences have at least one alignment which is complex. It should not be assumed that AMR and DG representations of a sentence are, or could trivially be made to be, isomorphic. It is worth noting that our analysis suggests that DG and AMR could be made more similar by applying simple transformations targeting problematic constructions like coordination and named entities.

4 Evaluation of automatic aligners

We use our annotations to measure the accuracy of AMR aligners on specific phenomena that were inexpressible in previous annotation schemes. Our experiments evaluate the JAMR heuristic aligner (Flanigan et al., 2014), the ISI statistical aligner (Pourdamghani et al., 2014), and a heuristic rule-based aligner that we developed specifically for structural alignment.

4.1 Rule-based aligner

Our aligner operates in two passes: one for lexical alignment and one for structural alignment.

Lexical alignment algorithm. AMR concepts are cognate with English words, so we align them by lexical similarity. This algorithm does not make use of the DG. Before alignment, we remove sense identifiers on AMR node labels, and lemmatize DG node labels. Then for every pair of nodes a from the AMR and d from the DG we align them if any of the following conditions holds:

1. The Levenshtein distance of a and d is 15% or less of the length of the longer word.[13]
2. The label of a is the morphological negation of d (e.g. *prudent* ~ `imprudent`).[14]
3. The label of a is – (AMR's annotation of negation) and the parent of a aligns to d via rule 2.
4. The label of a is – and d is one of *no*, *none*, *not*, or *never*.
5. The label of a consists of multiple words, and the label of d matches any one of them under rule 1. (e.g. *sit* ~ `sit-down`, *war-torn* ~ `war`).[15]
6. Labels of a and d likely have the same morphological root. We determine this by segmenting each word with Morfessor (Grönroos et al., 2014) trained on Wiki data and applying rule 1 to the first morpheme of each word.

Note that if a word type is repeated in a sentence, each repetition is aligned to the same AMR nodes under the above rules.

Structural alignment algorithm. We align subgraphs using the procedure below, first from AMR to DG, then from DG to AMR. For clarity, the explanation refers to the first case.

[12] An AMR concept evoked by a preposition usually dominates the structure (after$\xrightarrow{\text{op1}}$date-entity$\xrightarrow{\text{decade}}$nineties), which is at odds with UD's prepositions-as-case-markers policy (*nineties*$\xrightarrow{\textit{case}}$*after*).

[13] Threshold was determined empirically on a 10% sample from the dataset.

[14] We use a list of morphologically negated words provided by Ulf Hermjakob.

[15] This rule misaligns some AMR-specific node types, such as *government* ~ `government-organization`.

aligner	dataset: our			ISI			JAMR		
our	89	85	**87**	88	77	82	55	81	65
ISI	71	68	70	96	85	**90**	47	67	55
JAMR	86	63	72	95	66	78	92	85	**88**

Table 3: Lexical alignment (precision, recall, F_1-score). Our *lexical* alignment algorithm does not use syntax.

Local phase. For every AMR edge e_a whose endpoints are lexically aligned nodes a_1 (aligned to d_1) and a_2 (aligned to d_2), we attempt to align minimal and connected AMR and dependency subgraphs, a' and d':

1. If there is a DG edge e_d whose endpoints are d_1 and d_2, then $a' \leftarrow e_a$ and $d' \leftarrow e_d$.
2. Otherwise, let π_d be the shortest undirected path between d_1 and d_2. If all lexically aligned nodes in π_d are aligned to a_1 or a_2, then $a' \leftarrow e_a$ and $d' \leftarrow \pi_d$.
3. Otherwise, let a'' be the smallest subgraph covering all AMR nodes that are lexically aligned to nodes in π_d. If all the nodes in a'' are aligned only to nodes in π_d, then $a' \leftarrow a''$ and $d' \leftarrow \pi_d$.
4. Otherwise, the attempt is abandoned.
5. Finally, if the top node of a' has a parent node labeled with an entity type concept, extend a' to include the parent. (This step is performed only in the AMR-to-DG step.)

Global phase. The local phase might produce alignments that violate the Subsumption Principle (§2.3.1), so we filter them out heuristically. For every pair of structural alignments, $\pi_d \sim \pi_a$ and $\pi'_d \sim \pi'_a$ where π_a overlaps with π'_a, or π_d with π'_d, if the region of overlap is not itself an aligned subgraph, we prune both alignments.[16]

4.2 Experiments

We evaluate JAMR, ISI, and our aligner on two distinct tasks.

Lexical alignment. Lexical alignment involves aligning AMR nodes to words, a task all three systems can perform. We evaluate against three datasets: our own, the JAMR dataset (Flanigan et al., 2014), and the ISI dataset (Pourdamghani et al., 2014).[17] Results (table 3) suggest that this task is already well-addressed, but also that there exist marked differences between how lexical alignment is defined in each dataset and that aligners are fine-tuned to their dataset.

For our aligner, errors are due to faulty morphological analysis, duplicated words, and both accidental string similarity between AMR concepts and words and occasional lack of similarity between concepts and words that should be aligned.

lexical alignments	prec., rec., F_1 using gold DGs			prec., rec., F_1 using automatic DGs		
gold	79	73	76	70	63	66
our aligner	68	56	61	63	48	55
ISI	65	50	57	58	44	50
JAMR	71	41	52	61	34	44

Table 4: Structural alignment (§4.1) scores, with different sources of input lexical alignments. Scores are shown for gold standard and automatic UD trees.

Structural alignment. An important goal of our experiments is to establish baselines for the structural alignment task. While we cannot evaluate the JAMR and ISI aligners directly on this task, we can use the lexical alignments they output in place of the first pass of our aligner. The only dataset for this task is our own. The results (table 4) evaluate accuracy of structural alignments only and do not count lexical alignments.

The automatic alignments have lower coverage of AMRs than the gold alignments do: our best aligner leaves 13.3% of AMR nodes and 30.0% of AMR edges unaligned, compared to 0.07% and 2.8% in the gold standard. The aligner also leaves 39.2% of DG nodes and 47.7% of DG edges unaligned, compared to 28.6% and 34.8% in the gold standard. The relatively low F-score for the gold standard lexical alignments and DGs condition suggests that substantial improvements to our structural alignment algorithm are possible. The two most common reasons for low recall were missing one of the conjuncts in a coordinate structure and aligning structures that violate the principle of minimality.

Our corpus gives alignments between AMRs and gold standard dependency parses. To see how much performance degrades when such parses are not available we also evaluate on automatic parses.[18] Both precision and recall are substantially worse when the aligner relies on automatic syntax.

5 Improving error analysis for AMR parsers

Our corpus of manually aligned AMRs can be used to identify linguistic constructions which cause

[16] This could be order-dependent since the removal of one alignment could trigger the removal of others, but our aligner does not account for this.

[17] We remove span alignments in the JAMR dataset and edge alignments in the ISI dataset.

[18] We use the CoreNLP dependency parser with settings as described in §3.

UD structure	missed	mislabeled
nsubj	103 (40%)	14 (6%)
nmod + case	74 (44%)	26 (16%)
compound	55 (41%)	7 (5%)
amod	40 (26%)	9 (6%)
dobj	40 (33%)	6 (5%)
advmod	30 (39%)	7 (9%)
cc + conj	29 (57%)	4 (8%)
nmod	21 (60%)	1 (3%)

Table 5: Error analysis of the AMR parser of Damonte et al. (2017). Frequency of dependency structures aligned to AMR edges which the automatic AMR parser missed altogether or mislabeled; absolute count (% of all such aligned structures in the corpus).

problems for an AMR parser. We parsed the sentences from our corpus with the parser of Damonte et al. (2017).[19] We map the nodes of the resulting automatic AMRs to the gold AMRs using the smatch evaluation tool (Cai and Knight, 2013), and on the basis of this mapping identify those nodes and edges of the gold AMRs which are missing or mislabeled in the automatic AMRs.

We then measured the number and rate of erroneous AMR fragments associated with each UD relation or construction (table 5). The largest proportion of recall errors were for fragments associated with the subject relation, prepositional phrases, and nominal compounds. Focusing on the subject relation, we can further say that 69% of the missing or mislabeled edges have the gold label `ARG0`, 19% `ARG1`, and the rest are distributed amongst `domain`, `ARG2`, `purpose` and `mod`. Inspecting the errors we see that phenomena underlying them include pronominal coreference, sharing arguments between conjoined predicates, auxiliary verb constructions, and control and raising.[20]

Our corpus facilitates fine-grained error analysis of AMR parsers with respect to individual syntactic constructions. We release the code for the above analysis in order to encourage syntactically-informed comparison and improvement of systems.

6 Conclusion

We have presented a new framework and corpus for aligning AMRs to dependency syntax. Our data and analysis show that the vast majority of the semantics in AMR graphs can be mapped to the lexical and syntactic structure of a sentence, though current alignment systems do not fully capture this correspondence. The syntax–semantics correspondences are often structurally divergent (non-isomorphic). Simple algorithms for lexical and structural alignment establish baselines for the new alignment task; we expect statistical models will be brought to bear on this task in future work. Our framework also facilitates syntactically-based analysis of AMR parsers. We release our data and code for the benefit of the research community.

[19] The overall smatch score of the parser on this dataset was 0.65.

[20] The missing edge counts include gold edges for which the parser failed to produce one or both endpoints.

Acknowledgments

This work was supported in part by EU ERC Advanced Fellowship 249520 GRAMPLUS and EU ERC H2020 Advanced Fellowship GA 742137 SEMANTAX.

We thank Sameer Bansal, Marco Damonte, Lucia Donatelli, Federico Fancellu, Sharon Goldwater, Andreas Grivas, Yova Kementchedjhieva, Junyi Li, Joana Ribeiro, and the anonymous reviewers for helpful discussion of this work and comments on previous drafts of the paper.

References

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. In *Proc. of EMNLP*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proc. of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*.

Emily M. Bender, Dan Flickinger, Stephan Oepen, Woodley Packard, and Ann A. Copestake. 2015. Layers of interpretation: On grammar and compositionality. In *Proc. of IWCS*.

Roger Brown. 1973. *A first language: The early stages*. Harvard University Press.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proc. of ACL*.

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proc. of EMNLP*.

Wei-Te Chen and Martha Palmer. 2017. Unsupervised AMR-dependency parse alignment. In *Proc. of EACL*.

Chenhui Chu and Sadao Kurohashi. 2016. Supervised syntax-based alignment between English sentences and Abstract Meaning Representation graphs. *arXiv preprint* `http://arxiv.org/abs/1606.02126`.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for Abstract Meaning Representation. In *Proc. of EACL*.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime G. Carbonell. 2016. Generation from Abstract Meaning Representation using tree transducers. In *Proc. of HLT-NAACL*.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the Abstract Meaning Representation. In *Proc. of ACL*.

Stig-Arne Grönroos, Sami Virpioja, Peter Smit, and Mikko Kurimo. 2014. Morfessor FlatCat: An HMM-based method for unsupervised and semi-supervised learning of morphology. In *Proc. of COLING*.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2017. A transition-based directed acyclic graph parser for UCCA. In *Proc. of ACL*.

Kevin Knight, Laura Baranescu, Claire Bonial, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Daniel Marcu, Martha Palmer, and Nathan Schneider. 2014. Abstract Meaning Representation (AMR) Annotation Release 1.0. Technical Report LDC2014T12, Linguistic Data Consortium, Philadelphia, Pennsylvania, USA. https://catalog.ldc.upenn.edu/LDC2014T12.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proc. of ACL*.

Geert-Jan M. Kruijff. 2006. Dependency grammar. In Keith Brown, editor, *Encyclopedia of Language and Linguistics (Second Edition)*, Elsevier, Oxford, pages 444–450.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proc. of ACL System Demonstrations*.

David Mareček, Martin Popel, Loganathan Ramasamy, Jan Štěpánek, Daniel Zeman, Zdeněk Žabokrtský, and Jan Hajič. 2013. Cross-language study on influence of coordination style on dependency parsing performance. Technical Report 49, ÚFAL MFF UK.

Dipendra Kumar Misra and Yoav Artzi. 2016. Neural shift-reduce CCG semantic parsing. In *Proc. of EMNLP*.

Joakim Nivre. 2005. Dependency grammar and dependency parsing. Technical Report MSI report 05133, Växjö University School of Mathematics and Systems Engineering, Växjö, Sweden.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajič, Angelina Ivanova, and Yi Zhang. 2014. SemEval 2014 Task 8: Broad-Coverage Semantic Dependency Parsing. In *Proc. of SemEval*.

Barbara H. Partee. 2014. A brief history of the syntax-semantics interface in Western formal linguistics. *Semantics-Syntax Interface* 1:1–20.

Xiaochang Peng, Chuan Wang, Daniel Gildea, and Nianwen Xue. 2017. Addressing the data sparsity issue in neural AMR parsing. In *Proc. of EACL*.

Nima Pourdamghani, Yang Gao, Ulf Hermjakob, and Kevin Knight. 2014. Aligning English strings with Abstract Meaning Representation graphs. In *Proc. of EMNLP*.

Nima Pourdamghani, Kevin Knight, and Ulf Hermjakob. 2016. Generating English from Abstract Meaning Representations. In *Proc. of INLG*.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming dependency structures to logical forms for semantic parsing. *Transactions of the Association for Computational Linguistics* 4:127–140.

Sebastian Schuster and Christopher D. Manning. 2016. Enhanced English Universal Dependencies: an improved representation for natural language understanding tasks. In *Proc. of LREC*.

Gabriel Stanovsky, Jessica Ficler, Ido Dagan, and Yoav Goldberg. 2016. Getting more out of syntax with PropS. *arXiv preprint* http://arxiv.org/abs/1603.01648.

Rik van Noord and Johan Bos. 2017. Neural semantic parsing by character-based translation: experiments with Abstract Meaning Representations. *arXiv preprint* http://arxiv.org/abs/1705.09980.

Chuan Wang, Nianwen Xue, Sameer Pradhan, and Sameer Pradhan. 2015. A transition-based algorithm for AMR parsing. In *Proc. of NAACL-HLT*.

Aaron Steven White, Drew Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme. 2016. Universal Decompositional Semantics on Universal Dependencies. In *Proc. of EMNLP*.

Sheng Zhang, Rachel Rudinger, and Benjamin Van Durme. 2017. An evaluation of PredPatt and Open IE via stage 1 semantic role labeling. In *Proc. of IWCS*.

A Details of alignment guidelines

A.1 Lexical alignments

Names. In proper names, individual strings denoting words in the name are lexically aligned, but the entity as a whole is structurally aligned.

Entity types. If the entity type is based on a common noun which occurs in the sentence, it is lexically aligned: e.g., *Jon, a clumsy man, has a cat* would involve the alignment *man* ~ `man`. Most often, however, an entity type is not explicitly mentioned in the sentence and is taken from AMR's ontology of entity types (`http://www.isi.edu/~ulf/amr/lib/ne-types.html`), in which case it will not be lexically aligned.

Case marking and prepositions. The possessive marker *'s* and many prepositions participate in structural but not lexical alignments because they are inherently relational. However, we align a preposition if it carries sufficient lexical content to be included as an AMR node (e.g., the AMR for *The cat is under the table* would include `under` $\xrightarrow{\texttt{op1}}$ `table`).

Wh-questions. The special concept `amr-unknown` aligns lexically to the wh-word whose referent is questioned. For multiword wh-expressions like *how much*, the expression is aligned structurally (not lexically) to `amr-unknown`.

Sentence mood. In AMR, non-wh questions are indicated by $\xrightarrow{\texttt{mode}}$`interrogative`, imperatives by $\xrightarrow{\texttt{mode}}$`imperative`, and exclamations/interjections by $\xrightarrow{\texttt{mode}}$`expressive`. UD parses do not encode sentence mood, which can be conveyed by non-canonical word order (subject-auxiliary inversion for questions) or argument omission (subject omission for imperatives), rather than the presence of certain relations or words. Sometimes the sentence includes an appropriate alignment point, e.g. complementizers *whether* and *if* for `interrogative`, allowing for a lexical alignment. More often the parse has no obvious alignment point, and the constant `interrogative`, `imperative`, or `expressive` is left unaligned.[21]

A.2 Structural alignments

Copulas. In UD, copulas are treated as modifiers of a predicate nominal or adjective, which is linked directly to the subject of the sentence via an ***nsubj*** dependency. We do not align copulas or the ***cop*** edge. Thus, in figure 3, there is a structural alignment between *general* $\xleftarrow{\textit{nsubj}}$ *confident* and the AMR subgraph connecting the lexically aligned nodes.

Control. The subject of the control verb and the controlled predicate are connected by the ***nsubj-xsubj*** edge, which can be structurally aligned with the corresponding AMR argument relation, as in e.g. figure 2.

Relative clauses. In enhanced UD the noun governing a relative clause and the embedded predicate are linked by edges in both directions: a "surface syntax" ***acl-relcl*** edge headed by the noun, and a "deep syntax" edge such as ***nsubj***, ***dobj***, ***iobj***, or ***nmod*** headed by the embedded predicate. Each participates in a structural alignment with the corresponding AMR subgraph. The relative pronoun is left unaligned.

Coordination. Coordination does not naturally lend itself to analysis with dependencies, and different dependency grammar traditions offer different approaches (Nivre, 2005; Mareček et al., 2013). UD follows the Stanford style, where the first conjunct serves as the head of the remaining conjuncts, and the conjunction is a dependent of one of the conjuncts.[22] In AMR the conjunction heads all the conjuncts (Prague style). In light of this mismatch, we use a subgraph alignment to group the conjunction with its conjuncts on each side. A simple example is illustrated in figure 2. A quirk of UD's approach to coordination is that it does not distinguish modifiers of the first conjunct from modifiers of the coordinate structure as a whole. The basic UD parse of *her glee and greed* is therefore ambiguous. We rely on an extra edge in the enhanced parse between *her* and *greed* to establish an alignment for the AMR edge `greed` $\xrightarrow{\texttt{ARG0}}$ `person`.

The coordination in figure 3 is more complex: the coordinated modifier *defense and security* distributes over *capabilities* (i.e., there are two kinds of capabilities). In the enhanced parse, *defense* and *security* are both attached as modifiers of *capabilities*. This is expressed semantically via **duplicate** AMR nodes labeled `capable`, each receiving different modifiers corresponding to different conjuncts. Independent of coordination, the two `capable` nodes also share a common argument, `nation`. The three syntactic modifiers give rise to three subgraph alignments, and the subgraph alignment covering the coordinate structure (cyan in the figure) envelops two of these. **Ellipsis** construc-

[21] Among the UD community there has been discussion of possibly adding sentence-level marking of mood (`https://github.com/UniversalDependencies/docs/issues/458`), which could provide a convenient alignment point.

[22] In UD version 1, and therefore the examples in this paper, the conjunction attaches to the first conjunct, whereas in version 2 it attaches to the next successive conjunct (`http://universaldependencies.org/v2/summary.html`).

tions can also trigger node duplication in AMR, requiring similar structural alignments.

Named entities. AMR annotates each named entity with a node representing the name, linked to the strings of the name and headed by an entity type. This full structure is aligned to the full name in the dependency parse.

Coreferent mentions. Coreference often causes an AMR structure to align to multiple DG subgraphs. For example, in figure 2, both the pronoun *her* and the name align to the AMR subgraph representing the entity. This mechanism suffices to represent coreference between mentions in the sentence.

Light verbs. Light verbs have no lexical alignment, but a subgraph alignment covers the light verb construction as a unit (e.g. *makes* $\xrightarrow{dobj}$ *attempt* ~ `attempt-01` in figure 2). All subgraph alignments which involve the light verb or its complement have to involve to whole unit, as shown in the alignment highlighted in red in figure 2.

Multiword expressions. In verb-particle constructions and fixed grammatical expressions the AMR node lexically aligns to all words in the expression, and additionally to the DG subgraph spanning the whole expression. (e.g. *pay* ~ `pay-off-02`, *off* ~ `pay-off-02`, and *pay* $\xrightarrow{compound\text{-}prt}$ *off* ~ `pay-off-02`).

Prepositional phrases. PP modifiers typically involve an extra dependency edge for the preposition attachment, as with *lies* $\xrightarrow{nmod\text{-}in}$ *sun* $\xrightarrow{case}$ in ~ `lie-07` $\xrightarrow{location}$ `sun`.

Semantically decomposed words. When one word has multiple lexical alignments because of morphological decomposition, there also exists a structural alignment between that word and an AMR subgraph representing the decomposition: e.g., in figure 2, *evildoer* ~ `person` $\xrightarrow{ARG0\text{-}of}$ `do-02` $\xrightarrow{ARG1}$ `thing` $\xrightarrow{mod}$ `evil`, and in figure 3, *general* ~ `person` $\xrightarrow{ARG0\text{-}of}$ `have-org-role-91` $\xrightarrow{ARG2}$ `general`. AMR decomposes certain words by convention which must always be structurally aligned, such as *ago* ~ `before` $\xrightarrow{op1}$ `now` and *government* ~ `government-organization` $\xrightarrow{ARG0\text{-}of}$ `govern-01`.

Date, time, and value expressions. These expressions are aligned similarly to named entities, even though the normalized constants may not exactly match the words in the sentence. For example, the DG structure *9:00* $\xleftarrow{nummod}$ *pm* would be represented in the AMR as `date-entity` $\xrightarrow{time}$ `21:00`; tokens *9:00* and *pm* are treated as a multiword expression: each is lexically aligned to `"21:00"`. Moreover, we also align *9:00* $\xleftarrow{nummod}$ *pm* ~ `21:00` and *9:00* $\xleftarrow{nummod}$ *pm* ~ `date-entity` $\xrightarrow{time}$ `21:00`.

End-to-end Graph-based TAG Parsing with Neural Networks

Jungo Kasai[♣] **Robert Frank**[♣] **Pauli Xu**[♣]
William Merrill[♣] **Owen Rambow**[♡]
[♣]Department of Linguistics, Yale University
[♡]Elemental Cognition, LLP
{jungo.kasai,bob.frank,pauli.xu,william.merrill}@yale.edu
owenr@elementalcognition.com

Abstract

We present a graph-based Tree Adjoining Grammar (TAG) parser that uses BiLSTMs, highway connections, and character-level CNNs. Our best end-to-end parser, which jointly performs supertagging, POS tagging, and parsing, outperforms the previously reported best results by more than 2.2 LAS and UAS points. The graph-based parsing architecture allows for global inference and rich feature representations for TAG parsing, alleviating the fundamental trade-off between transition-based and graph-based parsing systems. We also demonstrate that the proposed parser achieves state-of-the-art performance in the downstream tasks of Parsing Evaluation using Textual Entailments (PETE) and Unbounded Dependency Recovery. This provides further support for the claim that TAG is a viable formalism for problems that require rich structural analysis of sentences.

1 Introduction

Tree Adjoining Grammar (TAG, Joshi and Schabes (1997)) and Combinatory Categorial Grammar (CCG, Steedman and Baldridge (2011)) are both mildly context-sensitive grammar formalisms that are lexicalized: every elementary structure (elementary tree for TAG and category for CCG) is associated with exactly one lexical item, and every lexical item of the language is associated with a finite set of elementary structures in the grammar (Rambow and Joshi, 1994). In TAG and CCG, the task of parsing can be decomposed into two phases (e.g. TAG: Bangalore and Joshi (1999); CCG: Clark and Curran (2007)): *supertagging*, where elementary units or *supertags* are assigned to each lexical item and *parsing* where these supertags are combined together. The first phase of supertagging can be considered as "almost parsing" because supertags for a sentence almost always determine a unique parse (Bangalore and Joshi, 1999). This near uniqueness of a parse given a gold sequence of supertags has been confirmed empirically (TAG: Bangalore et al. (2009); Chung et al. (2016); Kasai et al. (2017); CCG: Lewis et al. (2016)).

We focus on TAG parsing in this work. TAG differs from CCG in having a more varied set of supertags. Concretely, the TAG-annotated version of the WSJ Penn Treebank (Marcus et al., 1993) that we use (Chen et al., 2005) includes 4727 distinct supertags (2165 occur once) while the CCG-annotated version (Hockenmaier and Steedman, 2007) only includes 1286 distinct supertags (439 occur once). This large set of supertags in TAG presents a severe challenge in supertagging and causes a large discrepancy in parsing performance with gold supertags and predicted supertags (Bangalore et al., 2009; Chung et al., 2016; Kasai et al., 2017).

In this work, we present a supertagger and a parser that substantially improve upon previously reported results. We propose crucial modifications to the bidirectional LSTM (BiLSTM) supertagger in Kasai et al. (2017). First, we use character-level Convolutional Neural Networks (CNNs) for encoding morphological information instead of suffix embeddings. Secondly, we perform concatenation after each BiLSTM layer. Lastly, we explore the impact of adding additional BiLSTM layers and highway connections. These techniques yield an increase of 1.3% in accuracy. For parsing, since the derivation tree in a lexicalized TAG is a type of dependency tree (Rambow and Joshi, 1994), we can directly apply dependency parsing models. In particular, we use the biaffine graph-based parser proposed by Dozat and Manning (2017) together with our novel techniques for supertagging.

In addition to these architectural extensions for supertagging and parsing, we also explore multi-task learning approaches for TAG parsing. Specif-

Proceedings of NAACL-HLT 2018, pages 1181–1194
New Orleans, Louisiana, June 1 - 6, 2018.

ically, we perform POS tagging, supertagging, and parsing using the same feature representations from the BiLSTMs. This joint modeling has the benefit of avoiding a time-consuming and complicated pipeline process, and instead produces a full syntactic analysis, consisting of supertags and the derivation that combines them, simultaneously. Moreover, this multi-task learning framework further improves performance in all three tasks. We hypothesize that our multi-task learning yields feature representations in the LSTM layers that are more linguistically relevant and that generalize better (Caruana, 1997). We provide support for this hypothesis by analyzing syntactic analogies across induced vector representations of supertags (Kasai et al., 2017; Friedman et al., 2017). The end-to-end TAG parser substantially outperforms the previously reported best results.

Finally, we apply our new parsers to the downstream tasks of Parsing Evaluation using Textual Entailements (PETE, Yuret et al. (2010)) and Unbounded Dependency Recovery (Rimell et al., 2009). We demonstrate that our end-to-end parser outperforms the best results in both tasks. These results illustrate that TAG is a viable formalism for tasks that benefit from the assignment of rich structural descriptions to sentences.

2 Our Models

TAG parsing can be decomposed into *supertagging* and *parsing*. Supertagging assigns to words elementary trees (supertags) chosen from a finite set, and parsing determines how these elementary trees can be combined to form a derivation tree that yield the observed sentence. The combinatory operations consist of *substitution*, which inserts obligatory arguments, and *adjunction*, which is responsible for the introduction of modifiers, function words, as well as the derivation of sentences involving long-distance dependencies. In this section, we present our supertagging models, parsing models, and joint models.

2.1 Supertagging Model

Recent work has explored neural network models for supertagging in TAG (Kasai et al., 2017) and CCG (Xu et al., 2015; Lewis et al., 2016; Vaswani et al., 2016; Xu, 2016), and has shown that such models substantially improve performance beyond non-neural models. We extend previously proposed BiLSTM-based models (Lewis et al., 2016; Kasai et al., 2017) in three ways: 1) we add character-level Convolutional Neural Networks (CNNs) to the input layer, 2) we perform concatenation of both directions of the LSTM not only after the final layer but also after each layer, and 3) we use a modified BiLSTM with highway connections.

2.1.1 Input Representations

The input for each word is represented via concatenation of a 100-dimensional embedding of the word, a 100-dimensional embedding of a predicted part of speech (POS) tag, and a 30-dimensional character-level representation from CNNs that have been found to capture morphological information (Santos and Zadrozny, 2014; Chiu and Nichols, 2016; Ma and Hovy, 2016). The CNNs encode each character in a word by a 30 dimensional vector and 30 filters produce a 30 dimensional vector for the word. We initialize the word embeddings to be the pre-trained GloVe vectors (Pennington et al., 2014); for words not in GloVe, we initialize their embedding to a zero vector. The other embeddings are randomly initialized. We obtain predicted POS tags from a BiLSTM POS tagger with the same configuration as in Ma and Hovy (2016).

2.1.2 Deep Highway BiLSTM

The core of the supertagging model is a deep bidirectional Long Short-Term Memory network (Graves and Schmidhuber, 2005). We use the following formulas to compute the activation of a single LSTM cell at time step t:

$$i_t = \sigma\left(W_i[x_t; h_{t-1}] + b_i\right) \quad (1)$$
$$f_t = \sigma\left(W_f[x_t; h_{t-1}] + b_f\right) \quad (2)$$
$$\tilde{c}_t = \tanh\left(W_c[x_t; h_{t-1}] + b_c\right) \quad (3)$$
$$o_t = \sigma\left(W_o[x_t; h_{t-1}] + b_o\right) \quad (4)$$
$$c_t = f \odot c_{t-1} + i_t \odot \tilde{c}_t \quad (5)$$
$$h_t = o \odot \tanh\left(c_t\right) \quad (6)$$

Here a semicolon ; means concatenation, $\odot$ is element-wise multiplication, and σ is the sigmoid function. In the first BiLSTM layer, the input x_t is the vector representation of word t. (The sequence is reversed for the backwards pass.) In all subsequent layers, x_t is the corresponding output from the previous BiLSTM; the output of a BiLSTM at timestep t is equal to $[h_t^f; h_t^b]$, the concatenation of hidden state corresponding to input t in the forward and backward pass. This concatenation af-

ter each layer differs from Kasai et al. (2017) and Lewis et al. (2016), where concatenation happens only after the final BiLSTM layer. We will show in a later section that concatenation after each layer contributes to improvement in performance.

We also extend the models in Kasai et al. (2017) and Lewis et al. (2016) by allowing highway connections between LSTM layers. A highway connection is a gating mechanism that combines the current and previous layer outputs, which can prevent the problem of vanishing/exploding gradients (Srivastava et al., 2015). Specifically, in networks with highway connections, we replace Eq. 6 by:

$$r_t = \sigma\left(W_r[x_t; h_{t-1}] + b_r\right)$$
$$h_t = r_t \odot o_t \odot \tanh\left(c_t\right) + (1 - r_t) \odot W_h x_t$$

Indeed, our experiments will show that highway connections play a crucial role as we add more BiLSTM layers.

We generally follow the hyperparameters chosen in Lewis et al. (2016) and Kasai et al. (2017). Specifically, we use BiLSTMs layers with 512 units each. Input, layer-to-layer, and recurrent (Gal and Ghahramani, 2016) dropout rates are all 0.5. For the CNN character-level representation, we used the hyperparameters from Ma and Hovy (2016).

We train this network, including the embeddings, by optimizing the negative log-likelihood of the observed sequences of supertags in a mini-batch stochastic fashion with the Adam optimization algorithm with batch size 100 and $\ell = 0.01$ (Kingma and Ba, 2015). In order to obtain predicted POS tags and supertags of the training data for subsequent parser input, we also perform 10-fold jackknife training. After each training epoch, we test the supertagger on the dev set. When classification accuracy does not improve on five consecutive epochs, training ends.

2.2 Parsing Model

Until recently, TAG parsers have been grammar based, requiring as input a set of elemenetary trees (supertags). For example, Bangalore et al. (2009) proposes the MICA parser, an Earley parser that exploits a TAG grammar that has been transformed into a variant of a probabilistic CFG. One advantage of such a parser is that its parses are guaranteed to be well-formed according to the TAG grammar provided as input.

More recent work, however, has shown that data-driven transition-based parsing systems outperform such grammar-based parsers (Chung et al., 2016; Kasai et al., 2017; Friedman et al., 2017). Kasai et al. (2017) and Friedman et al. (2017) achieved state-of-the-art TAG parsing performance using an unlexicalized shift-reduce parser with feed-forward neural networks that was trained on a version of the Penn Treebank that had been annotated with TAG derivations. Here, we pursue this data-driven approach, applying a graph-based parser with deep biaffine attention (Dozat and Manning, 2017) that allows for global training and inference.

2.2.1 Input Representations

The input for each word is the concatenation of a 100-dimensional embedding of the word and a 30-dimensional character-level representation obtained from CNNs in the same fashion as in the supertagger.[1] We also consider adding 100-dimensional embeddings for a predicted POS tag (Dozat and Manning, 2017) and a predicted supertag (Kasai et al., 2017; Friedman et al., 2017). The ablation experiments in Kiperwasser and Goldberg (2016) illustrated that adding predicted POS tags boosted performance in Stanford Dependencies. In Universal Dependencies, Dozat et al. (2017) empirically showed that their dependency parser gains significant improvements by using POS tags predicted by a Bi-LSTM POS tagger. Indeed, Kasai et al. (2017) and Friedman et al. (2017) demonstrated that their unlexicalized neural network TAG parsers that only get as input predicted supertags can achieve state-of-the-art performance, with lexical inputs providing no improvement in performance. We initialize word embeddings to be the pre-trained GloVe vectors as in the supertagger. The other embeddings are randomly initialized.

2.2.2 Biaffine Parser

We train our parser to predict edges between lexical items in an LTAG derivation tree. Edges are labeled by the operations together with the deep syntactic roles of substitution sites (0=underlying subject, 1=underlying direct object, 2=underlying indirect object, 3,4=oblique arguments, CO=co-head for prepositional/particle verbs, and adj=all adjuncts). Figure 1 shows our biaffine parsing ar-

[1] We fix the embedding of the ROOT token to be a 0-vector.

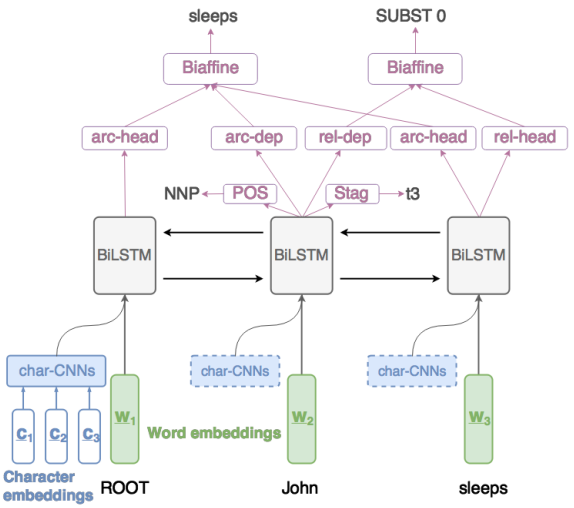


Figure 1: Biaffine parsing architecture. For the dependency from *John* to *sleeps* in the sentence *John sleeps*, the parser first predicts the head of *John* and then predicts the dependency label by combining the dependent and head representations. In the joint setting, the parser also predicts POS tags and supertags.

chitecture. Following Dozat and Manning (2017) and Kiperwasser and Goldberg (2016), we use BiLSTMs to obtain features for each word in a sentence. We add highway connections in the same fashion as our supertagging model.

We first perform unlabeled arc-factored scoring using the final output vectors from the BiLSTMs, and then label the resulting arcs. Specifically, suppose that we score edges coming into the ith word in a sentence i.e. assigning scores to the potential parents of the ith word. Denote the final output vector from the BiLSTM for the kth word by h_k and suppose that h_k is d-dimensional. Then, we produce two vectors from two separate multilayer perceptrons (MLPs) with the ReLU activation:

$$h_k^{\text{arc-dep}} = \text{MLP}^{(\text{arc-dep})}(h_k)$$
$$h_k^{\text{arc-head}} = \text{MLP}^{(\text{arc-head})}(h_k)$$

where $h_k^{\text{arc-dep}}$ and $h_k^{\text{arc-head}}$ are d_{arc}-dimensional vectors that represent the kth word as a dependent and a head respectively. Now, suppose the kth row of matrix $H^{(\text{arc-head})}$ is $h_k^{\text{arc-head}}$. Then, the probability distribution s_i over the potential heads of the ith word is computed by

$$s_i = \text{softmax}(H^{(\text{arc-head})} W^{(\text{arc})} h_i^{\text{arc-dep}} + H^{(\text{arc-head})} b^{(\text{arc})}) \quad (7)$$

where $W^{(\text{arc})} \in \mathbb{R}^{d_{arc} \times d_{arc}}$ and $b^{(arc)} \in \mathbb{R}^{d_{arc}}$. In training, we simply take the greedy maximum probability to predict the parent of each word. In the testing phase, we use the heuristics formulated by Dozat and Manning (2017) to ensure that the resulting parse is single-rooted and acyclic.

Given the head prediction of each word in the sentence, we assign labeling scores using vectors obtained from two additional MLP with ReLU. For the kth word, we obtain:

$$h_k^{\text{rel-dep}} = \text{MLP}^{(\text{rel-dep})}(h_k)$$
$$h_k^{\text{rel-head}} = \text{MLP}^{(\text{rel-head})}(h_k)$$

where $h_k^{\text{rel-dep}}, h_k^{\text{rel-head}} \in \mathbb{R}^{d_{rel}}$. Let p_i be the index of the predicted head of the ith word, and r be the number of dependency relations in the dataset. Then, the probability distribution ℓ_i over the possible dependency relations of the arc pointing from the p_ith word to the ith word is calculated by:

$$\ell_i = \text{softmax}(h_{p_i}^{T(\text{rel-head})} U^{(\text{rel})} h_i^{(\text{rel-dep})} + W^{(\text{rel})}(h_i^{(\text{rel-head})} + h_{p_i}^{(\text{rel-head})}) + b^{(\text{rel})}) \quad (8)$$

where $U^{(\text{rel})} \in \mathbb{R}^{d_{rel} \times d_{rel} \times r}$, $W^{(\text{rel})} \in \mathbb{R}^{r \times d_{rel}}$, and $b^{(\text{rel})} \in \mathbb{R}^r$.

We generally follow the hyperparameters chosen in Dozat and Manning (2017). Specifically, we use BiLSTMs layers with 400 units each. Input, layer-to-layer, and recurrent dropout rates are all 0.33. The depths of all MLPs are all 1, and the MLPs for unlabeled attachment and those for labeling contain 500 (d_{arc}) and 100 (d_{rel}) units respectively. For character-level CNNs, we use the hyperparameters from Ma and Hovy (2016).

We train this model with the Adam algorithm to minimize the sum of the cross-entropy losses from head predictions (s_i from Eq. 7) and label predictions (ℓ_i from Eq. 8) with $\ell = 0.01$ and batch size 100 (Kingma and Ba, 2015). After each training epoch, we test the parser on the dev set. When labeled attachment score (LAS)[2] does not improve on five consecutive epochs, training ends.

2.3 Joint Modeling

The simple BiLSTM feature representations for parsing presented above are conducive to joint modeling of POS tagging and supertagging; rather than using POS tags and supertags to predict a derivation tree, we can instead use the BiLSTM hidden vectors derived from lexical inputs alone

[2]We disregard pure punctuation when evaluating LAS and UAS, following prior work (Bangalore et al., 2009; Chung et al., 2016; Kasai et al., 2017; Friedman et al., 2017).

to predict POS tags and supertags along with the TAG derivation tree.

$$h_k^{\text{pos}} = \text{MLP}^{(\text{pos})}(h_k)$$
$$h_k^{\text{stag}} = \text{MLP}^{(\text{stag})}(h_k)$$

where $h_k^{\text{pos}} \in \mathbb{R}^{d_{pos}}$ and $h_k^{\text{stag}} \in \mathbb{R}^{d_{stag}}$. We obtain probability distribution over the POS tags and supertags by:

$$\text{softmax}(W^{(\text{pos})} h_k^{\text{pos}} + b^{(\text{pos})}) \quad (9)$$
$$\text{softmax}(W^{(\text{stag})} h_k^{\text{stag}} + b^{(\text{stag})}) \quad (10)$$

where $W^{(\text{pos})}$, $b^{(\text{pos})}$, $W^{(\text{stag})}$, and $b^{(\text{stag})}$ are in $\mathbb{R}^{n_{pos} \times d_{pos}}$, $\mathbb{R}^{n_{pos}}$, $\mathbb{R}^{n_{stag} \times d_{stag}}$, and $\mathbb{R}^{n_{stag}}$ respectively, with n_{pos} and n_{stag} the numbers of possible POS tags and supertags respectively.

We use the same hyperparameters as in the parser. The MLPs for POS tagging and supertagging both contain 500 units. We again train this model with the Adam algorithm to minimize the sum of the cross-entropy losses from head predictions (s_i from Eq. 7), label predictions (ℓ_i from Eq. 8), POS predictions (Eq. 9), and supertag predictions (Eq. 10) with $\ell = 0.01$ and batch size 100. After each training epoch, we test the parser on the dev set and compute the percentage of each token that is assigned the correct parent, relation, supertag, and POS tag. When the percentage does not improve on five consecutive epochs, training ends.

This joint modeling has several advantages. First, the joint model yields a full syntactic analysis simultaneously without the need for training separate models or performing jackknife training. Secondly, joint modeling introduces a bias on the hidden representations that could allow for better generalization in each task (Caruana, 1997). Indeed, in experiments described in a later section, we show empirically that predicting POS tags and supertags does indeed benefit performance on parsing (as well as the tagging tasks).

3 Results and Discussion

We follow the protocol of Bangalore et al. (2009), Chung et al. (2016), Kasai et al. (2017), and Friedman et al. (2017); we use the grammar and the TAG-annotated WSJ Penn Tree Bank extracted by Chen et al. (2005). Following that work, we use Sections 01-22 as the training set, Section 00 as the dev set, and Section 23 as the test set. The training, dev, and test sets comprise 39832, 1921, and 2415 sentences, respectively. We implement all of our models in TensorFlow (Abadi et al., 2016).[3]

3.1 Supertaggers

Our BiLSTM POS tagger yielded 97.37% and 97.53% tagging accuracy on the dev and test sets, performance on par with the state-of-the-art (Ling et al., 2015; Ma and Hovy, 2016).[4] Seen in the middle section of Table 1 is supertagging performance obtained from various model configurations. "Final concat" in the model name indicates that vectors from forward and backward pass are concatenated only after the final layer. Concatenation happens after each layer otherwise. Numbers immediately after BiLSTM indicate the numbers of layers. CNN, HW, and POS denote respectively character-level CNNs, highway connections, and pipeline POS input from our BiLSTM POS tagger. Firstly, the differences in performance between BiLSTM2 (final concat) and BiLSTM2 and between BiLSTM2 and BiLSTM2-CNN suggest an advantage to performing concatenation after each layer and adding character-level CNNs. Adding predicted POS to the input somewhat helps supertagging though the difference is small. Adding a third BiLSTM layer helps only if there are highway connections, presumably because deeper BiLSTMs are more vulnerable to the vanishing/exploding gradient problem. Our supertagging model (BiLSTM3-HW-CNN-POS) that performs best on the dev set achieves an accuracy of 90.81% on the test set, outperforming the previously best result by more than 1.3%.

3.2 Parsers

Table 3 shows parsing results on the dev set. Abbreviations for models are as before with one addition: Stag denotes pipeline supertag input from our best supertagger (BiLSTM3-HW-CNN-POS in Table 1). As with supertagging, we observe a gain from adding character-level CNNs. Interestingly, adding predicted POS tags or supertags deteriorates performance with BiLSTM3. These results suggest that morphological information and word information from character-level CNNs and word embeddings overwhelm the in-

[3] Our code is available online for easy replication of our results at https://github.com/jungokasai/graph_parser.

[4] We cannot directly compare these results because the data split is different in the POS tagging literature.

Supertagger	Dev	Test
Bangalore et al. (2009)	88.52	86.85
Chung et al. (2016)	87.88	–
Kasai et al. (2017)	89.32	89.44
BiLSTM2 (final concat)	88.96	–
BiLSTM2	89.60	–
BiLSTM2-CNN	89.97	–
BiLSTM2-CNN-POS	90.03	–
BiLSTM2-HW-CNN-POS	90.12	–
BiLSTM3-CNN-POS	90.12	–
BiLSTM3-HW-CNN-POS	90.45	90.81
BiLSTM4-CNN-POS	89.99	–
BiLSTM4-HW-CNN-POS	90.43	–
Joint (Stag)	90.51	–
Joint (POS+Stag)	**90.67**	**91.01**

Table 1: Supertagging Results. Joint (Stag) and Joint (POS+Stag) indicate joint parsing models that perform supertagging, and POS tagging and supertagging respectively.

POS tagger	Dev	Test
BiLSTM	97.37	97.53
Joint (POS+Stag)	97.54	97.73

Table 2: POS tagging results.

formation from predicted POS tags and supertags. Again, highway connections become crucial as the number of layers increases. We finally evaluate the parsing model with the best dev performance (BiLSTM4-HW-CNN) on the test set (Table 3). It achieves 91.37 LAS points and 92.77 UAS points, improvements of 1.8 and 1.7 points respectively from the state-of-the-art.

3.3 Joint Models

We provide joint modeling results for supertagging and parsing in Tables 2 and 3. For these joint models, we employed the best parsing configuration (4 layers of BiLSTMs, character-level CNNs, and highway connections), with and without POS tagging added as an additional task. We can observe that our full joint model that performs

Parser	Dev		Test	
	UAS	LAS	UAS	LAS
Bangalore et al. (2009)	87.60	85.80	86.66	84.90
Chung et al. (2016)	89.96	87.86	–	–
Friedman et al. (2017)	90.36	88.91	90.31	88.96
Kasai et al. (2017)	90.88	89.39	90.97	89.68
BiLSTM3	91.75	90.22	–	–
BiLSTM3-CNN	92.27	90.76	–	–
BiLSTM3-CNN-POS	92.07	90.53	–	–
BiLSTM3-CNN-Stag	92.15	90.65	–	–
BiLSTM3-HW-CNN	92.29	90.71	–	–
BiLSTM4-CNN	92.11	90.66	–	–
BiLSTM4-HW-CNN	92.78	91.26	92.77	91.37
BiLSTM5-CNN	92.34	90.77	–	–
BiLSTM5-HW-CNN	92.64	91.11	–	–
Joint (Stag)	92.97	91.48	–	–
Joint (POS+Stag)	**93.22**	**91.80**	**93.26**	**91.89**
Joint (Shuffled Stag)	92.23	90.56	–	–

Table 3: Parsing results on the dev and test sets.

POS tagging, supertagging, and parsing further improves performance in all of the three tasks, yielding the test result of 91.89 LAS and 93.26 UAS points, an improvement of more than 2.2 points each from the state-of-the-art.

Figures 2 and 3 illustrate the relative performance of the feed-forward neural network shift-reduce TAG parser (Kasai et al., 2017) and our joint graph-based parser with respect to two of the measures explored by McDonald and Nivre (2011), namely dependency length and distance between a dependency and the root of a parse. The graph-based parser outperforms the shift-reduce parser across all conditions. Most interesting is the fact that the graph-based parser shows less of an effect of dependency length. Since the shift-reduce parser builds a parse sequentially with one parsing action depending on those that come before it, we would expect to find a propogation of errors made in establishing shorter dependencies to the establishment of longer dependencies.

Lastly, it is worth noting our joint parsing ar-

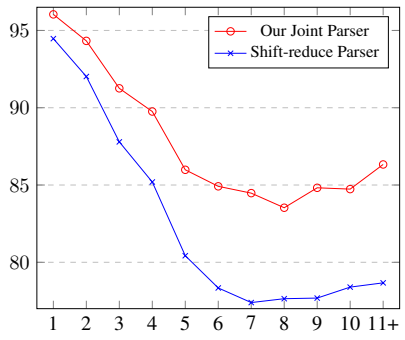


Figure 2: F1 Score with Dependency Length.

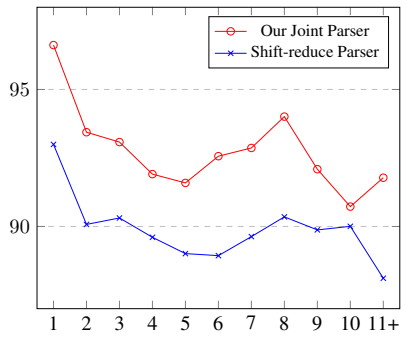


Figure 3: F1 Score with Distance to Root.

chitecture has a substantial advantage regarding parsing speed. Since POS tagging, supertagging, and parsing decisions are made independently for each word in a sentence, our system can parallelize computation once the sentence is encoded in the BiLSTM layers. Our current implementation processes 225 sentences per second on a single Tesla K80 GPU, an order of magnitude faster than the MICA system (Bangalore et al., 2009).[5]

4 Joint Modeling and Network Representations

Given the improvements we have derived from the joint models, we analyze the nature of inductive bias that results from multi-task training and attempt to provide an explanation as to why joint modeling improves performance.

4.1 Noise vs. Inductive Bias

One might argue that joint modeling improves performance merely because it adds noise to each task and prevents over-fitting. If the introduction of noise were the key, we would still expect to gain an improvement in parsing even if the target supertag were corrupted, say by shuffling the order of supertags for the entire training data (Caruana, 1997). We performed this experiment, and the result is shown as "Joint (Shuffled Stag)" in Table 3. Parsing performance falls behind the best non-joint parser by 0.7 LAS points. This suggests that inducing the parser to create representations to predict both supertags and a parse tree is beneficial for both tasks, beyond a mere introduction of noise.

4.2 Syntactic Analogies

We next analyze the induced vector representations in the output projection matrices of our supertagger and joint parsers using the syntactic analogy framework (Kasai et al., 2017). Consider, for instance, the analogy that an elementary tree representing a clause headed by a transitive verb (t27) is to a clause headed by an intransitive verb (t81) as a subject relative clause headed by a transitive verb (t99) is to a subject relative headed by an intransitive verb (t109). Following the ideas in Mikolov et al. (2013) for word analogies, we can express this structural analogy as t27 - t81 + t109 = t99 and test it by cosine similarity. Table 4 shows the results of the analogy test with 246 equations involving structural analogies with only the 300 most frequent supertags in the training data. While the embeddings (projection matrix) from the independently trained supertagger do not appear to reflect the syntax, those obtained from the joint models yield linguistic structure despite the fact that the supertag embeddings (projection matrix) is trained without any *a priori* syntactic knowledge about the elementary trees.

The best performance is obtained by the supertag representations obtained from the training of the transition-based parser Kasai et al. (2017) and Friedman et al. (2017). For the transition-based parser, it is beneficial to share statistics among the input supertags that differ only by a certain operation or property (Kasai et al., 2017) during the training phase, yielding the success in the analogy task. For example, a transitive verb supertag whose object has been filled by substitution should be treated by the parser in the same way as an intransitive verb supertag. In our graph-based parsing setting, we do not have a notion of parse history or partial derivations that directly connect intransitive and transitive verbs. However, syntactic analogies still hold to a considerable degree in the vector representations of supertags induced by our joint models, with average rank of the correct answer nearly the same as that obtained in the transition-based parser.

This analysis bolsters our hypothesis that joint training biases representation learning toward linguistically sensible structure. The supertagger is just trained to predict linear sequences of supertags. In this setting, many intervening supertags can occur, for instance, between a subject noun and its verb, and the supertagger might not be able to systematically link the presence of the two in the sequence. In the joint models, on the other hand, parsing actions will explicitly guide the network to associate the two supertags.

5 Downstream Tasks

Previous work has applied TAG parsing to the downstream tasks of syntactically-oriented textual entailment (Xu et al., 2017) and semantic role labeling (Chen and Rambow, 2003). In this work, we apply our parsers to the textual entailment and unbounded dependency recovery tasks and achieve state-of-the-art performance. These re-

[5] While such computational resources were not available in 2009, our parser differs from the MICA chart parser in being able to better exploit parallel computation enabled by modern GPUs.

Parser / Supertagger	%correct	Avg. rank
Transition-based	67.07	2.36
Our Supertagger	0.00	152.46
Our Joint (Stag)	29.27	2.55
Our Joint (POS+Stag)	30.08	2.57

Table 4: Syntactic analogy test results on the 300 most frequent supertags. Avg. rank is the average position of the correct choice in the ranked list of the closest neighbors; the top line indicates the result of using supertag embeddings that are trained jointly with a transition based parser (Friedman et al., 2017).

sults bolster the significance of the improvements gained from our joint parser and the utility of TAG parsing for downstream tasks.

5.1 PETE

Parser Evaluation using Textual Entailments (PETE) is a shared task from the SemEval-2010 Exercises on Semantic Evaluation (Yuret et al., 2010). The task was intended to evaluate syntactic parsers across different formalisms, focusing on entailments that could be determined entirely on the basis of the syntactic representations of the sentences that are involved, without recourse to lexical semantics, logical reasoning, or world knowledge. For example, syntactic knowledge alone tells us that the sentence *John, who loves Mary, saw a squirrel* entails *John saw a squirrel* and *John loves Mary* but not, for instance, that *John knows Mary* or *John saw an animal*. Prior work found the best performance was achieved with parsers using grammatical frameworks that provided rich linguistic descriptions, including CCG (Rimell and Clark, 2010; Ng et al., 2010), Minimal Recursion Semantics (MRS) (Lien, 2014), and TAG (Xu et al., 2017). Xu et al. (2017) provided a set of linguistically-motivated transformations to use TAG derivation trees to solve the PETE task. We follow their procedures and evaluation for our new parsers.

We present test results from two configurations in Table 5. One configuration is a pipeline approach that runs our BiLSTM POS tagger, supertagger, and parser. The other one is a joint approach that only uses our full joint parser. The joint method yields 78.1% in accuracy and 76.4% in F1, improvements of 2.4 and 2.7 points over the previously reported best results.

System	%A	%P	%R	F1
Rimell and Clark (2010)	72.4	79.6	62.8	70.2
Ng et al. (2010)	70.4	68.3	**80.1**	73.7
Lien (2014)	70.7	**88.6**	50.0	63.9
Xu et al. (2017)	75.7	88.1	61.5	72.5
Our Pipeline Method	77.1	86.6	66.0	74.9
Our Joint Method	**78.1**	86.3	68.6	**76.4**

Table 5: PETE test results. Precision (P), recall (R), and F1 are calculated for "entails."

5.2 Unbounded Dependency Recovery

The unbounded dependency corpus (Rimell et al., 2009) specifically evaluates parsers on unbounded dependencies, which involve a constituent moved from its original position, where an unlimited number of clause boundaries can intervene. The corpus comprises 7 constructions: object extraction from a relative clause (ObRC), object extraction from a reduced relative clause (ObRed), subject extraction from a relative clause (SbRC), free relatives (Free), object wh-questions (ObQ), right node raising (RNR), and subject extraction from an embedded clause (SbEm).

Because of variations across formalisms in their representational format for unbounded depdendencies, past work has conducted manual evaluation on this corpus (Rimell et al., 2009; Nivre et al., 2010). We instead conduct an automatic evaluation using a procedure that converts TAG parses to structures directly comparable to those specified in the unbounded dependency corpus. To this end, we apply two types of structural transformation in addition to those used for the PETE task:[6] 1) a more extensive analysis of coordination, 2) resolution of differences in dependency representations in cases involving copula verbs and co-anchors (e.g., verbal particles). See Appendix A for details. After the transformations, we simply check if the resulting dependency graphs contain target labeled arcs given in the dataset.

Table 6 shows the results. Our joint parser outperforms the other parsers, including the neural network shift-reduce TAG parser (Kasai et al., 2017). Our data-driven parsers yield relatively low performance in the ObQ and RNR constructions. Performance on ObQ is low, we expect, because of their rarity in the data on which the parser is

[6]One might argue that since the unbounded dependency evaluation is recall-based, we added too many edges by the transformations. However, it turns out that applying all the transformations for the corpus even improves performance on PETE (77.6 F1 score), which considers precision and recall, verifying that our transformations are reasonable.

System	ObRC	ObRed	SbRC	Free	ObQ	RNR	SbEm	Total	Avg
C&C (CCG)	59.3	62.6	80.0	72.6	**72.6**	**49.4**	22.4	53.6	61.1
Enju (HPSG)	47.3	65.9	82.1	76.2	32.5	47.1	32.9	54.4	54.9
Stanford (PCFG)	22.0	1.1	74.7	64.3	41.2	45.4	10.6	38.1	37.0
MST (Stanford Dependencies)	34.1	47.3	78.9	65.5	41.2	45.4	37.6	49.7	50.0
MALT (Stanford Dependencies)	40.7	50.5	**84.2**	70.2	31.2	39.7	23.5	48.0	48.5
NN Shift-Reduce TAG Parser	60.4	75.8	68.4	79.8	53.8	45.4	44.7	59.4	61.2
Our Joint Method	**72.5**	**78.0**	81.1	**85.7**	56.3	47.1	**49.4**	**64.9**	**67.0**

Table 6: Parser accuracy on the unbounded dependency corpus. The results of the first five parsers are taken from Rimell et al. (2009) and Nivre et al. (2010). The Total and Avg columns indicate the percentage of correctly recovered dependencies out of all dependencies and the average of accuracy on the 7 constructions.

trained.[7] For RNR, rarity may be an issue as well as the limits of the TAG analysis of this construction. Nonetheless, we see that the rich structural representations that a TAG parser provides enables substantial improvements in the extraction of unbounded dependencies. In the future, we hope to evaluate state-of-the-art Stanford dependency parsers automatically.

6 Related Work

The two major classes of data-driven methods for dependency parsing are often called transition-based and graph-based parsing (Kübler et al., 2009). Transition-based parsers (e.g. MALT (Nivre, 2003)) learn to predict the next transition given the input and the parse history. Graph-based parsers (e.g. MST (McDonald et al., 2005)) are trained to directly assign scores to dependency graphs.

Empirical studies have shown that a transition-based parser and a graph-based parser yield similar overall performance across languages (McDonald and Nivre, 2011), but the two strands of data-driven parsing methods manifest the fundamental trade-off of parsing algorithms. The former prefers rich feature representations with parsing history over global training and exhaustive search, and the latter allows for global training and inference at the expense of limited feature representations (Kübler et al., 2009).

Recent neural network models for transition-based and graph-based parsing can be viewed as remedies for the aforementioned limitations. Andor et al. (2016) developed a transition-based parser using feed-forward neural networks that performs global training approximated by beam search. The globally normalized objective addresses the label bias problem and makes global training effective in the transition-based parsing setting. Kiperwasser and Goldberg (2016) incorporated a dynamic oracle (Goldberg and Nivre, 2013) in a BiLSTM transition-based parser that remedies global error propagation. Kiperwasser and Goldberg (2016) and Dozat and Manning (2017) proposed graph-based parsers that have access to rich feature representations obtained from BiLSTMs.

Previous work integrated CCG supertagging and parsing using belief propagation and dual decomposition approaches (Auli and Lopez, 2011). Nguyen et al. (2017) incorporated a graph-based dependency parser (Kiperwasser and Goldberg, 2016) with POS tagging. Our work followed these lines of effort and improved TAG parsing performance.

7 Conclusion and Future Work

In this work, we presented a state-of-the-art TAG supertagger, a parser, and a joint parser that performs POS tagging, supertagging, and parsing. The joint parser has the benefit of giving a full syntactic analysis of a sentence simultaneously. Furthermore, the joint parser achieved the best performance, an improvement of over 2.2 LAS points from the previous state-of-the-art. We have also seen that the joint parser yields state-of-the-art in textual entailment and unbounded dependency recovery tasks, and raised the possibility that TAG can provide useful structural analysis of sentences for other NLP tasks. We will explore more applications of our TAG parsers in future work.

[7] The substantially better performance of the C&C parser is in fact the result of additions that were made to the training data.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* .

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *ACL*. Association for Computational Linguistics, Berlin, Germany, pages 2442–2452. http://www.aclweb.org/anthology/P16-1231.

Michael Auli and Adam Lopez. 2011. A comparison of loopy belief propagation and dual decomposition for integrated CCG supertagging and parsing. In *ACL*. Association for Computational Linguistics, pages 470–480. http://www.aclweb.org/anthology/P11-1048.

Srinivas Bangalore, Pierre Boullier, Alexis Nasr, Owen Rambow, and Benoît Sagot. 2009. MICA: A probabilistic dependency parser based on tree insertion grammars (application note). In *NAACL-HLT (short)*. Association for Computational Linguistics, Boulder, Colorado, pages 185–188. http://www.aclweb.org/anthology/N/N09/N09-2047.

Srinivas Bangalore and Aravind K. Joshi. 1999. Supertagging: An Approach to Almost Parsing. *Computational Linguistics* 25:237–266.

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

Rich Caruana. 1997. Multitask learning. *Machine Learning* 28:41–75.

John Chen, Srinivas Bangalore, and K. Vijay-Shanker. 2005. Automated extraction of tree-adjoining grammars from treebanks. *Natural Language Engineering* 12(3):251–299.

John Chen and Owen Rambow. 2003. Use of deep linguistic features for the recognition and labeling of semantic arguments. In *EMNLP*. pages 41–48. http://aclanthology.coli.uni-saarland.de/pdf/W/W03/W03-1006.pdf.

Jason Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *TACL* 4:357–370. https://transacl.org/ojs/index.php/tacl/article/view/792.

Wonchang Chung, Suhas Siddhesh Mhatre, Alexis Nasr, Owen Rambow, and Srinivas Bangalore. 2016. Revisiting supertagging and parsing: How to use supertags in transition-based parsing. In *TAG+*. pages 85–92. http://www.aclweb.org/anthology/W16-3309.

Stephen Clark and James R Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics* 33(4):493–552. http://www.newdesign.aclweb.org/anthology-new/J/J07/J07-4004.pdf.

Timothy Dozat and Christopher Manning. 2017. Deep biaffine attention for neural dependency parsing. In *ICLR*.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the CoNLL 2017 shared task. In *CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics, Vancouver, Canada, pages 20–30. http://www.aclweb.org/anthology/K17-3002.

Dan Friedman, Jungo Kasai, R. Thomas McCoy, Robert Frank, Forrest Davis, and Owen Rambow. 2017. Linguistically rich vector representations of supertags for TAG parsing. In *TAG+*. Association for Computational Linguistics, Umeå, Sweden, pages 122–131. http://www.aclweb.org/anthology/W17-6213.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *NIPS*.

Yoav Goldberg and Joakim Nivre. 2013. Training deterministic parsers with non-deterministic oracles. *TACL* 1:403–414. https://www.aclweb.org/anthology/Q/Q13/Q13-1033.pdf.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks* 18(5):602–610.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics* 33(3):355–396.

Aravind K. Joshi and Yves Schabes. 1997. Tree-adjoining grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages, Volume 3: Beyond Words*, Springer, New York, pages 69–124.

Jungo Kasai, Robert Frank, Tom McCoy, Owen Rambow, and Alexis Nasr. 2017. TAG parsing with neural networks and vector representations of supertags. In *EMNLP*. Association for Computational Linguistics, Copenhagen, Denmark, pages 1713–1723. https://www.aclweb.org/anthology/D17-1180.

Diederik P. Kingma and Jimmy Lei Ba. 2015. ADAM: A Method for Stochastic Optimization. In *ICLR*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *TACL* 4:313–327. https://transacl.org/ojs/index.php/tacl/article/view/885.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. *Dependency Parsing*. Morgan & Claypool Publishers.

Mike Lewis, Kenton Lee, and Luke Zettlemoyer. 2016. LSTM CCG parsing. In *HLT-NAACL*. pages 221–231.

Elisabeth Lien. 2014. Using minimal recursion semantics for entailment recognition. In *Proceedings of the Student Research Workshop at EACL*. Gothenburg, Sweden, page 7684.

Wang Ling, Tiago Luís, Luís Marujo, Rámon Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *EMNLP*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *ACL*. Association for Computational Linguistics, Berlin, Germany, pages 1064–1074. http://www.aclweb.org/anthology/P16-1101.

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

Ryan McDonald and Joakim Nivre. 2011. Analyzing and integrating dependency parsers. *Computational Linguistics* 37(1):197–230. http://dx.doi.org/10.1162/coli_a_00039.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *EMNLP*. Association for Computational Linguistics, Vancouver, British Columbia, Canada, pages 523–530. http://www.aclweb.org/anthology/H/H05/H05-1066.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *NIPS*, Curran Associates, Inc., pages 3111–3119.

Dominick Ng, James W.D. Constable, Matthew Honnibal, and James R. Curran. 2010. SCHWA: PETE using CCG dependencies with the C&C parser. In *SemEval*. page 313316.

Dat Quoc Nguyen, Mark Dras, and Mark Johnson. 2017. A novel neural network model for joint pos tagging and graph-based dependency parsing. In *CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics, pages 134–142. https://doi.org/10.18653/v1/K17-3014.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *IWPT*.

Joakim Nivre, Laura Rimell, Ryan McDonald, and Carlos Gómez Rodríguez. 2010. Evaluation of dependency parsers on unbounded dependencies. In *COLING*. Coling 2010 Organizing Committee, Beijing, China, pages 833–841. http://www.aclweb.org/anthology/C10-1094.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*. pages 1532–1543.

Owen Rambow and Aravind Joshi. 1994. A formal look at dependency grammars and phrase-structure grammars, with special consideration of word-order phenomena. In Leo Wanner, editor, *Recent Trends in Meaning-Text Theory*, Amsterdam and Philadelphia, pages 167–190.

Laura Rimell and Stephen Clark. 2010. Cambridge: Parser evaluation using textual entailment by grammatical relation comparison. In *SemEval*. pages 268–271.

Laura Rimell, Stephen Clark, and Mark Steedman. 2009. Unbounded dependency recovery for parser evaluation. In *EMNLP*. Singapore, page 813821.

Cicero D. Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In Tony Jebara and Eric P. Xing, editors, *ICML*. JMLR Workshop and Conference Proceedings, pages 1818–1826. http://jmlr.org/proceedings/papers/v32/santos14.pdf.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387* .

Mark Steedman and Jason Baldridge. 2011. Combinatory categorial grammar. In Robert Borsley and Kersti Börjars, editors, *Non-Transformational Syntax: Formal and Explicit Models of Grammar*, Wiley-Blackwell.

Ashish Vaswani, Yonatan Bisk, Kenji Sagae, and Ryan Musa. 2016. Supertagging with LSTMs. In *NAACL*. Association for Computational Linguistics, San Diego, California, pages 232–237. http://www.aclweb.org/anthology/N16-1027.

Pauli Xu, Robert Frank, Jungo Kasai, and Owen Rambow. 2017. TAG parser evaluation using textual entailments. In *TAG+*. Association for Computational Linguistics, Umeå, Sweden, pages 132–141. http://www.aclweb.org/anthology/W17-6214.

Wenduan Xu. 2016. LSTM shift-reduce CCG parsing. In *EMNLP*. Association for Computational Linguistics, Austin, Texas, pages 1754–1764. https://aclweb.org/anthology/D16-1181.

Wenduan Xu, Michael Auli, and Stephen Clark. 2015. CCG supertagging with a recurrent neural network. In *ACL*. Association for Computational Linguistics,

Beijing, China, pages 250–255. http://www.aclweb.org/anthology/P15-2041.

Deniz Yuret, Aydin Han, and Zehra Turgut. 2010. SemEval-2010 task 12: Parser evaluation using textual entailments. In *SemEval*. Association for Computational Linguistics, Uppsala, Sweden, pages 51–56. http://www.aclweb.org/anthology/S10-1009.

A Transformations for Unbounded Dependency Recovery Corpus

For automatic evaluation on the unbounded dependency recovery corpus (UDR, Rimell et al. (2009)), we run simple conversion of dependency labels in UDR to those in our TAG grammar (See Table 7) with a couple of exceptions.

- Change arcs from verbs to wh-adverbs as in "where is the city located?" to adjunction.
- Reflect causative-inchoative alternation in the subject embedded construction. Concretely, change the role of "door" in "hold the door shut" from the subject to the object of "shut."

We then transform TAG dependency trees. Finally, we simply check if the resulting dependency graphs contain target labeled arcs given in the dataset.

Below is a full description of transformations. This set of structural transformations is applied in the order in which we will present it, so that the output of previous transformations can feed subsequent ones. In the following, we denote an arc pointing from node B to node A with label C as (A, B, C) where A and B are called the child (dependent) and the parent (head) in the relation.

A.1 Transformations from PETE

We apply three types of transformation from Xu et al. (2017) to interpret the TAG parses.

Relative Clauses When an elementary tree of a relative clause adjoins into a noun, we add a reverse arc with the label reflecting the type of the relative clause elementary tree. For a subject relative, we add a 0-labeled arc, for an object relative, we add a 1-labeled arc, and so forth.

UDR Labels	TAG Labels
nsubj, cop	0
dobj, pobj, obj2, nsubjpass	1
others (advmod etc)	ADJ

Table 7: UD to TAG label conversion.

Sentential Complements Sentential complementation in TAG derivations can be analyzed via either adjoining the higher clause into the embedded clause (necessarily so in cases of long-distance extraction from the embedded clause) or substituting the embedded clause in the higher clause. In order to normalize this divergence, for an adjunction arc involving a predicative auxiliary elementary tree (supertag), we add a reverse arc involving the 1 relation (sentential complements).

A.2 Coordination

We roughly follow the method presented in Xu et al. (2017) with extensions. Under the TAG analysis, VP coordination involves a VP-recursive auxiliary tree headed by the coordinator that includes a VP substitution node (for the second conjunct) with label 1. In order to allow the first clauses subject argument (as well as modal verbs and negations) to be shared by the second verb, we add the relevant relations to the second verb. In addition, we analyze sentential coordination cases. Sentence coordination in our TAG grammar usually happens between two complete sentences and no modifiers or arguments are shared, and therefore it can be analyzed via substituting a sentence int the coordinator with label 1. However, when sentential coordination happens between two relative clause modifiers, our TAG grammar analyzes the second clause as a complete sentence, meaning that we need to recover the extracted argument by consulting the property of the first clause. Furthermore, the deep syntactic role of the extracted argument can be different in the two relative clauses. For instance, in the sentence, "... the same stump which had impaled the car of many a guest in the past thirty years and which he refused to have removed," we need to recover an arc from removed to stump with label 1 whereas the arc from impaled to stump has label 0. To resolve this issue, when there is coordination of two relative clause modifiers, we add an edge from the head of the second clause to the modified noun with the same label as the label that under which the relative pronoun is attached to the head.

A.3 Resolving Differences in Dependency Representations

Small Clauses The UDR corpus has inconsistency with regards to small clauses. UDR gives an analysis that a small clause contains a subject and a complement as in (nsubj, guy, liar) in

"the guy who I call a liar." in the subject embedded constructions. However, in the object question and object free relative constructions, a small clause is analyzed as two arguments of the verb. For instance, UDR specifies (what, adopted, dobj) in "we adopted what I would term pseudo-capitalism." To solve this problem we add an arc from the head of the matrix clause to the subject in a small clause with label 1.

Co-anchors In our TAG grammar, Co-anchor attachment represents the substitution into a node that is construed as a co-head of an elementary tree. For instance, "for" is deemed as a co-anchor to "hope" in the sentence *"that is exactly what I'm hoping for* (Figure 4). In this case, UDR would pick the relation (what, hope, pobj). Therefore, when there is a co-anchor to a head tree, we add all arcs that involve the head tree to the co-anchor tree.

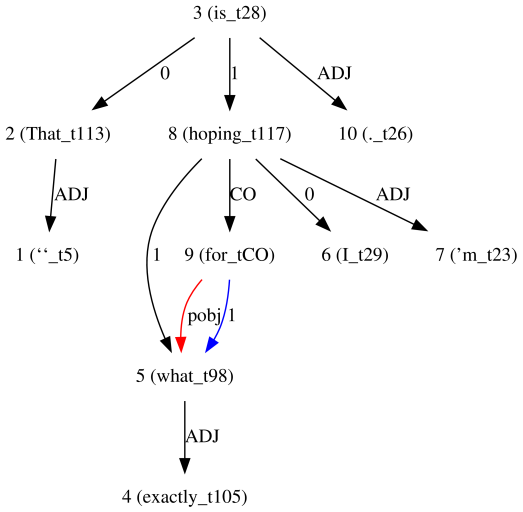


Figure 4: Co-anchor case from a sentence *"that is exactly what I'm hoping for.* The UDR gives the red arc (what, for, pobj). The blue arc (what, for, 1) is obtained from (what, hope, 1).

Wh-determiners and Wh-adverbs Our TAG grammar analyzes a wh-determiner via adjoining the noun into the wh-determiner (Figure 5). This is also true for cases where a wh-adverb is followed by an adjective and a noun as in ***how many battles*** *did she win?* In contrast, UDR corpus gives an analysis that the noun is the head of the constituent. In order to resolve this discrepancy, when a word adjoins into a wh-word,[8] we pick all arcs with the wh-word as the child and add the arcs obtained from such arcs by replacing the wh-word child by the word adjoining into the wh-word.

Copulas A copula is usually treated as a dependent to the predicate both in our TAG grammar (adjunction) and UDR. However, we found two situations where they differ from each other. First, when wh-extraction happens on the complement, as in "obviously there has been no agreement on what American conservatism is, or rather, what it should be," the TAG grammar analyzes it via substituting the wh-word ("what") into the copula ("is"). To reconcile this disagreement between the TAG grammar and UDR, when substitution happens into a *be* verb, we add the substitution into

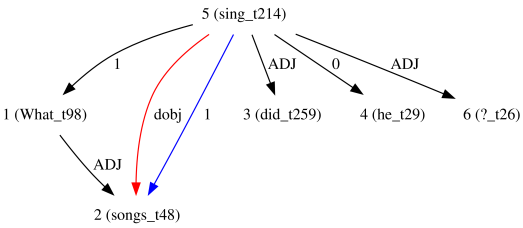


Figure 5: Wh-determiner case from a sentence *What songs did he sing?* The UDR gives the red arc (songs, sing, dobj). The blue arc (song, sing, 1) is obtained from (what, sing, 1) and (songs, what, ADJ).

[8] We considered imposing a more strict condition that the word adjoining into the wh-word is a noun, but we found cases that this method fails to cover; for example, UDR gives (dobj, get, much) for a sentence "opinion is mixed on how **much** of a boost the overall stock market would **get** even if dividend growth continues at double-digit levels."

the copula.[9] Second, UDR treats non-*be* copulas differently than *be* verbs. An example is the UDR relation (those, stayed, nsubj) "in the other hemisphere it is growing colder and nymphs, those who stayed alive through the summer, are being brought into nests for quickening and more growing" where our parser yields (those, alive, 0). For this reason, when a lemma of a verb is a non-*be* copula,[10] we add arcs involving the word to the copula adjoining into the copula.

PP attachment with multiple noun candidates We observed that PP attachment with multiple noun candidates is often at stake in UDR.[11] For instance, UDR provides (part, had, nsubj) and (several, tried, nsubj) for the sentences "... there is no part of the earth that has not had them" and "there were several on the Council who tried to live like Christians" while the TAG parser outputs (earth, had, nsubj) and (Council, tried, nsubj) respectively. While we count these cases as "wrong" since they manifest certain disambiguation (though not purely unbounded dependency recovery), we ignore superficial (conventional) differences in head selection. In our TAG grammar "a lot of people" would be headed by "lot" whereas UDR would recognize "people" as the head. Hence, when "lot/lots/kind/kinds/none of" occurs, we add all arcs with "lot/lots/kind/kinds/none" to the head of the phrase that is the object of "of."

Modals In the UDR corpus, a modal depends on an auxiliary verb following the modal, if there is one. For example, "Rosie reinvented this man, who may or may not have known about his child" is given the relation (may, have, aux). In the TAG grammar, both "may" and "have" adjoin into "known." Therefore, when the head of a modal has another child with adjunction, we add an arc from the child to the modal.

Existential *there* UDR gives the "cop" relation between an existential there and the *be* verb. For example, it gives (be, legislation, cop) in "... on how much social legislation there should be." On the other hand, our TAG grammar analyzes that "there" is attached to "be" with label 0.[12] To resolve this issue, for arcs that point into an existential there with label 0, we add a reverse edge with label 0.

Determiner modifying a sentence Finally, when a determiner followed by an adjective modifies a sentence via adjunction in our TAG as in "the more highly placed they are – that is, the more they know – the more concerned they have become," we add an edge from the verb to the adjective with label 1.

[9] We use the nltk lemmatizer (Bird et al., 2009) to identify *be* verbs.

[10] We chose " stay," "become," "seem," and "remain."

[11] This is indeed one of the problems with UDR. Performance on UDR is not purely reflective of unbounded dependency recovery.

[12] Usually, "there" is attached to the noun, not the *be* verb, but in this case, extraction is happening on the noun, so the *be* verb becomes the head. See the discussion on copulas above.

Colorless green recurrent networks dream hierarchically

Kristina Gulordava*
Department of Linguistics
University of Geneva
kristina.gulordava@unige.ch

Piotr Bojanowski
Facebook AI Research
Paris
bojanowski@fb.com

Edouard Grave
Facebook AI Research
New York
egrave@fb.com

Tal Linzen
Department of Cognitive Science
Johns Hopkins University
tal.linzen@jhu.edu

Marco Baroni
Facebook AI Research
Paris
mbaroni@fb.com

Abstract

Recurrent neural networks (RNNs) have achieved impressive results in a variety of linguistic processing tasks, suggesting that they can induce non-trivial properties of language. We investigate here to what extent RNNs learn to track abstract hierarchical syntactic structure. We test whether RNNs trained with a generic language modeling objective in four languages (Italian, English, Hebrew, Russian) can predict long-distance number agreement in various constructions. We include in our evaluation nonsensical sentences where RNNs cannot rely on semantic or lexical cues ("The colorless green ideas I ate with the chair sleep furiously"), and, for Italian, we compare model performance to human intuitions. Our language-model-trained RNNs make reliable predictions about long-distance agreement, and do not lag much behind human performance. We thus bring support to the hypothesis that RNNs are not just shallow-pattern extractors, but they also acquire deeper grammatical competence.

1 Introduction

Recurrent neural networks (RNNs; Elman, 1990) are general sequence processing devices that do not explicitly encode the hierarchical structure that is thought to be essential to natural language (Everaert et al., 2015). Early work using artificial languages showed that they may nevertheless be able to approximate context-free languages (Elman, 1991). More recently, RNNs have achieved impressive results in large-scale tasks such as language modeling for speech recognition and machine translation, and are by now standard tools for sequential natural language tasks (e.g., Mikolov et al., 2010; Graves, 2012; Wu et al., 2016). This suggests that RNNs may learn to track grammatical structure even when trained on noisier natural data. The conjecture is supported by the success of RNNs as feature extractors for syntactic parsing (e.g., Cross and Huang, 2016; Kiperwasser and Goldberg, 2016; Zhang et al., 2017).

Linzen et al. (2016) directly evaluated the extent to which RNNs can approximate hierarchical structure in corpus-extracted natural language data. They tested whether RNNs can learn to predict English subject-verb agreement, a task thought to require hierarchical structure in the general case ("the girl the boys like… is or are?"). Their experiments confirmed that RNNs can, in principle, handle such constructions. However, in their study RNNs could only succeed when provided with explicit supervision on the target task. Linzen and colleagues argued that the unsupervised language modeling objective is not sufficient for RNNs to induce the syntactic knowledge necessary to cope with long-distance agreement.

The current paper reevaluates these conclusions. We strengthen the evaluation paradigm of Linzen and colleagues in several ways. Most importantly, their analysis did not rule out the possibility that RNNs might be relying on *semantic* or *collocational/frequency-based* information, rather than purely on syntactic structure. In "dogs in the neighbourhood often bark", an RNN might get the right agreement by encoding information

*The work was conducted during the internship at Facebook AI Research, Paris.

Proceedings of NAACL-HLT 2018, pages 1195–1205
New Orleans, Louisiana, June 1 - 6, 2018.

about what typically barks (dogs, not neighbourhoods), without relying on more abstract structural cues. In a follow-up study to Linzen and colleagues', Bernardy and Lappin (2017) observed that RNNs are better at long-distance agreement when they construct rich lexical representations of words, which suggests effects of this sort might indeed be at play.

We introduce a method to probe the syntactic abilities of RNNs that abstracts away from potential lexical, semantic and frequency-based confounds. Inspired by Chomsky's (1957) insight that "grammaticalness cannot be identified with meaningfulness" (p. 106), we test long-distance agreement both in standard corpus-extracted examples and in comparable nonce sentences that are grammatical but completely meaningless, e.g., (paraphrasing Chomsky): "The colorless green ideas I ate with the chair sleep furiously".

We extend the previous work in three additional ways. First, alongside English, which has few morphological cues to agreement, we examine Italian, Hebrew and Russian, which have richer morphological systems. Second, we go beyond subject-verb agreement and develop an automated method to harvest a variety of long-distance number agreement constructions from treebanks. Finally, for Italian, we collect human judgments for the tested sentences, providing an important comparison point for RNN performance.[1]

We focus on the more interesting unsupervised setup, where RNNs are trained to perform generic, large-scale language modeling (LM): they are not given explicit evidence, at training time, that they must focus on long-distance agreement, but they are rather required to track a multitude of cues that might help with word prediction in general.

Our results are encouraging. RNNs trained with a LM objective solve the long-distance agreement problem well, even on nonce sentences. The pattern is consistent across languages, and, crucially, not far from human performance in Italian. Moreover, RNN performance on language modeling (measured in terms of perplexity) is a good predictor of long-distance agreement accuracy. This suggests that the ability to capture structural generalizations is an important aspect of what makes the best RNN architectures so good at language modeling. Since our positive results contradict, to some extent, those of Linzen et al. (2016), we also replicate their relevant experiment using our best RNN (an LSTM). We outperform their models, suggesting that a careful architecture/hyperparameter search is crucial to obtain RNNs that are not only good at language modeling, but able to extract syntactic generalizations.

[1]The code to reproduce our experiments and the data used for training and evaluation, including the human judgments in Italian, can be found at https://github.com/facebookresearch/colorlessgreenRNNs.

2 Constructing a long-distance agreement benchmark

Overview. We construct our number agreement test sets as follows. **Original** sentences are automatically extracted from a dependency treebank. They are then converted into **nonce** sentences by substituting all content words with random words with the same morphology, resulting in grammatical but nonsensical sequences. An LM is evaluated on its predictions for the target (second) word in the dependency, in both the original and nonce sentences.

Long-distance agreement constructions. Agreement relations, such as subject-verb agreement in English, are an ideal test bed for the syntactic abilities of LMs, because the form of the second item (the target) is predictable from the first item (the cue). Crucially, the cue and the target are linked by a *structural* relation, where linear order in the word sequence does not matter (Everaert et al., 2015). Consider the following subject-verb agreement examples: "the girl thinks…", "the girl [you met] thinks…", "the girl [you met yesterday] thinks…", "the girl [you met yesterday through her friends] thinks…". In all these cases, the number of the main verb "thinks" is determined by its subject ("girl"), and this relation depends on the syntactic structure of the sentence, not on the linear sequence of words. As the last sentence shows, the word directly preceding the verb can even be a noun with the opposite number ("friends"), but this does not influence the structurally-determined form of the verb.

When the cue and the target are adjacent ("the girl thinks…"), an LM can predict the target without access to syntactic structure: it can simply extract the relevant morphosyntactic features of words (e.g., number) and record the co-occurrence frequencies of patterns such as N_{Plur} V_{Plur} (Mikolov et al., 2013). Thus, we focus here on *long-distance* agreement, where an arbitrary num-

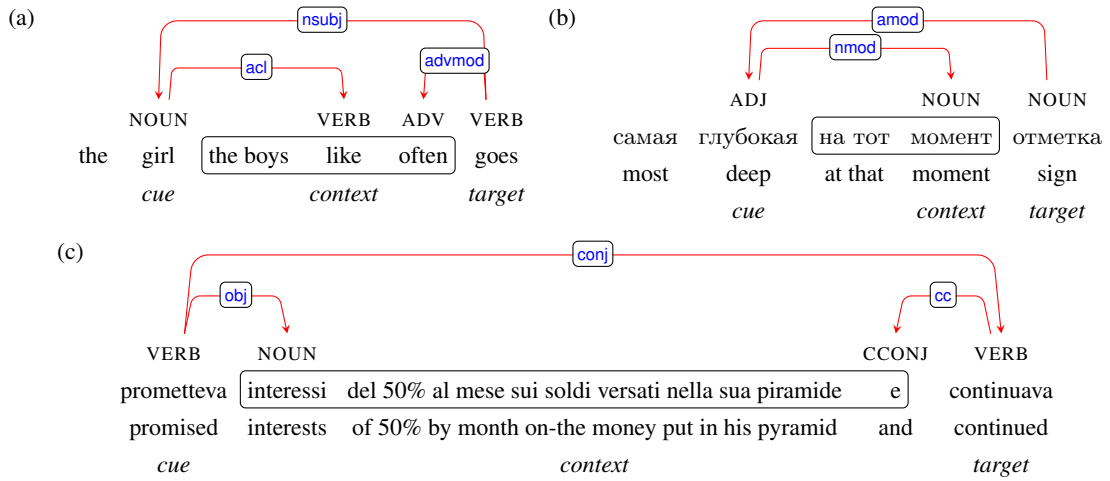


Figure 1: Example agreement constructions defined by a dependency and the separating context, in (a) English, (b) Russian and (c) Italian.

ber of words can occur between the elements of the agreement relation. We limit ourselves to *number* agreement (plural or singular), as it is the only overt agreement feature shared by all of the languages we study.

Identifying candidate constructions. We started by collecting pairs of part-of-speech (POS) tags connected by a dependency arc. Independently of which element is the head of the relation, we refer to the first item as the **cue** and to the second as the **target**. We additionally refer to the POS sequence characterizing the entire pattern as a **construction**, and to the elements in the middle as **context**.

For each candidate construction, we collected all of the contexts in the corpus that intervene between the cue and the target (we define contexts as the sequence of POS tags of the top-level nodes in the dependency subtrees). For example, for the English subject-verb agreement construction shown in Fig. 1a, the context is defined by VERB (head of the relative clause) and ADV (adverbial modifier of the target verb), which together dominate the sequence "the boys like often". For the Russian adjective-noun agreement construction in Fig. 1b, the context is NOUN, because in the dependency grammar we use the noun "moment" is the head of the prepositional phrase "at that moment", which modifies the adjective "deep". The candidate agreement pair and the context form a construction, which is characterized by a sequence of POS tags, e.g., NOUN VERB ADV VERB or VERB NOUN CCONJ VERB (Fig. 1c).

Our constructions do not necessarily correspond to standard syntactic structures. The English subject-verb agreement construction NOUN VERB VERB, for example, matches both object and subject relative clause contexts, e.g., "girl the boys like is" and "girls who stayed at home were". Conversely, standard syntactic structures might be split between different constructions, e.g., relative clause contexts occur in both NOUN VERB VERB and NOUN VERB ADV VERB constructions (the latter is illustrated by the English example in Fig. 1a).

Construction contexts can contain a variable numbers of words. Since we are interested in challenging cases, we only considered cases in which at least three tokens intervened between the cue and the target.

Excluding non-agreement constructions. In the next step, we excluded constructions in which the candidate cue and target did not agree in number in all of the instances of the construction in the treebank (if both the cue and the target were morphologically annotated for number). This step retained English subject-verb constructions, for example, but excluded verb-object constructions, since any form of a verb can appear both with singular and plural objects. To focus on robust agreement patterns, we only kept constructions with at least 10 instances of both plural and singular agreement.

When applied to the treebanks we used (see Section 3), this step resulted in between two (English) and 21 (Russian) constructions per lan-

guage. English has the poorest morphology and consequently the lowest number of patterns with identifiable morphological agreement. Only the VP-conjunction construction (Fig. 1c) was identified in all four languages. Subject-verb agreement constructions were extracted in all languages but Russian; Russian has relatively flexible word order and a noun dependent preceding a head verb is not necessarily its subject. The full list of extracted constructions in English and Italian is given in Tables 2 and 3, respectively. For the other languages, see the Supplementary Material (SM).[2]

Original sentence test set. Our "original" sentence test set included all sentences from each construction where all words from the cue and up to and including the target occurred in the LM vocabulary (Section 3), and where the singular/plural counterpart of the target occurred in the treebank and in the language model vocabulary (this is required by the evaluation procedure outlined below). The total counts of constructions and original sentences in our test sets are provided in Table 1. The average number of context words separating the cue and the target ranged from 3.6 (Hebrew) to 4.5 (Italian).

Generating nonce sentences. We generated nine nonce variants of each original sentence as follows. Each content word (noun, verb, adjective, proper noun, numeral, adverb) in the sentence was substituted by another random content word from the treebank with matching POS and morphological features. To avoid forms that are ambiguous between several POS, which are particularly frequent in English (e.g., plural noun and singular verb forms), we excluded the forms that appeared with a different POS more than 10% of the time in the treebank. Function words (determiners, pronouns, adpositions, particles) and punctuation were left intact. For example, we generated the nonce (1b) from the original sentence (1a):

(1) a. It presents the case for marriage equality and states...
b. It stays the shuttle for honesty insurance and finds...

Note that our generation procedure is based on morphological features and does not guarantee that argument structure constraints are respected (e.g., "it stays the shuttle" in (1b)).

Evaluation procedure. For each sentence in our test set, we retrieved from our treebank the form that is identical to the agreement target in all morphological features except number (e.g., "finds" instead of "find" in (1b)). Given a sentence with prefix p up to and excluding the target, we then compute the probabilities $P(t_1|p)$ and $P(t_2|p)$ for the singular and plural variants of the target, t_1 and t_2, based on the language model. Following Linzen et al. (2016), we say that the model identified the correct target if it assigned a higher probability to the form with the correct number. In (1b), for example, the model should assign a higher probability to "finds" than "find".[3]

3 Experimental setup

Treebanks. We extracted our test sets from the Italian, English, Hebrew and Russian Universal Dependency treebanks (UD, v2.0, Nivre et al., 2016). The English and Hebrew treebanks were post-processed to obtain a richer morphological annotation at the word level (see SM for details).

LM training data. Training data for Italian, English and Russian were extracted from the respective Wikipedias. We downloaded recent dumps, extracted the raw text from them using WikiExtractor[4] and tokenized it with TreeTagger (Schmid, 1995). We also used the TreeTagger lemma annotation to filter out sentences with more than 5% unknown words. For Hebrew, we used the preprocessed Wikipedia corpus made available by Yoav Goldberg.[5] We extracted 90M token subsets for each language, shuffled them by sentence and split them into training and validation sets (8-to-1 proportion). For LM training, we included the 50K most frequent words in each corpus in the vocabulary, replacing the other tokens with the UNK symbol. The validation set perplexity values we report below exclude unknown tokens.

RNN language models. We experimented with simple RNNs (sRNNs, Elman, 1990), and their most successful variant, long-short term memory models (LSTMs, Hochreiter and Schmidhu-

[2] The SM is available as a standalone file on the project's public repository.

[3] Obviously, in the nonce cases, the LMs never assigned the highest overall probability to either of the two candidates. Qualitatively, in such cases LMs assigned the largest absolute probabilities to plausible frequent words.

[4] https://github.com/attardi/wikiextractor

[5] http://u.cs.biu.ac.il/~yogo/hebwiki/

ber, 1997). We use the PyTorch RNN implementation.[6] We trained the models with two hidden layer dimensionalities (650 and 200 units), and a range of batch sizes, learning rates and dropout rates. See SM for details on hyperparameter tuning. In general, a larger hidden layer size was the best predictor of lower perplexity. Given that our LSTMs outperformed our sRNNs, our discussion of the results will focus on the former; we will use the terms LSTM and RNN interchangeably.[7]

Baselines. We consider three baselines: first, a **unigram** baseline, which picks the most frequent form in the training corpus out of the two candidate target forms (singular or plural); second, a 5-gram model with Kneser-Ney smoothing (**KN**, Kneser and Ney, 1995) trained using the IRSTLM package (Federico et al., 2008) and queried using KenLM (Heafield, 2011); and third, a **5-gram LSTM**, which only had access to windows of five tokens (Chelba et al., 2017). Compared to KN, the 5-gram LSTM can generalize to unseen n-grams thanks to its embedding layer and recurrent connections. However, it cannot discover long-distance dependency patterns that span more than five words. See SM for details on the hyperparameters of this baseline.

Human experiment in Italian. We presented the full Italian test set (119 original and 1071 nonce sentences) to human subjects through the Amazon Mechanical Turk interface.[8] We picked Italian because, being morphologically richer, it features more varied long-distance constructions than English. Subjects were requested to be native Italian speakers. They were presented with a sentence up to and excluding the target. The singular and plural forms of the target were presented below the sentence (in random order), and subjects were asked to select the more plausible form.

To prevent long-distance agreement patterns from being too salient, we mixed the test set with the same number of filler sentences. We started from original fillers, which were random treebank-extracted sentences up to a content word in singular or plural form. We then generated nonce fillers from the original ones using the procedure outlined in Section 2. A control subset of 688 fillers was manually selected by a linguistically-trained Italian native speaker as unambiguous cases. To make sure we were only using data from native (or at least highly proficient) Italian speakers, we filtered out the responses of subjects who chose the wrong target in more than 20% of the fillers.

We collected on average 9.5 judgments for each item (minimum 5 judgments). To account for the variable number of judgments across sentences, accuracy rates were first calculated within each sentence and then averaged across sentences.

[6] https://github.com/pytorch/examples/tree/master/word_language_model

[7] Detailed results for sRNNs can be found in the SM.

[8] https://www.mturk.com/

	IT	EN	HE	RU
#constructions	8	2	18	21
#original	119	41	373	442
Unigram				
Original	54.6	65.9	67.8	60.2
Nonce	54.1	42.5	63.1	54.0
5-gram KN				
Original	63.9	63.4	72.1	73.5
Nonce	52.8	43.4	61.7	56.8
Perplexity	147.8	168.9	122.0	166.6
5-gram LSTM				
Original	81.8 ±3.2	70.2 ±5.8	90.9 ±1.2	91.5 ±0.4
Nonce	78.0 ±1.3	58.2 ±2.1	77.5 ±0.8	85.7 ±0.7
Perplexity	62.6 ±0.2	71.6 ±0.3	59.9 ±0.2	61.1 ±0.4
LSTM				
Original	92.1 ±1.6	81.0 ±2.0	94.7 ±0.4	96.1 ±0.7
Nonce	85.5 ±0.7	74.1 ±1.6	80.8 ±0.8	88.8 ±0.9
Perplexity	45.2 ±0.3	52.1 ±0.3	42.5 ±0.2	48.9 ±0.6

Table 1: Experimental results for all languages averaged across the five best models in terms of perplexity on the validation set. Original/Nonce rows report percentage accuracy, and the numbers in small print represent standard deviation within the five best models.

4 Results

The overall results are reported in Table 1. We report results averaged across the five models with the lowest validation perplexity, as well as standard deviations across these models. In summary,

		N V V	V NP conj V
Italian	Original	$93.3_{\pm 4.1}$	$83.3_{\pm 10.4}$
	Nonce	$92.5_{\pm 2.1}$	$78.5_{\pm 1.7}$
English	Original	$89.6_{\pm 3.6}$	$67.5_{\pm 5.2}$
	Nonce	$68.7_{\pm 0.9}$	$82.5_{\pm 4.8}$
Hebrew	Original	$86.7_{\pm 9.3}$	$83.3_{\pm 5.9}$
	Nonce	$65.7_{\pm 4.1}$	$83.1_{\pm 2.8}$
Russian	Original	-	$95.2_{\pm 1.9}$
	Nonce	-	$86.7_{\pm 1.6}$

Table 2: LSTM accuracy in the constructions N V V (subject-verb agreement with an intervening embedded clause) and V NP conj V (agreement between conjoined verbs separated by a complement of the first verb).

the LSTM clearly outperformed the other LMs. Rather surprisingly, its performance on nonce sentences was only moderately lower than on original ones; in Italian this gap was only 6.6%.

The KN LM performed poorly; its accuracy on nonce sentences was comparable to that of the unigram baseline. This confirms that the number of the target in nonce sentences cannot be captured by shallow n-gram patterns. The 5-gram LSTM model greatly improved over the KN baseline; its accuracy dropped only modestly between the original and nonce sentences, demonstrating its syntactic generalization ability. Still, the results are substantially below those of the LSTM with unlimited history. This confirms that our test set contains hard long-distance agreement dependencies, and, more importantly, that the more general LSTM model can exploit broader contexts to learn about and track long-distance syntactic relations.

The increase in accuracy scores across the three LMs (KN, 5-gram LSTM and unbounded-context LSTM) correlates well with their validation perplexities in the language modeling task. We also found a strong correlation between agreement accuracy and validation perplexity across all the LSTM variants we explored in the hyperparameter search (68 models per language), with Pearson correlation coefficients ranging from $r = -0.55$ in Hebrew to $r = -0.78$ in English ($p < 0.001$ in all languages). This suggests that acquiring abstract syntactic competence is a natural component of the skills that improve the generic language modeling performance of RNNs.

Differences across languages. English was by far the hardest language. We conjecture that this is due to its poorer morphology and higher POS ambiguity, which might not encourage a generic language model to track abstract syntactic configurations. There is an alternative hypothesis, however. We only extracted two constructions for English, both of which can be argued to be linguistically complex: subject-verb agreement with an intervening embedded clause, and agreement between two conjoined verbs with a nominal complement intervening between the verbs. Yet the results on these two constructions, comparable across languages (with the exception of the subject-verb construction in Russian, which was not extracted), confirm that English is particularly hard (Table 2). A qualitative inspection suggests that the low accuracy in the verb conjunction case (67.5%) is due to ambiguous sentences such as "if you have any questions or need/needs", where the target can be re-interpreted as a noun that is acceptable in the relevant context.[9]

In languages such as Italian and Russian, which have richer morphology and less ambiguity at the part-of-speech level than English, the LSTMs show much better accuracy and a smaller gap between original and nonce sentences. These results are in line with human experimental studies that found that richer morphology correlates with fewer agreement attraction errors (Lorimor et al., 2008). The pattern of accuracy rates in general, and the accuracy for the shared V NP conj V construction in particular, are consistent with the finding that Russian is less prone to human attraction errors than Italian, which, in turn, shows less errors than English.

The largest drop in accuracy between original and nonce sentences occurred in Hebrew. A qualitative analysis of the data in this language suggests that this might be due to the numerical prevalence of a few constructions that can have multiple alternative readings, some of which can license the incorrect number. We leave a more systematic analysis of this finding for future research.

Human results. To put our results in context and provide a reasonable upper bound on the LM performance, in particular for nonce sentences, we next compare model performance to that of human

[9]The nonce condition has higher accuracy because our substitution procedure in English tends to reduce POS ambiguity.

Construction	#original	Original		Nonce	
		Subjects	LSTM	Subjects	LSTM
DET [AdjP] NOUN	14	98.7	$98.6_{\pm 3.2}$	98.1	$91.7_{\pm 0.4}$
NOUN [RelC / PartP] clitic VERB	6	93.1	$100_{\pm 0.0}$	95.4	$97.8_{\pm 0.8}$
NOUN [RelC / PartP] VERB	27	97.0	$93.3_{\pm 4.1}$	92.3	$92.5_{\pm 2.1}$
ADJ [conjoined ADJs] ADJ	13	98.5	$100_{\pm 0.0}$	98.0	$98.1_{\pm 1.1}$
NOUN [AdjP] relpron VERB	10	95.9	$98.0_{\pm 4.5}$	89.5	$84.0_{\pm 3.3}$
NOUN [PP] ADVERB ADJ	13	91.5	$98.5_{\pm 3.4}$	79.4	$76.9_{\pm 1.4}$
NOUN [PP] VERB (participial)	18	87.1	$77.8_{\pm 3.9}$	73.4	$71.1_{\pm 3.3}$
VERB [NP] CONJ VERB	18	94.0	$83.3_{\pm 10.4}$	86.8	$78.5_{\pm 1.7}$
(Micro) average		94.5	$92.1_{\pm 1.6}$	88.4	$85.5_{\pm 0.7}$

Table 3: Subject and LSTM accuracy on the Italian test set, by construction and averaged.

subjects in Italian.

Table 3 reports the accuracy of the LSTMs and the human subjects, grouped by construction.[10] There was a consistent gap in human accuracy between original and nonce sentences (6.1% on average). The gap in accuracy between the human subjects and the model was quite small, and was similar for original and nonce sentences (2.4% and 2.9%, respectively).

In some of the harder constructions, particularly subject-verb agreement with an embedded clause, the accuracy of the LSTMs on nonce sentences was comparable to human accuracy ($92.5_{\pm 2.1}$ vs. 92.3%). To test whether the human subjects and the models struggle with the same sentences, we computed for each sentence (1) the number of times the human subjects selected the correct form of the target minus the number of times they selected the incorrect form, and (2) the difference in model log probability between the correct and incorrect form. The Spearman correlation between these quantities was significant, for both original ($p < 0.05$) and nonce sentences ($p < 0.001$). This indicates that humans were more likely to select the correct form in sentences in which the models were more confident in a correct prediction.

Moreover, some of the easiest and hardest constructions are the same for the human subjects and the models. In the easy constructions DET [AdjP] NOUN[11] and ADJ [conjoined ADJs] ADJ, one or more adjectives that intervene between the cue and the target agree in number with the target, providing shorter-distance evidence about its correct number. For example, in

(2) un film inutile ma almeno festivo e
a movie useless but at.least festive and
giovanile
youthful
"A useless but at least festive and youthful movie"

the adjective "festivo" is marked for singular number, offering a nearer reference for the target number than the cue "inutile". At the other end, NOUN [PP] VERB (participial) and NOUN [PP] ADVERB ADJ are difficult. Particularly in the nonce condition, where semantics is unhelpful or even misleading, the target could easily be interpreted as a modifier of the noun embedded in the preceding prepositional phrase. For example, for the nonce case:

(3) orto di regolamenti davvero pedonale/i
orchard of rules truly pedestrian
"truly pedestrian orchard of rules"

both the subjects and the model preferred to treat "pedestrian" as a modifier of "rules" ("orchard of truly pedestrian rules"), resulting in the wrong agreement given the intended syntactic structure.

Attractors. We define attractors as words with the same POS as the cue but the opposite number, which intervene in the linear order of the sen-

[10] The SM contains the results for the other languages broken down by construction. Note that Table 3 reports linguistically intuitive construction labels. The corresponding POS patterns are (in same order as table rows): DET ADJ NOUN, NOUN VERB PRON VERB, NOUN VERB VERB, ADJ ADJ CCONJ ADJ, NOUN ADJ PUNCT PRON VERB, NOUN NOUN ADV ADJ, NOUN NOUN VERB, VERB NOUN CCONJ VERB.

[11] The relatively low nonce LSTM performance on this construction is due to a few adjectives that could be reinterpreted as nouns.

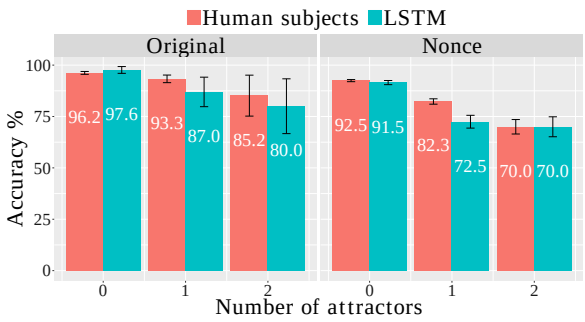


Figure 2: Accuracy by number of attractors in Italian. Human performance is shown in red and LSTM in blue (median model among top 5 ranked by perplexity). Error bars show standard error.

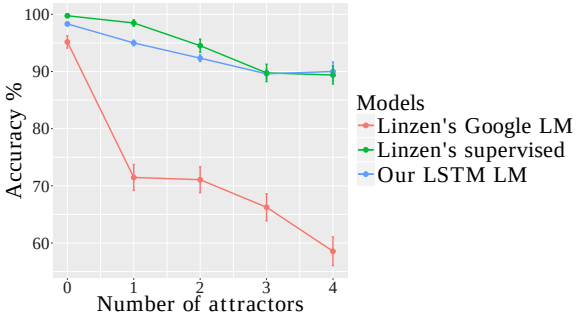


Figure 3: Linzen's attractor set. Our LM-trained LSTM (blue; "median" model) compared to their LSTM with explicit number supervision (green) and their best LM-trained LSTM (red).

tence between the cue and the target. Attractors constitute an obvious challenge for agreement processing (Bock and Miller, 1991). We show how their presence affects human and model behavior in Fig. 2. We limit our analysis to a maximum of two attractors, since there were only two original sentences in the test corpus with three attractors or more. Both model and human accuracies degraded with the number of attractors; the drop in accuracy was sharper in the nonce condition. While the model performed somewhat worse than humans, the overall pattern was comparable.

Our results suggest that the LSTM is quite robust to the presence of attractors, in contrast to what was reported by Linzen et al. (2016). We directly compared our English LSTM LM to theirs by predicting verb number on the Linzen et al. (2016) test set. We extracted sentences where all of the words between subject and verb were in our LM vocabulary. Out of those sentences, we sampled 2000 sentences with 0, 1 and 2 attractors and kept all the sentences with 3 and 4 attractors (1329 and 347 sentences, respectively). To ensure that our training set and Linzen's test set do not overlap (both are based on Wikipedia texts), we filtered out all of test sentences that appeared in our training data (187 sentences).

Fig. 3 compares our results to the results of the best LM-trained model in Linzen et al. (2016) (their "Google LM").[12] Not only did our LM greatly outperform theirs, but it approached the performance of their supervised model.[13] This difference in results points to the importance of careful tuning of LM-trained LSTMs, although we must leave to a further study a more detailed understanding of which differences crucially determine our better performance.

5 Related work

Early work showed that RNNs can, to a certain degree, handle data generated by context-free and even context-sensitive grammars (e.g., Elman, 1991, 1993; Rohde and Plaut, 1997; Christiansen and Chater, 1999; Gers and Schmidhuber, 2001; Cartling, 2008). These experiments were based on small and controlled artificial languages, in which complex hierarchical phenomena were often overrepresented compared to natural languages.

Our work, which is based on naturally occurring data, is most closely related to that of Linzen et al. (2016) and Bernardy and Lappin (2017), which we discussed in the introduction. Other recent work has focused on the morphological and grammatical knowledge that RNN-based machine-translation systems and sentence embeddings encode, typically by training classifiers to decode various linguistic properties from hidden states of the network (e.g., Adi et al., 2017; Belinkov et al., 2017; Shi et al., 2016), or looking at whether the end-to-end system correctly translates sentences with challenging constructions (Sennrich, 2017).

Previous work in neurolinguistics and psycholinguistics used jabberwocky, or pseudo-word, sentences to probe how speakers process syntactic information (Friederici et al., 2000; Moro et al.,

[12] These subject-verb agreement results are in general higher than for our own subject-verb agreement construction (NOUN VERB VERB) because the latter always includes an embedded clause, and it is therefore harder on average.

[13] Similarly high performance of LM-trained RNNs on Linzen's dataset was recently reported by Yogatama et al. (2018).

2001; Johnson and Goldberg, 2013). Such sentences are obtained by substituting original words with morphologically and phonologically acceptable nonce forms. We are not aware of work that used nonce sentences made of real words to evaluate the syntactic abilities of models or human subjects. As a proof of concept, Pereira (2000) and, later, Mikolov (2012) computed the probability of Chomsky's famous "colorless green ideas" sentence using a class-based bigram LM and an RNN, respectively, and showed that it is much higher than the probability of its shuffled ungrammatical variants.

6 Conclusion

We ran an extensive analysis of the abilities of RNNs trained on a generic language-modeling task to predict long-distance number agreement. Results were consistent across four languages and a number of constructions. They were above strong baselines even in the challenging case of nonsense sentences, and not far from human performance. We are not aware of other collections of human long-distance agreement judgments on nonsensical sentences, and we thus consider our publicly available data set an important contribution of our work, of interest to students of human language processing in general.

The constructions we considered are quite infrequent (according to a rough estimate based on the treebanks, the language in which they are most common is Hebrew, and even there they occur with average 0.8% sentence frequency). Moreover, they vary in the contexts that separate the cue and the target. So, RNNs are not simply memorizing frequent morphosyntactic sequences (which would already be impressive, for systems learning from raw text). We tentatively conclude that LM-trained RNNs can construct abstract grammatical representations of their input. This, in turn, suggests that the input itself contains enough information to trigger some form of syntactic learning in a system, such as an RNN, that does not contain an explicit prior bias in favour of syntactic structures.

In future work, we would like to better understand what kind of syntactic information RNNs are encoding, and how. On the one hand, we plan to adapt methods to inspect information flow across RNN states (e.g., Hupkes et al., 2017). On the other, we would like to expand our empirical investigation by focusing on other long-distance phenomena, such as overt case assignment (Blake, 2001) or parasitic gap licensing (Culicover and Postal, 2001). While it is more challenging to extract reliable examples of such phenomena from corpora, their study would probe more sophisticated syntactic capabilities, possibly even shedding light on the theoretical analysis of the underlying linguistic structures. Finally, it may be useful to complement the corpus-driven approach used in the current paper with constructed evaluation sentences that isolate particular syntactic phenomena, independent of their frequency in a natural corpus, as is common in psycholinguistics (Enguehard et al., 2017).

Acknowledgements

We thank the reviewers, Germán Kruszewski, Gerhard Jäger, Adam Liška, Tomas Mikolov, Gemma Boleda, Brian Dillon, Cristophe Pallier, Roberto Zamparelli and the Paris Syntax and Semantics Colloquium audience for feedback and advice.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *Proceedings of ICLR Conference Track*. Toulon, France. Published online: https://openreview.net/group?id=ICLR.cc/2017/conference.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do neural machine translation models learn about morphology? In *Proceedings of ACL*. Vancouver, Canada, pages 861–872.

Jean-Philippe Bernardy and Shalom Lappin. 2017. Using deep neural networks to learn syntactic agreement. *Linguistic Issues in Language Technology* 15(2):1–15.

Barry Blake. 2001. *Case*. MIT Press, Cambridge, MA.

Kathryn Bock and Carol Miller. 1991. Broken agreement. *Cognitive Psychology* 23(1):45–93.

Bo Cartling. 2008. On the implicit acquisition of a context-free grammar by a simple recurrent neural network. *Neurocomputing* 71:1527–1537.

Ciprian Chelba, Mohammad Norouzi, and Samy Bengio. 2017. N-gram language modeling using recurrent neural network estimation. *arXiv preprint arXiv:1703.10724* .

Noam Chomsky. 1957. *Syntactic Structures*. Mouton, Berlin, Germany.

Morten Christiansen and Nick Chater. 1999. Toward a connectionist model of recursion in human linguistic performance. *Cognitive Science* 23(2):157–205.

James Cross and Liang Huang. 2016. Incremental parsing with minimal features using bi-directional LSTM. In *Proceedings of ACL (Short Papers)*. Berlin, Germany, pages 32–37.

Peter Culicover and Paul Postal, editors. 2001. *Parasitic gaps*. MIT Press, Cambridge, MA.

Jeffrey Elman. 1990. Finding structure in time. *Cognitive Science* 14:179–211.

Jeffrey Elman. 1991. Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning* 7:195–225.

Jeffrey Elman. 1993. Learning and development in neural networks: The importance of starting small. *Cognition* 48:71–99.

Émile Enguehard, Yoav Goldberg, and Tal Linzen. 2017. Exploring the syntactic abilities of RNNs with multi-task learning. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. pages 3–14.

Martin Everaert, Marinus Huybregts, Noam Chomsky, Robert Berwick, and Johan Bolhuis. 2015. Structures, not strings: Linguistics as part of the cognitive sciences. *Trends in Cognitive Sciences* 19(12):729–743.

Marcello Federico, Nicola Bertoldi, and Mauro Cettolo. 2008. Irstlm: An open source toolkit for handling large scale language models. In *Ninth Annual Conference of the International Speech Communication Association*. pages 1618–1621.

Angela D. Friederici, Martin Meyer, and D. Yves Von Cramon. 2000. Auditory language comprehension: An event-related fMRI study on the processing of syntactic and lexical information. *Brain and Language* 74(2):289–300.

Felix Gers and Jürgen Schmidhuber. 2001. LSTM recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks* 12(6):1333–1340.

Alex Graves. 2012. *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer, Berlin.

Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, pages 187–197.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9(8):1735–178–.

Dieuwke Hupkes, Sara Veldhoen, and Willem Zuidema. 2017. Visualisation and diagnostic classifiers reveal how recurrent and recursive neural networks process hierarchical structure. http://arxiv.org/abs/1711.10203.

Matt A. Johnson and Adele E. Goldberg. 2013. Evidence for automatic accessing of constructional meaning: Jabberwocky sentences prime associated verbs. *Language and Cognitive Processes* 28(10):1439–1452.

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

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*. IEEE, volume 1, pages 181–184.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics* 4:521–535.

Heidi Lorimor, Kathryn Bock, Ekaterina Zalkind, Alina Sheyman, and Robert Beard. 2008. Agreement and attraction in Russian. *Language and Cognitive Processes* 23(6):769–799.

Tomas Mikolov. 2012. *Statistical language models based on neural networks*. Dissertation, Brno University of Technology.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of INTERSPEECH*. Makuhari, Japan, pages 1045–1048.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of NAACL*. Atlanta, Georgia, pages 746–751.

Andrea Moro, Marco Tettamanti, Daniela Perani, Caterina Donati, Stefano F Cappa, and Ferruccio Fazio. 2001. Syntax and the brain: disentangling grammar by selective anomalies. *Neuroimage* 13(1):110–118.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno,

Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association (ELRA), Paris, France.

Fernando Pereira. 2000. Formal grammar and information theory: together again? *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 358(1769):1239–1253.

Douglas Rohde and David Plaut. 1997. Simple recurrent networks and natural language: How important is starting small? In *Proceedings of CogSci*. Stanford, CA, pages 656–661.

Helmut Schmid. 1995. Improvements in part-of-speech tagging with an application to German. In *Proceedings of the EACL-SIGDAT Workshop*. Dublin, Ireland.

Rico Sennrich. 2017. How grammatical is character-level neural machine translation? assessing MT quality with contrastive translation pairs. In *Proceedings of EACL (Short Papers)*. Valencia, Spain, pages 376–382.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *Proceedings of EMNLP*. Austin, Texas, pages 1526–1534.

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

Dani Yogatama, Yishu Miao, Gabor Melis, Wang Ling, Adhiguna Kuncoro, Chris Dyer, and Phil Blunsom. 2018. Memory architectures in recurrent neural network language models. In *International Conference on Learning Representations*. https://openreview.net/forum?id=SkFqf0lAZ.

Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. 2017. Dependency parsing as head selection. In *Proceedings of EACL*. Valencia, Spain, pages 665–676.

Diverse Few-Shot Text Classification with Multiple Metrics

Mo Yu* **Xiaoxiao Guo*** **Jinfeng Yi*** **Shiyu Chang**
Saloni Potdar **Yu Cheng** **Gerald Tesauro** **Haoyu Wang** **Bowen Zhou**

AI Foundations – Learning, IBM Research
IBM T. J. Watson Research Center, Yorktown Heights, NY 10598

Abstract

We study few-shot learning in natural language domains. Compared to many existing works that apply either metric-based or optimization-based meta-learning to image domain with low inter-task variance, we consider a more realistic setting, where tasks are diverse. However, it imposes tremendous difficulties to existing state-of-the-art metric-based algorithms since a single metric is insufficient to capture complex task variations in natural language domain. To alleviate the problem, we propose an adaptive metric learning approach that automatically determines the best weighted combination from a set of metrics obtained from meta-training tasks for a newly seen few-shot task. Extensive quantitative evaluations on real-world sentiment analysis and dialog intent classification datasets demonstrate that the proposed method performs favorably against state-of-the-art few shot learning algorithms in terms of predictive accuracy. We make our code and data available for further study.[1]

1 Introduction

Few-shot learning (FSL) (Miller et al., 2000; Li et al., 2006; Lake et al., 2015) aims to learn classifiers from few examples per class. Recently, deep learning has been successfully exploited for FSL via learning meta-models from a large number of **meta-training tasks**. These meta-models can be then used for rapid-adaptation for the **target/meta-testing tasks** that only have few training examples. Examples of such meta-models include: (1) metric-/similarity-based models, which learn contextual, and task-specific similarity measures (Koch, 2015; Vinyals et al., 2016; Snell et al., 2017); and (2) optimization-based models, which receive the input of gradients from a FSL task and predict either model parameters or parameter updates (Ravi and Larochelle, 2017; Munkhdalai and Yu, 2017; Finn et al., 2017; Wang et al., 2017).

In the past, FSL has mainly considered image domains, where all tasks are often sampled from one huge collection of data, such as Omniglot (Lake et al., 2011) and ImageNet (Vinyals et al., 2016), making tasks come from a single domain thus related. Due to such a simplified setting, almost all previous works employ a common meta-model (metric-/optimization-based) for all few-shot tasks. However, this setting is far from the realistic scenarios in many real-world applications of few-shot text classification. For example, on an enterprise AI cloud service, many clients submit various tasks to train text classification models for business-specific purposes. The tasks could be classifying customers' comments or opinions on different products/services, monitoring public reactions to different policy changes, or determining users' intents in different types of personal assistant services. As most of the clients cannot collect enough data, their submitted tasks form a few-shot setting. Also, these tasks are significantly diverse, thus a common metric is insufficient to handle all these tasks.

We consider a more realistic FSL setting in this paper, where tasks are diverse. In such a scenario, the optimal meta-model may vary across tasks. Our solution is based on the metric-learning approach (Snell et al., 2017) and the key idea is to maintain multiple metrics for FSL. The meta-learner selects and combines multiple metrics for learning the target task using **task clustering** on the meta-training tasks. During the meta-training, we propose to first partition the meta-training tasks into clusters, making the tasks in each cluster

*Equal contributions from the corresponding authors: yum@us.ibm.com, xiaoxiao.guo@ibm.com, jinfengy@us.ibm.com.

[1] https://github.com/Gorov/DiverseFewShot_Amazon

Proceedings of NAACL-HLT 2018, pages 1206–1215
New Orleans, Louisiana, June 1 - 6, 2018.

likely to be related. Then within each cluster, we train a deep embedding function as the metric. This ensures the common metric is only shared across tasks within the same cluster. Further, during meta-testing, each target FSL task is assigned to a task-specific metric, which is a linear combination of the metrics defined by different clusters. In this way, the diverse few-shot tasks can derive different metrics from the previous learning experience.

The key of the proposed FSL framework is the task clustering algorithm. Previous works (Kumar and Daume III, 2012; Kang et al., 2011; Crammer and Mansour, 2012; Barzilai and Crammer, 2015) mainly focused on convex objectives, and assumed the number of classes is the same across different tasks (*e.g.* binary classification is often considered). To make task clustering (i) compatible with deep networks and (ii) able to handle tasks with a various number of labels, we propose a **matrix-completion based task clustering** algorithm. The algorithm utilizes task similarity measured by cross-task transfer performance, denoted by matrix $\mathbf{S}$. The (i, j)-entry of $\mathbf{S}$ is the estimated accuracy by adapting the learned representations on the i-th (source) task to the j-th (target) task. We rely on matrix completion to deal with missing and unreliable entries in $\mathbf{S}$ and finally apply spectral clustering to generate the task partitions.

To the best of our knowledge, our work is the first one addressing the diverse few-shot learning problem and reporting results on real-world few-shot text classification problems. The experimental results show that the proposed algorithm provides significant gains on few-shot sentiment classification and dialog intent classification tasks. It provides positive feedback on the idea of using multiple meta-models (metrics) to handle diverse FSL tasks, as well as the proposed task clustering algorithm on automatically detecting related tasks.

2 Problem Definition

Few-Shot Learning Since we focus on **diverse metric-based FSL**, the problem can be formulated in two stages: (1) **meta-training**, where a set of metrics $\mathcal{M} = \{\Lambda_1, \cdots, \Lambda_K\}$ is learned on the **meta-training tasks** $\mathcal{T}$. Each Λ_i maps two input (x_1, x_2) to a scalar of similarity score. Here $\mathcal{T} = \{\mathrm{T}_1, \mathrm{T}_2, \cdots, \mathrm{T}_N\}$ is a collection of N tasks. Here K is a pre-defined number (usually $K \ll N$). Each task T_i consists of training, validation, and testing set denoted as $\left\{D_i^{train}, D_i^{valid}, D_i^{test}\right\}$, respectively. Note that the definition of $\mathcal{T}$ is a generalized version of $\mathcal{D}^{(meta-train)}$ in (Ravi and Larochelle, 2017), since each task T_i can be either few-shot (where D_i^{valid} is empty) or regular[2]. (2) **meta-testing**: the trained metrics in $\mathcal{M}$ is applied to **meta-testing tasks** denoted as $\mathcal{T}' = \{\mathrm{T}'_1, \cdots, \mathrm{T}'_{N'}\}$, where each T'_i is a few-shot learning task consisting of both training and testing data as $\left\{D_i'^{train}, D_i'^{test}\right\}$. $D_i'^{train}$ is a small labeled set for generating the prediction model M'_i for each T'_i. Specifically, M'_is are kNN-based predictors built upon the metrics in $\mathcal{M}$. We will detail the construction of M'_i in Section 3, Eq. (6). It is worth mentioning that the definition of $\mathcal{T}'$ is the same as $\mathcal{D}^{(meta-test)}$ in (Ravi and Larochelle, 2017). The **performance of few-shot learning** is the macro-average of M'_i's accuracy on all the testing set $D_i'^{test}$s.

Our definitions can be easily generalized to other meta-learning approaches (Ravi and Larochelle, 2017; Finn et al., 2017; Mishra et al., 2017). The motivation of employing multiple metrics is that when the tasks are diverse, one metric model may not be sufficient. Note that previous metric-based FSL methods can be viewed as a special case of our definition where $\mathcal{M}$ only contains a single Λ, as shown in the two base model examples below.

Base Model: Matching Networks In this paper we use the metric-based model Matching Network (MNet) (Vinyals et al., 2016) as the base metric model. The model (Figure 1b) consists of a neural network as the embedding function (**encoder**) and an augmented memory. The encoder, $f(\cdot)$, maps an input x to a d-length vector. The learned metric Λ is thus the similarity between the encoded vectors, $\Lambda(x_1, x_2) = f(x_1)^T f(x_2)$, i.e. the metric Λ is modeled by the encoder f. The augmented memory stores a support set $S = \{(x_i, y_i)\}_{i=1}^{|S|}$, where x_i is the supporting instance and y_i is its corresponding label in a one-hot format. The MNet explicitly defines a classifier M conditioned on the supporting set S. For any new data $\hat{x}$, M predicts its label via a similarity function $\alpha(.,.)$

[2]For example, the methods in (Triantafillou et al., 2017) can be viewed as training meta-models from any sampled batches from one single meta-training dataset.

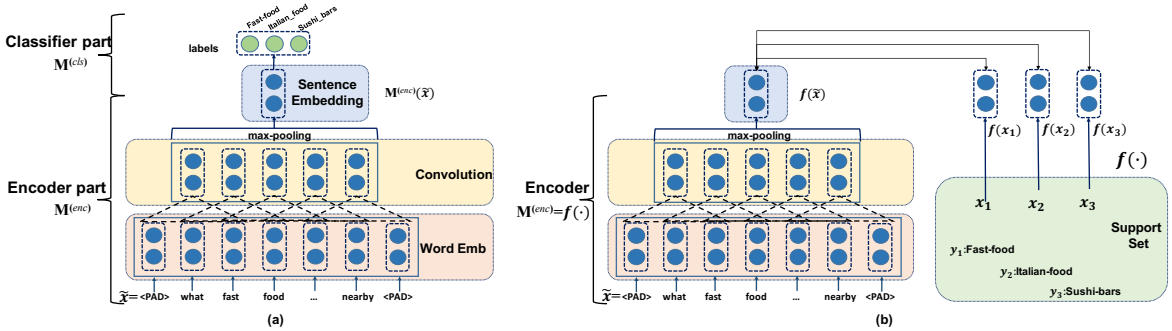


Figure 1: The Convolutional Neural Networks (CNN) used in this work: (a) A CNN classifier. The encoder component takes the sentence as input and outputs a fixed-length sentence embedding vector; the classifier component predicts class labels with the sentence embedding. (b) A Matching Network, which only contains an encoder like in (a), and makes prediction via a k-Nearest-Neighbor classifier with the similarity defined by the encoder.

between the test instance $\hat{x}$ and the support set S:

$$y = P(.|\hat{x}, S) = \sum_{i=1}^{|S|} \alpha(\hat{x}, x_i; \theta) y_i, \quad (1)$$

where we defined $\alpha(.,.)$ to be a softmax distribution given $\Lambda(\hat{x}, x_i)$, where x_i is a supporting instance, *i*.e., $\alpha(\hat{x}, x_i; \theta) = \exp(f(\hat{x})^T f(x_i)) / \sum_{j=1}^{|S|} \exp(f(\hat{x})^T f(x_j))$, where θ are the parameters of the encoder f. Thus, y is a valid distribution over the supporting set's labels $\{y_i\}_{i=1}^{|S|}$. To adapt the MNet to text classification, we choose encoder f to be a convolutional neural network (CNN) following (Kim, 2014; Johnson and Zhang, 2016). Figure 1 shows the MNet with the CNN architecture. Following (Collobert et al., 2011; Kim, 2014), the model consists of a convolution layer and a max-pooling operation over the entire sentence.

To train the MNets, we first sample the training dataset D for task T from all tasks $\mathcal{T}$, with notation simplified as $D \sim \mathcal{T}$. For each class in the sampled dataset D, we sample k random instances in that class to construct a support set S, and sample a batch of training instances B as training examples, i.e., $B, S \sim D$. The training objective is to minimize the prediction error of the training samples given the supporting set (with regard to the encoder parameters θ) as follows:

$$\mathop{\mathbb{E}}_{D\sim\mathcal{T}} \Big[\mathop{\mathbb{E}}_{B,S\sim D} \Big[\sum_{(x,y)\in B} \log(P(y|x, S; \theta)) \Big] \Big]. \quad (2)$$

Base Model: Prototypical Networks Prototypical Network (ProtoNet) (Snell et al., 2017) is a variation of Matching Network, which also depends on metric learning but builds the classifier M different from Eq. (1):

$$y = P(.|\hat{x}, S) = \sum_{i=1}^{L} \alpha(\hat{x}, S_i; \theta) y_i. \quad (3)$$

L is the number of classes and $S_i = \{x | (x, y) \in S \wedge y = y_i\}$ is the support set of class y_i. $\alpha(\hat{x}, S_i; \theta) = \exp\Big(f(\hat{x})^T \sum_{x\in S_i} f(x)\Big) / \sum_{j=1}^{L} \exp\Big(f(\hat{x})^T \sum_{x'\in S_j} f(x')\Big)$.

3 Methodology

We propose a task-clustering framework to address the diverse few-shot learning problem stated in Section 2. We have the FSL algorithm summarized in Algorithm 1. Figure 2 gives an overview of our idea. The initial step of the algorithm is a novel task clustering algorithm based on matrix completion, which is described in Section 3.1. The few-shot learning method based on task clustering is then introduced in Section 3.2.

3.1 Robust Task Clustering by Matrix Completion

Our task clustering algorithm is shown in Algorithm 2. The algorithm first evaluates the transfer performance by applying a single-task model i to another task j (Section 3.1.1), which will result in a (partially observed) cross-task transfer performance matrix $\mathbf{S}$. The matrix $\mathbf{S}$ is then cleaned and completed, giving a symmetry task similarity matrix $\mathbf{Y}$ for spectral clustering (Ng et al., 2002).

3.1.1 Estimation of Cross-Task Transfer Performance

Using single-task models, we can compute performance scores s_{ij} by adapting each M_i to each task

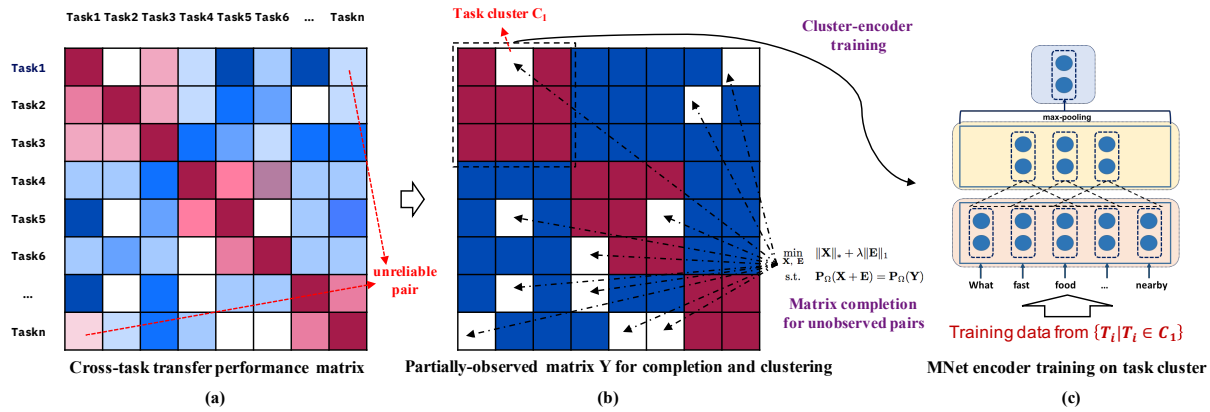


Figure 2: Overview of the idea of our multi-metric learning approach for few-shot learning. (a) an illustration of the sparse cross-tasks transfer-performance matrix with unobserved entries (white blocks) and unreliable values (top-right and bottom-left corners), where red colors indicate positive transfer and blue colors indicate negative transfer; (b) the constructed binary partially-observed matrix with low-rank constraint for matrix completion and clustering (see Section 3.1 for the details); (c) an encoder trained with the matching network objective Eq. (2) on a task cluster (tasks 1, 2 and 3 in the example).

$T_j (j \neq i)$. This forms an $n \times n$ pair-wise classification performance matrix $\mathbf{S}$, called the *transfer-performance matrix*. Note that $\mathbf{S}$ is asymmetric since usually $\mathbf{S}_{ij} \neq \mathbf{S}_{ji}$.

Algorithm 1: ROBUSTTC-FSL: Task Clustering for Few-Shot Learning

Input : N meta-training tasks $\mathcal{T}$=$\{\mathrm{T}_1, \mathrm{T}_2, \cdots, \mathrm{T}_n\}$; number of clusters K; N' target few-shot meta-testing tasks $\mathcal{T}'$

Output: Meta-model $\mathcal{M} = \{C_{1:K}$ (K task clusters), $\mathcal{F} = \{f_1, f_2, \cdots, f_K\}$ (K task encoders)$\}$. One classifier M'_i for each target task T'.

1 **Robust Task Clustering**: $C_{1:K}$ = ROBUSTTC($\mathcal{T}$,K) (Algorithm 2)
2 **Cluster-Model Training**: Train one encoder (multi-task MNet) f_i on each task cluster C_i (Section 3.2.1)
3 **Few-Shot Learning on Cluster-models**: Train a model M_{trg} on task T_{trg} with the method in Section 3.2.2.

Ideally, the transfer performance could be estimated by training a MNet on task i and directly evaluating it on task j. However, the limited training data usually lead to generally low transfer performance of single-task MNet. As a result we adopt the following approach to estimate $\mathbf{S}$:

We train a CNN classifier (Figure 1(a)) on task i, then take only the encoder M_i^{enc} from M_i and freeze it to train a classifier on task j. This gives us a new task j model, and we test this model on D_j^{valid} to get the accuracy as the transfer-performance $\mathbf{S}_{ij}$. The score shows how the representations learned on task i can be adapted to task j, thus indicating the similarity between tasks.

Algorithm 2: ROBUSTTC: Robust Task Clustering based on Matrix Completion

Input : A set of n tasks $\mathcal{T} = \{\mathrm{T}_1, \mathrm{T}_2, \cdots, \mathrm{T}_n\}$, number of task clusters K

Output: K task clusters $C_{1:K}$

1 **Learning of Single-Task Models**: train single-task models M_i for each task T_i
2 **Evaluation of Transfer-Performance Matrix**: get performance matrix $\mathbf{S}$ (Section 3.1.1)
3 **Score Filtering**: Filter the uncertain scores in $\mathbf{S}$ and construct the symmetric matrix $\mathbf{Y}$ using Eq. (4)
4 **Matrix Completion**: Complete the similar matrix $\mathbf{X}$ from $\mathbf{Y}$ using Eq. (5)
5 **Task Clustering**: $C_{1:K}$=SpectralClustering($\mathbf{X}$, K)

Remark: Out-of-Vocabulary Problem In text classification tasks, transferring an encoder with fine-tuned word embeddings from one task to another is difficult as there can be a significant difference between the two vocabularies. Hence, while learning the single-task CNN classifiers, we always make the word embeddings fixed.

3.1.2 Task Clustering Method

Directly using the transfer performance for task clustering may suffer from both efficiency and accuracy issues. First, evaluation of all entries in the matrix $\mathbf{S}$ involves conducting the source-target transfer learning $O(n^2)$ times, where n is the number of meta-training tasks. For a large number of diverse tasks where the n can be larger than 1,000, evaluation of the full matrix is unacceptable (over 1M entries to evaluate). Second, the estimated cross-task performance (i.e. some $\mathbf{S}_{ij}$ or $\mathbf{S}_{ji}$ scores) is often unreliable due to small data

size or label noise. When the number of the uncertain values is large, they can collectively mislead the clustering algorithm to output an incorrect task-partition. To address the aforementioned challenges, we propose a novel task clustering algorithm based on the theory of matrix completion (Candès and Tao, 2010). Specifically, we deal with the huge number of entries by randomly sample task pairs to evaluate the $\mathbf{S}_{ij}$ and $\mathbf{S}_{ji}$ scores. Besides, we deal with the unreliable entries and asymmetry issue by keeping only task pairs (i, j) with consistent $\mathbf{S}_{ij}$ and $\mathbf{S}_{ji}$ scores. as will be introduced in Eq. (4). Below, we describe our method in detail.

Score Filtering First, we use only reliable task pairs to generate a *partially-observed* similarity matrix $\mathbf{Y}$. Specifically, if $\mathbf{S}_{ij}$ and $\mathbf{S}_{ji}$ are high enough, then it is likely that tasks $\{i, j\}$ belong to a same cluster and share significant information. Conversely, if $\mathbf{S}_{ij}$ and $\mathbf{S}_{ji}$ are low enough, then they tend to belong to different clusters. To this end, we need to design a mechanism to determine if a performance is high or low enough. Since different tasks may vary in difficulty, a fixed threshold is not suitable. Hence, we define a dynamic threshold using the mean and standard deviation of the target task performance, i.e., $\mu_j = \text{mean}(\mathbf{S}_{:j})$ and $\sigma_j = \text{std}(\mathbf{S}_{:j})$, where $\mathbf{S}_{:j}$ is the j-th column of $\mathbf{S}$. We then introduce two positive parameters p_1 and p_2, and define high and low performance as $\mathbf{S}_{ij}$ greater than $\mu_j + p_1\sigma_j$ or lower than $\mu_j - p_2\sigma_j$, respectively. When both $\mathbf{S}_{ij}$ and $\mathbf{S}_{ji}$ are high and low enough, we set their pairwise similarity as 1 and 0, respectively. Other task pairs are treated as uncertain task pairs and are marked as unobserved, and don't influence our clustering method. This leads to a partially-observed symmetric matrix $\mathbf{Y}$, i.e.,

$$\mathbf{Y}_{ij}=\mathbf{Y}_{ji}=\begin{cases} 1 & \text{if } \mathbf{S}_{ij} > \mu_j + p_1\sigma_j \\ & \text{and } \mathbf{S}_{ji} > \mu_i + p_1\sigma_i \\ 0 & \text{if } \mathbf{S}_{ij} < \mu_j - p_2\sigma_j \\ & \text{and } \mathbf{S}_{ji} < \mu_i - p_2\sigma_i \\ \text{unobserved} & \text{otherwise} \end{cases} \quad (4)$$

Matrix Completion Given the partially observed matrix $\mathbf{Y}$, we then reconstruct the full similarity matrix $\mathbf{X} \in \mathbb{R}^{n\times n}$. We first note that the similarity matrix $\mathbf{X}$ should be of low-rank (proof deferred to appendix). Additionally, since the observed entries of $\mathbf{Y}$ are generated based on high and low enough performance, it is safe to assume that most observed entries are correct and only a few may be incorrect. Therefore, we introduce a sparse matrix $\mathbf{E}$ to capture the observed incorrect entries in $\mathbf{Y}$. Combining the two observations, $\mathbf{Y}$ can be decomposed into the sum of two matrices $\mathbf{X}$ and $\mathbf{E}$, where $\mathbf{X}$ is a low rank matrix storing similarities between task pairs, and $\mathbf{E}$ is a sparse matrix that captures the errors in $\mathbf{Y}$. The matrix completion problem can be cast as the following convex optimization problem:

$$\begin{aligned} \min_{\mathbf{X},\,\mathbf{E}} \quad & \|\mathbf{X}\|_* + \lambda\|\mathbf{E}\|_1 \\ \text{s.t.} \quad & \mathbf{P}_\Omega(\mathbf{X}+\mathbf{E}) = \mathbf{P}_\Omega(\mathbf{Y}), \end{aligned} \quad (5)$$

where $\|\circ\|_*$ denotes the matrix nuclear norm, the convex surrogate of rank function. Ω is the set of observed entries in $\mathbf{Y}$, and $\mathbf{P}_\Omega : \mathbb{R}^{n\times n} \mapsto \mathbb{R}^{n\times n}$ is a matrix projection operator defined as

$$[\mathbf{P}_\Omega(\mathbf{A})]_{ij} = \begin{cases} \mathbf{A}_{ij} & \text{if } (i,j) \in \Omega \\ 0 & \text{otherwise} \end{cases}$$

Finally, we apply spectral clustering on the matrix $\mathbf{X}$ to get the task clusters.

Remark: Sample Efficiency In the Appendix A, we show a Theorem 7.1 as well as its proof, implying that under mild conditions, the problem (5) can perfectly recover the underlying similarity matrix $\mathbf{X}^*$ if the number of observed correct entries is at least $O(n \log^2 n)$. This theoretical guarantee implies that for a large number n of training tasks, only a tiny fraction of all task pairs is needed to reliably infer similarities over all task pairs.

3.2 Few-Shot Learning with Task Clusters

3.2.1 Training Cluster Encoders

For each cluster C_k, we train a multi-task MNet model (Figure 1(b)) with all tasks in that cluster to encourage parameter sharing. The result, denoted as f_k is called the **cluster-encoder** of cluster C_k. The k-th metric of the cluster is thus $\Lambda(x_1, x_2) = f_k(x_1)^\intercal f_k(x_2)$.

3.2.2 Adapting Multiple Metrics for Few-Shot Learning

To build a predictor M with access to only a limited number of training samples, we make the prediction probability by linearly combining prediction from learned cluster-encoders:

$$p(y|x) = \sum_k \alpha_k P(y|x; f_k). \quad (6)$$

where f_k is the learned (and frozen) encoder of the k-th cluster, $\{\alpha_k\}_{k=1}^K$ are adaptable parameters trained with few-shot training examples. And the predictor $P(y|x; f_k)$ from each cluster is

$$P(y = y_l|x; f_k) = \frac{\exp\{f_k(x_l)^\intercal f_k(x)\}}{\sum_i \exp\{f_k(x_i)^\intercal f_k(x)\}} \quad (7)$$

x_l is the corresponding training sample of label y_l.

Remark: Joint Method versus Pipeline Method End-to-end joint optimization on training data becomes a popular methodology for deep learning systems, but it is not directly applicable to diverse FSL. One main reason is that deep networks could easily fit any task partitions if we optimize on training loss only, making the learned metrics not generalize, as discussed in Section 6. As a result, this work adopts a pipeline training approach and employing validation sets for task clustering. Combining reinforcement learning with meta-learning could be a potential solution to enable an end-to-end training for future work.

4 Tasks and Data Sets

We test our methods by conducting experiments on two text classification data sets. We used NLTK toolkit[3] for tokenization. The task are divided into meta-training tasks and meta-testing tasks (target tasks), where the meta-training tasks are used for clustering and cluster-encoder training. The meta-testing tasks are few-shot tasks, which are used for evaluating the method in Eq. (6).

4.1 Amazon Review Sentiment Classification

First, following Barzilai and Crammer (2015), we construct multiple tasks with the multi-domain sentiment classification (Blitzer et al., 2007) data set. The dataset consists of Amazon product reviews for 23 types of products (see Appendix D for the details). For each product domain, we construct three binary classification tasks with different thresholds on the ratings: the tasks consider a review as positive if it belongs to one of the following buckets $= 5$ stars, $>= 4$ stars or $>= 2$ stars.[4] These buckets then form the basis of the task-setup, giving us 23×3=69 tasks in total. For each domain we distribute the reviews uniformly to the 3 tasks. For evaluation, we select 12 (4×3) tasks from 4 domains (*Books, DVD, Electronics, Kitchen*) as the meta-testing (target) tasks out of all 23 domains. For the target tasks, we create 5-shot learning problems.

4.2 Real-World Tasks: User Intent Classification for Dialog System

The second dataset is from an online service which trains and serves intent classification models to various clients. The dataset comprises recorded conversations between human users and dialog systems in various domains, ranging from personal assistant to complex service-ordering or customer-service request scenarios. During classification, intent-labels[5] are assigned to user utterances (sentences). We use a total of 175 tasks from different clients, and randomly sample 10 tasks from them as our target tasks. For each meta-training task, we randomly sample 64% data into a training set, 16% into a validation set, and use the rest as the test set. The number of labels for these tasks varies a lot (from 2 to 100, see Appendix D for details), making regular k-shot settings not essentially limited-resource problems (e.g., 5-shot on 100 classes will give a good amount of 500 training instances). Hence, to adapt this to a FSL scenario, for target tasks we keep one example for each label (one-shot), plus 20 randomly picked labeled examples to create the training data. We believe this is a fairly realistic estimate of labeled examples one client could provide easily.

Remark: Evaluation of the Robustness of Algorithm 2 Our matrix-completion method could handle a large number of tasks via task-pair sampling. However, the sizes of tasks in the above two few-shot learning datasets are not too huge, so evaluation of the whole task-similarity matrix is still tractable. In our experiments, the incomplete matrices mainly come from the score-filtering step (see Eq. 4). Thus there is limited randomness involved in the generation of task clusters.

To strengthen the conclusion, we evaluate our algorithm on an additional dataset with a much larger number of tasks. The results are reported in the multi-task learning setting instead of the few-shot learning setting focused in this paper. Therefore we put the results to a non-archive version of

[3] http://www.nltk.org/

[4] Data downloaded from http://www.cs.jhu.edu/~mdredze/datasets/sentiment/, in which the 3-star samples were unavailable due to their ambiguous nature (Blitzer et al., 2007).

[5] In conversational dialog systems, intent-labels are used to guide the dialog-flow.

this paper[6] for further reference.

5 Experiments

5.1 Experiment Setup

Baselines We compare our method to the following baselines: (1) **Single-task CNN**: training a CNN model for each task individually; (2) **Single-task FastText**: training one FastText model (Joulin et al., 2016) with fixed embeddings for each individual task; (3) **Fine-tuned the holistic MTL-CNN**: a standard transfer-learning approach, which trains one MTL-CNN model on all the training tasks offline, then fine-tunes the classifier layer (i.e. $\mathrm{M}^{(cls)}$ Figure 1(a)) on each target task; (4) **Matching Network**: a metric-learning based few-shot learning model trained on all training tasks; (5) **Prototypical Network**: a variation of matching network with different prediction function as Eq. 3; (6) **Convex combining all single-task models**: training one CNN classifier on each meta-training task individually and taking the encoder, then for each target task training a linear combination of all the above single-task encoders with Eq. (6). This baseline can be viewed as a variation of our method without task clustering. We initialize all models with pre-trained 100-dim Glove embeddings (trained on 6B corpus) (Pennington et al., 2014).

Hyper-Parameter Tuning In all experiments, we set both p_1 and p_2 parameters in (4) to 0.5. This strikes a balance between obtaining enough observed entries in $\mathbf{Y}$, and ensuring that most of the retained similarities are consistent with the cluster membership. The window/hidden-layer sizes of CNN and the initialization of embeddings (random or pre-trained) are tuned during the cluster-encoder training phase, with the validation sets of meta-training tasks. We have the CNN with window size of 5 and 200 hidden units. The single-metric FSL baselines have 400 hidden units in the CNN encoders. On sentiment classification, all cluster-encoders use random initialized word embeddings for sentiment classification, and use Glove embeddings as initialization for intent classification, which is likely because the training sets of the intent tasks are usually small.

Since all the sentiment classification tasks are binary classification based on our dataset construction. A CNN classifier with binary output layer can be also trained as the cluster-encoder for each task cluster. Therefore we compared CNN classifier, matching network, and prototypical network on Amazon review, and found that CNN classifier performs similarly well as prototypical network. Since some of the Amazon review data is quite large which involves further difficulty on the computation of supporting sets, we finally use binary CNN classifiers as cluster-encoders in all the sentiment classification experiments.

Selection of the learning rate and number of training epochs for FSL settings, i.e., fitting αs in Eq. (6), is more difficult since there is no validation data in few-shot problems. Thus we pre-select a subset of meta-training tasks as meta-validation tasks and tune the two hyper-parameters on the meta-validation tasks.

5.2 Experimental Results

Table 1 shows the main results on (i) the 12 few-shot product sentiment classification tasks by leveraging the learned knowledge from the 57 previously observed tasks from other product domains; and (ii) the 10 few-shot dialog intent classification tasks by leveraging the 165 previously observed tasks from other clients' data.

Due to the limited training resources, all the supervised-learning baselines perform poorly. The two state-of-the-art metric-based FSL approaches, matching network (4) and prototypical network (5), do not perform better compared to the other baselines, since the single metric is not sufficient for all the diverse tasks. On intent classification where tasks are further diverse, all the single-metric or single-model methods (3-5) perform worse compared to the single-task CNN baseline (1). The convex combination of all the single training task models is the best performing baseline overall. However, on intent classification it only performs on par with the single-task CNN (1), which does not use any meta-learning or transfer learning techniques, mainly for two reasons: (i) with the growth of the number of meta-training tasks, the model parameters grow linearly, making the number of parameters (165 in this case) in Eq.(6) too large for the few-shot tasks to fit; (ii) the meta-training tasks in intent classification usually contain less training data, making the single-task encoders not generalize well.

In contrast, our ROBUSTTC-FSL gives consistently better results compared to all the baselines.

[6] https://arxiv.org/pdf/1708.07918.pdf

Model	Avg Acc	
	Sentiment	Intent
(1) Single-task CNN w/pre-trained emb	65.92	34.46
(2) Single-task FastText w/pre-trained emb	63.05	23.87
(3) Fine-tuned holistic MTL-CNN	76.56	30.36
(4) Matching Network (Vinyals et al., 2016)	65.73	30.42
(5) Prototypical Network (Snell et al., 2017)	68.15	31.51
(6) Convex combination of all single-task models	78.85	34.43
ROBUSTTC-FSL	**83.12**	**37.59**
Adaptive ROBUSTTC-FSL	-	**42.97**

Table 1: Accuracy of FSL on sentiment classification (Sentiment) and dialog intent classification (Intent) tasks. The target tasks of sentiment classification are 5-shot ones; and each intent target task contains one training example per class and 20 random labeled examples.

It outperforms the baselines in previous work (1-5) by a large margin of more than 6% on the sentiment classification tasks, and more than 3% on the intent classification tasks. It is also significantly better than our proposed baseline (6), showing the advantages of the usage of task clustering.

Adaptive ROBUSTTC-FSL Although the ROBUSTTC-FSL improves over baselines on intent classification, the margin is smaller compared to that on sentiment classification, because the intent classification tasks are more diverse in nature. This is also demonstrated by the training accuracy on the target tasks, where several tasks fail to find any cluster that could provide a metric that suits their training examples. To deal with this problem, we propose an improved algorithm to automatically discover whether a target task belongs to none of the task-clusters. If the task doesn't belong to any of the clusters, it cannot benefit from any previous knowledge thus falls back to single-task CNN. The target task is treated as "out-of-clusters" when none of the clusters could achieve higher than 20% accuracy (selected on meta-validation tasks) on its training data. We call this method **Adaptive ROBUSTTC-FSL**, which gives more than 5% performance boost over the best ROBUSTTC-FSL result on intent classification. Note that the adaptive approach makes no difference on the sentiment tasks, because they are more closely related so re-using cluster-encoders always achieves better results compared to single-task CNNs.

5.3 Analysis

Effect of the number of clusters Figure 3 shows the effect of cluster numbers on the two tasks. ROBUSTTC achieves best performance

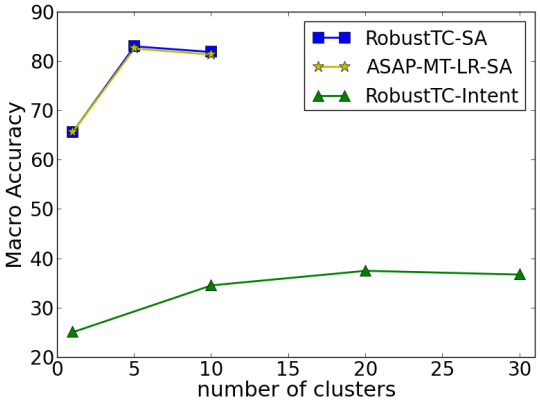


Figure 3: Effect of clusters. ROBUSTTC-SA and ROBUSTTC-Intent: the performance of our ROBUSTTC clusters on the sentiment and intent classification tasks. ASAP-MT-LR-SA: the state-of-the-art ASAP-MT-LR clusters on the sentiment-analysis tasks (the method is not applicable to the intent-classification tasks).

with 5 clusters on sentiment analysis (SA) and 20 clusters on intent classification (Intent). All clustering results significantly outperform the single-metric baselines (#cluster=1 in the figure).

Effect of the clustering algorithms Compared to previous task clustering algorithms, our ROBUSTTC is the only one that can cluster tasks with varying numbers of class labels (e.g. in intent classification tasks). Moreover, we show that even in the setting of all binary classifications tasks (e.g. the sentiment-analysis tasks) that previous task clustering research work on, our ROBUSTTC is still slightly better for the diverse FSL problems. Figure 3 compares with a state-of-the-art logistic regression based task clustering method (**ASAP-MT-LR**) (Barzilai and Crammer, 2015). Our ROBUSTTC clusters give slightly better FSL performance (e.g. 83.12 vs. 82.65 when #cluster=5).

	Clus0	Clus1	Clus2	Clus3	Clus4	Clus5	Clus6	Clus7	Clus8	Clus9
	automotive.t2 camera.t2 health.t2 magazines.t2 office.t2 outdoor.t2 sports.t2 sports.t4	apparel.t2 automotive.t4 baby.t2 cell.t2 computer.t2 computer.t4 computer.t5 jewelry.t4 music.t2 video.t2	baby.t5 magazines.t5 sports.t5 toys.t5 video.t5	automotive.t5 baby.t4 health.t4 health.t5	apparel.t5 camera.t5 grocery.t5 jewelry.t5	beauty.t4 beauty.t5 cell.t5 gourmet.t2 gourmet.t4 grocery.t2 grocery.t4 office.t4 outdoor.t4	camera.t4 software.t2 software.t4	gourmet.t5 magazines.t4 music.t4 music.t5 video.t4	cell.t4 software.t5 toys.t4	apparel.t4 toys.t2
dvd-t4	0.4844	0.4416	0.4625	**0.7843**	**0.7970**	0.7196	**0.8952**	0.3763	0.7155	0.6315
dvd-t5	0.0411	-0.2493	**0.5037**	**0.3567**	0.1686	-0.0355	**0.4150**	-0.2603	-0.0867	0.0547
kitchen-t4	0.6823	0.7268	0.7929	**1.2660**	**1.1119**	0.7255	**1.2196**	0.7065	0.6625	1.0945

Table 2: Visualization of clusters on the Amazon review domain. The top shows the training tasks assigned to the 10 clusters. Here the number N$\in \{2, 4, 5\}$ refers to the threshold of stars for positive reviews. At the bottom we show three tasks with largest improvement from ROBUSTTC-FSL. The top-3 most relevant task clusters (i.e. with highest weights αs in Eq.6) are highlighted with **blue bold** font.

Visualization of Task Clusters The top rows of Table 2 shows the ten clusters used to generate the sentiment classification results in Figure 3. From the results, we can see that tasks with same thresholds are usually grouped together; and tasks in similar domains also tend to appear in the same clusters, even the thresholds are slightly different (e.g. t2 vs t4 and t4 vs t5).

The bottom of the table shows the weights αs in Eq.(6) for the target tasks with the largest improvement. It confirms that our ROBUSTTC-FSL algorithm accurately adapts multiple metrics for the target tasks.

6 Related Work

Few Shot Learning FSL (Li et al., 2006; Miller et al., 2000) aims to learn classifiers for new classes with only a few training examples per class. Bayesian Program Induction (Lake et al., 2015) represents concepts as simple programs that best explain observed examples under a Bayesian criterion. Siamese neural networks rank similarity between inputs (Koch, 2015). Matching Networks (Vinyals et al., 2016) map a small labeled support set and an unlabeled example to its label, obviating the need for fine-tuning to adapt to new class types. These approaches essentially learn one metric for all tasks, which is sub-optimal when the tasks are diverse. An LSTM-based meta-learner (Ravi and Larochelle, 2017) learns the exact optimization algorithm used to train another learner neural-network classifier for the few-shot setting.

Previous FSL research usually adopts the k-shot, N-way setting, where all the few-shot tasks have the same number of N class labels, and each label has k training instances. Moreover, these few-shot tasks are usually constructed by sampling from one huge dataset, thus all the tasks are guaranteed to be related to each other. However, in real-world applications, the few-shot learning tasks could be diverse: there are different tasks with varying number of class labels and they are not guaranteed to be related to each other. As a result, a single meta-model or metric-model is usually not sufficient to handle all the few-shot tasks.

Task Clustering Previous task clustering methods measure the task relationships in terms of similarities among single-task model parameters (Kumar and Daume III, 2012; Kang et al., 2011); or jointly assign task clusters and train model parameters for each cluster to minimize the overall training loss (Crammer and Mansour, 2012; Barzilai and Crammer, 2015; Murugesan et al., 2017). These methods usually work on convex models but do not fit the deep networks, mainly because of (i) the parameters of deep networks are very high-dimensional and their similarities are not necessarily related to the functional similarities; and (ii) deep networks have flexible representation power so they may overfit to arbitrary cluster assignment if we consider training loss alone. Moreover, these methods require identical class label sets across different tasks, which does not hold in most of the realistic settings.

7 Conclusion

We propose a few-shot learning approach for diverse tasks based on task clustering. The proposed method can use multiple metrics, and performs significantly better compared to previous single-metric based methods when the few-shot tasks come from diverse domains. Future work includes generalizing our method to non-NLP problems, as well as applying the task-clustering idea to other few-shot learning frameworks (Ravi and Larochelle, 2017; Finn et al., 2017; Mishra et al., 2017; Cheng et al., 2017).

References

Aviad Barzilai and Koby Crammer. 2015. Convex multi-task learning by clustering. In *AISTATS*.

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

Emmanuel J Candès and Terence Tao. 2010. The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5):2053–2080.

Venkat Chandrasekaran, Sujay Sanghavi, Pablo A Parrilo, and Alan S Willsky. 2011. Rank-sparsity incoherence for matrix decomposition. *SIAM Journal on Optimization*, 21(2):572–596.

Yu Cheng, Mo Yu, Xiaoxiao Guo, and Bowen Zhou. 2017. Few-shot learning with meta metric learners. In *NIPS 2017 Workshop on Meta-Learning*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Koby Crammer and Yishay Mansour. 2012. Learning multiple tasks using shared hypotheses. In *Advances in Neural Information Processing Systems*, pages 1475–1483.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*.

Rie Johnson and Tong Zhang. 2016. Supervised and semi-supervised text categorization using one-hot lstm for region embeddings. *stat*, 1050:7.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Zhuoliang Kang, Kristen Grauman, and Fei Sha. 2011. Learning with whom to share in multi-task feature learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 521–528.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Gregory Koch. 2015. *Siamese neural networks for one-shot image recognition*. Ph.D. thesis, University of Toronto.

Abhishek Kumar and Hal Daume III. 2012. Learning task grouping and overlap in multi-task learning. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*.

Brenden M Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua B Tenenbaum. 2011. One shot learning of simple visual concepts. In *CogSci*, volume 172, page 2.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. 2015. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

Fei-Fei Li, Rob Fergus, and Pietro Perona. 2006. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611.

Erik G Miller, Nicholas E Matsakis, and Paul A Viola. 2000. Learning from one example through shared densities on transforms. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 1, pages 464–471. IEEE.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. 2017. A simple neural attentive meta-learner. In *NIPS 2017 Workshop on Meta-Learning*.

Tsendsuren Munkhdalai and Hong Yu. 2017. Meta networks. *arXiv preprint arXiv:1703.00837*.

Keerthiram Murugesan, Jaime Carbonell, and Yiming Yang. 2017. Co-clustering for multitask learning. *arXiv preprint arXiv:1703.00994*.

Andrew Y Ng, Michael I Jordan, and Yair Weiss. 2002. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Sachin Ravi and Hugo Larochelle. 2017. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, volume 1, page 6.

Jake Snell, Kevin Swersky, and Richard S Zemel. 2017. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*.

Eleni Triantafillou, Richard Zemel, and Raquel Urtasun. 2017. Few-shot learning through an information retrieval lens. In *Advances in Neural Information Processing Systems*, pages 2252–2262.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638.

Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. 2017. Learning to model the tail. In *Advances in Neural Information Processing Systems 30*, pages 7032–7042.

Early Text Classification using Multi-Resolution Concept Representations

A. Pastor López-Monroy[⋆], **Fabio A. González**[†], **Manuel Montes-y-Gómez**[‡§], **Hugo Jair Escalante**[‡] and **Thamar Solorio**[⋆]
[⋆] Department of Computer Science, University of Houston,Texas USA
[†] Systems and Computer Engineering Department, Universidad Nacional de Colombia
[‡] Instituto Nacional de Astrofísica, Óptica y Electrónica, Puebla México
[§] PRHLT Research Center, Universitat Politècnica de València, Spain
alopezmonroy@uh.edu, fagonzalezo@unal.edu.co,
{mmontesg, hugojair}@ccc.inaoep.mx, solorio@cs.uh.edu

Abstract

This paper proposes a novel document representation, called Multi-Resolution Representation (MulR), to improve the early detection of risks in social media sources. The goal is to effectively identify the potential risk using as little evidence as possible and with as much anticipation as possible. MulR allows us to generate multiple "views" of the text. These views capture different semantic meanings for words and documents at different levels of granularity, which is very useful in early scenarios to model the variable amounts of evidence. The experimental evaluation shows that MulR using low resolution is better suited for modeling short documents (very early stages), whereas large documents (medium/late stages) are better modeled with higher resolutions. We evaluate the proposed ideas in two different tasks where anticipation is critical: *sexual predator detection* and *depression detection*. The experimental evaluation for these early tasks revealed that the proposed approach outperforms previous methodologies by a considerable margin.

1 Introduction

Everyday there is a huge amount of people interacting in many social media sites. Unfortunately this immense cyber-world has been misused by cyber-criminals, who hide in the depths of the web. For this reason, the social media information has been increasingly studied in the context of applications related to security, forensics and e-commerce. Recently the early prediction scenarios have attracted the attention of the scientific community (Losada et al., 2017), which aims to prevent major threats in a number of practical situations by analyzing the text as evidence (e.g., sexual harassment, cyberbullying, etc).

In Natural Language Processing this emerging field is called early text classification and the goal is to identify risky-target categories by using as few text as possible and with as much anticipation as possible. In real scenarios the amount of evidence available from users under analysis is continuously growing. Consider for instance chat rooms, or posts and comments in social networks, these text sources comprise cumulative evidence for early prediction that can be used to better capture the phenomenon under study (Escalante et al., 2017; Losada et al., 2017). This scenario has challenging particularities. For example, in early stages where 10% or 20% of the information is available it is necessary to model very short length documents, which tend to produce sparse and low discriminative representations. On the other hand late stages require to exploit as much evidence as possible to make accurate predictions. This dynamism between the document length and classification stages makes necessary an adequate representation, that naturally copes with the dynamic amount of evidence in short and long texts generated by users at each stage. Traditional textual representations, such as Bag-of-Words (BoW) (Joachims, 1998), have problems dealing with social media short texts since they cause the representation to be high dimensional and very sparse. Moreover, in the particular case of early risk prediction, class unbalance and noisy text also represent a challenge.

In this paper we propose a representation that deals with these challenges by taking advantage of word vectors into a novel methodology for representing documents. This representation generates high-level features, that we called meta-words, which capture concepts at different resolution levels. A meta-word is a primitive construction represented by a vector that summarizes the information of semantically related words. Our methodology associates words with similar semantic meaning to the same meta-words. These meta-words

Proceedings of NAACL-HLT 2018, pages 1216–1225
New Orleans, Louisiana, June 1 - 6, 2018.

are obtained by applying clustering techniques to word representations, where the resultant "centroids" comprise the meta-words. Documents are then represented by a Bag-of-Centroids (BoC), that is, a histogram accounting for the occurrence of coarse thematic/semantic primitives, i.e., the meta-words. This part of the work is inspired by the Bag-of-Visual-Words (BoVW), which is widely used in computer vision to represent images (Sivic and Zisserman, 2004; Lazebnik et al., 2006).

The key aspect for early scenarios is that the number and size of meta-words, allow us to manipulate the level of granularity or the *resolution* of the representation. This property is very useful to capture discriminative information along the growing amount of available evidence at each early stage. We thus propose a multi-resolution approach, in which primitives at different resolutions are combined to capture feature concepts at multiple levels of detail. The contributions of this paper are twofold: (i) a new Multi-Resolution (MulR) document representation, a generalization to represent documents by exploiting word-vectors at different levels of resolutions; (ii) an empirical validation of the usefulness of multiple resolution levels for early risk detection on social media documents. Our experimental results show that this approach is a promising alternative for early text classification scenarios, where there is a need to make predictions as soon as possible, with little evidence, while at the same time, being robust to incorporate more evidence as it becomes available. We recorded experimental results of an extensive evaluation of our proposed techniques over two benchmarks for early scenarios: sexual predator detection and depression detection. Results showed that in all cases our methodology outperforms state-of-the-art methodologies.

Interestingly, document representations based on partitioning the word-embedding space, like ours, are somewhat similar to topic modeling based representations. In the experimental section we also compare the performance of our method to different topic-based representations like Latent Semantic Analysis (LSA) (Deerwester et al., 1990) and Latent Dirichlet Allocation (LDA) (Blei et al., 2003). Experimental results showed that our method outperformed the reference techniques. We elaborate on the benefits and limitations of our proposed techniques later in this paper.

2 Related Work

The Early Text Categorization problem is an emerging research topic with scant work (Dulac-Arnold et al., 2011; Escalante et al., 2016, 2017). Recently, the relevance of the problem has motivated specialized forums such as eRisk-CLEF17 (Losada et al., 2017). One of the first attempts is based on processing documents in a sentence-level basis (Dulac-Arnold et al., 2011). At every time t, the method reads a sentence and attempts to determine the class of the document. The key aspect of the work is a Markov Decision Process (MDP), where each sentence is modeled in a TFIDF vector. More recently, (Escalante et al., 2016) proposed a straightforward solution for early detection scenarios by using the naïve Bayes classifier. The idea consists in training with full documents, but when partial information has to be classified, the maximum a posteriori probability was estimated over the available text. Using this simple yet effective approach, the authors obtained competitive performance with the method in (Dulac-Arnold et al., 2011). Furthermore, results reported in (Escalante et al., 2016) were the first evaluation on early sexual predator detection.

In (Escalante et al., 2017) the authors propose methods to exploit Profile Based Representations (PBR's) for words (López-Monroy et al., 2015). PBRs are Distributional Term Representations of terms in the vocabulary. Similar to word embeddings these representations build a vector for each word, which aim to extract/learn concepts from simple occurrence statistics of terms in the target classes. PBRs capture discriminative information in a very low dimensional and non-sparse space suitable for early text classification problems. In other work, (Errecalde et al., 2017) successfully adapted a version of PBR's for the problem of early depression detection in the context of the eRisk-CLEF17 shared task. The evidence about PBRs suggests that this representation can naturally cope with missing information and obtain discriminative representations for incomplete documents. Nevertheless, just as the vast majority of word embeddings in the literature for standard text classification, there is no consensus about how to exploit these term vectors to represent entire phrases or documents (e.g., the most common strategy is to average the term vectors in documents).

The proposed method is based on creating meta-

words to represent documents. Clustering words into meaningful groups based on some measure of similarity to represent text is not a new concept. One of the classic approaches is term clustering in an *unsupervised* manner that was first investigated by (Lewis, 1992). He called his method *reciprocal nearest neighbor clustering*. His method consists of joining words that are similar according to a measure of similarity. In other work, Brown et al. (1992) explored the idea of discovering similarities between words to obtain clusters at different levels. One key difference with our proposal is that in (Brown et al., 1992), terms are deterministically/probabilistically associated with a discrete class, where terms that are in the same class are similar in some aspect. However in our proposed strategy, we exploit word vectors instead of a discrete random deterministic variable (e.g., soft/hard partitions of word sets). This makes possible to discover different clusters and meta-words if we change the word representation. Thus, the proposed strategy is highly adaptable to other domains, where the specialization would be achieved by changing the word representation for the problem. In other work, Li and Jain (1998) found that term grouping helps to reduce the feature dimensionality, and at the same time, overcomes the generalization problem of feature selection. The evidence has showed that the performance of the classifier is, at least maintained (Li and Jain, 1998; Slonim and Tishby, 2001). Finally, other authors have also studied the problem of term clustering under a *supervised* scheme. For example, Baker and McCallum (1998) used a *supervised* scheme to cluster similar words. They carried out experiments using a Naive Bayes classifier and found results improvement by using a single word representation.

The methods proposed in this research work follow a line of thinking focused on the document representation rather than term representation. Hence, the proposed method takes advantage of specialized vector representation of words (e.g., PBR), but several extensions can be envisioned using other word embeddings in the literature. The benefits of our approach are that it is model independent, easy to implement, and computes lower dimensional and less-sparse representations than traditional BoW. More important, our method improves over state of the art methods, outperforming the methods in (Errecalde et al., 2017; Escalante et al., 2017) that in turn, outperform that in (Dulac-Arnold et al., 2011; Escalante et al., 2016).

3 Multi-Resolution Document Representation

We propose a multi resolution representation that allows to generate multiple "views" of the analyzed document. The intuition behind the proposal of a multi-resolution representation is that words will activate differently each view according to the amount of available text. We assume that having different resolution levels will allow to effectively represent the content of short and large texts as needed along different early stages. The proposed multi-resolution framework is depicted in Figure 1. The idea consists in associating words with similar meaning to the same meta-words in each resolution space. Documents are then represented by multiple Bag-of-Centroids (BoC), that is, multiple histograms accounting for the occurrence of coarse concepts. Hence, this representation can be seen as multiple BoW representations that incorporate multiple semantic resolutions. In Section 3.1 we describe the process to build a Bag-of-Centroids at a single resolution, then in Section 3.2 we formally present the Multi-Resolution variant.

3.1 Single Resolution: Bag of Centroids

Let $\mathcal{D} = \{(d_1, y_1), \ldots, (d_h, y_h)\}$ be a training set of h-pairs of documents d_i and class labels y_i. Also let $\mathcal{V} = \{w_1, \ldots, w_r\}$ denote the vocabulary of terms (in our case words). In order to create the Bag of Centroids (***BoC***) representation of each document, we first compute the vector representation v_i of each word w_i in the vocabulary of the collection. Note that our framework is agnostic to the underlying process for learning word representations and therefore any word vector representation can be used, for example word embeddings (Mikolov et al., 2013) or distributional term representations (Lavelli et al., 2004).

The proposed framework is based on the idea of clustering words using the semantic "distance" in the word embedding space. Thus, the first step of the algorithm consists of clustering the word embedding vectors v_i and finding the cluster centers to create the proposed meta-words. The representation of the vocabulary collection in the word embedding space W is the input for the clustering

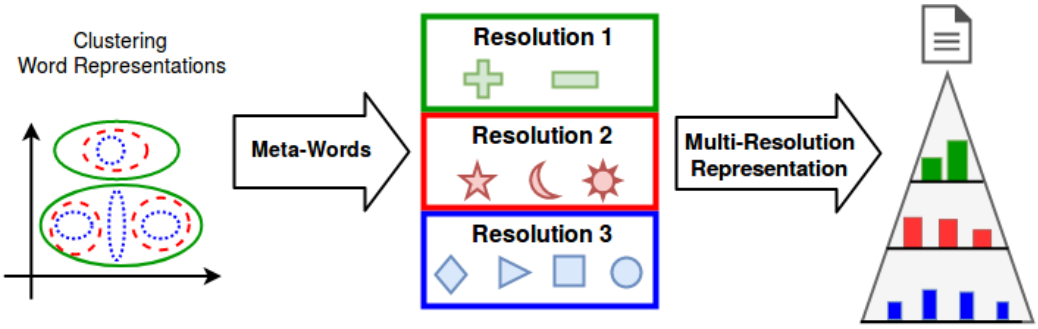


Figure 1: Algorithm to represent documents as meta-words using three hypothetical resolutions. The document is represented using the "meta-words" defined by the clustering of the vector representations of the vocabulary.

algorithm. For this purpose a variety of clustering approaches can be used. In our experimental evaluation, we explored different algorithms and found out that k-means offers a good trade-off between performance and speed. We applied k-means to the W representation to find the center of the clusters $C = \{c_1, c_2, \ldots, c_k\}$, with k being the number of selected centroids. Then, based on these cluster centers and using l_1-norm, we found a one-to-one association of each word to the *closest* cluster center in the word embedding space. In other words, for each word v_i, we can find an associated cluster center or meta-word c_u with $u \in \{1, 2, \ldots, k\}$. We denote this mapping by $c_u = closest(v_i, C)$, where $closest$ returns the centroid in C with the minimum distance to v_i. Finally, the $\mathbf{BoC}_k$ representation for each document d_j corresponds to $\mathbf{BoC}_k(d_j) = \{(c_\ell, n_\ell)\}_{\ell=1\ldots k}$ where c_ℓ corresponds to each of the k centroids and $n_\ell = |\{v_i | \forall v_i \in d_j, c_\ell = closest(v_i, C)\}|$. In other words, $\mathbf{BoC}_k(d_j)$ corresponds to a histogram of centroid frequencies, where each pair (c_ℓ, n_ℓ) represents a centroid (meta-word) and its corresponding frequency in the document.

The BoC algorithm depends on one parameter: the number of clusters used to represent each document. This parameter is associated with the level of semantic coarseness used in the representation. In this regard, coarseness refers to the level of meta-word inclusivity: the more words associated with a single meta-word, the coarser the representation. Conversely, with fewer words, the representation becomes more granular. Note that this representation has well known parallels in the extreme cases. When each word becomes a centroid, the resulting representation is equivalent to the typical BoW representation, whereas a coarser representation, with only one meta word, will be equivalent to having the average meta-word of the entire collection.

3.2 Multi-Resolution BoC

The above proposed framework is particularly suitable for incorporating multi-resolution processing, given that the main parameter is related to the granularity or coarseness of the representation. As we will show in our analysis, this property is useful for early scenarios, since few/coarse meta-words allow to better encode documents with little text, whereas many/granular meta-words are useful when more text become available. We propose to exploit this multi-resolution version of the **BoC** representation. In this extension of the basic algorithm, we use a partition of the word embedding space at multiple levels and concatenate them into a new representation. Combining the different granularities into a single representation results in a more robust document model that can help to capture different amounts of text as needed. Intuitively, the coarser levels sufficiently classify documents in early stages, while the more granular levels exploit the additional evidence from longer documents on late stages. We present quantitative and qualitative experiments that support this claim in two datasets: Sexual Predator Detection and Depression Detection.

We call this variation of the BoC representation Multi-Resolution-BoC (**MulR**). Formally: $\mathbf{MulR}(d_j) = \{\mathbf{BoC}_{k_1}(d_j) \cup \mathbf{BoC}_{k_2}(d_j) \cup \ldots \cup \mathbf{BoC}_{k_n}(d_j)\}$, where $\{k_1, k_2, \ldots k_n\}$ correspond to a set of granular levels. Figure 1 shows the general framework, graphically depicting the process involved in transforming a document into a representation based on meta-words. The figure also includes the process of multi-resolution modification described above. In the figure, the

meta-words depicted with 'blue' represent the more granular clusters, and those depicted with 'green' represent the less granular clusters. The multi-resolution BoC variation improves the performance by combining the information present at various levels of granularity. Moreover, when documents are closely related, more fine grained features allow to capture finer details and therefore produces better text classification results. This multi-resolution approach combines the advantages of both approaches to create an overall more effective classification method.

4 Data collections

For experiments we considered the two data sets described in Table 2. The tasks are Sexual Predator Detection (SPD) and Depression Detection, where clearly early detection is crucial. For the former we used the only publicly available data set for sexual predator detection (Inches and Crestani, 2012). This data set was released in the context of the sexual predator identification task at PAN-CLEF'12 and comprises a large number of chat conversations that include real sexual predators. Thus, the task approached is that of identifying those conversations that potentially include a sexual predator, as in (Villatoro-Tello et al., 2012; Escalante et al., 2013, 2016). For the depression detection task we use the dataset presented in (Losada et al., 2017). In this dataset, each instance has the post history for a user, and depressed users were self-identified as having been diagnosed with depression.

5 Evaluation Framework

For our experiments, we lower case the text in documents and use words and punctuation marks as terms[1]. The representation obtained for each document is then processed by a Support Vector Machine (SVM) with a linear kernel.

For the evaluation of the *earliness* performance, we report the performance of the different methods when using increasing amounts of textual evidence (chunk by chunk evaluation). This evaluation allows to quantify prediction performance when using partial information in documents, and it is a strategy that has been used to evaluate early classification (Escalante et al., 2016; Errecalde et al., 2017; Losada et al., 2017). For the evaluation of performance we used the $f_1 = \frac{2 \times precision \times recall}{precision + recall}$ measure. This decision was made in agreement with previous work that reports this metric for the positive class (Errecalde et al., 2017). Please note that, contrary to other measures, such as accuracy, f_1 measure accounts for the class imbalance problem when only the positive class is analyzed. This is desirable for the data sets we consider as they are highly unbalanced.

Word-vector representations: As previously mentioned, the proposed MulR representation generalizes word-vector representations and thus can extend any representation that models each term in the vocabulary using a vector. For this purpose a wide variety of word embeddings or distributional term representations could be used. Both of them exploit the distributional hypothesis to build word vectors, nonetheless they differ in the strategy to capture the relevant information. In this work we use the widely used word2vec, but also other representations that have been used in recent works for these collections. In Table 1 we describe each of the word vector representations considered for this work[2].

Baselines: The main baselines in this work are methods based on the idea of topic modeling for text classification. Topic-based representations group words into topics defined by a set of related words[3]. Given the strong relation to our method we compare our proposal against Latent Semantic Analysis (LSA) and Latent Dirichlet Allocation (LDA). Furthermore, we also compare with Bag-of-Words using Term Frequency Inverse Document Frequency, since it is a traditional baseline in text categorization tasks.

6 Experimental Results

In this section we report the experimental results for the MulR representation and the selected approaches from the state-of-the-art. In all the experiments we trained the reference classifier (SVM) using full-length documents in the training dataset. In the testing phase, each approach uses all the available information in each of the ten chunks (each chunk increases the available text in 10%). More specifically, we generate document representations starting with the first chunk, and then incrementally adding one chunk at a time. The

[1] We used terms with frequency higher than 10 in the training datasets.

[2] For distributional representations we used the framework at https://github.com/lopez-monroy/FeatureSpaceTree

[3] We empirically set to 200 the size of the concept space.

Word Representation	Description
W2V (Mikolov et al., 2013)	Word2Vec uses the Skip-ngram model to find word representations that are useful to predict the surrounding words of a sentence or a document. The method is efficient for learning high-quality vector representations of words from large amounts of text data. We empirically set to 200 the vector dimension.
DOR (Lavelli et al., 2004)	Document Occurrence Representation (DOR) captures the semantics of a word by observing occurrence distribution over documents in the corpus. DOR represents each word v_i as a vector $\mathbf{t_i} = \langle t_{i,1}, \ldots, t_{i,\|\mathcal{D}\|} \rangle$, where $\|\mathcal{D}\|$ is the number of documents in the training collection, and $t_{i,k}$ indicates the relevance of the document D_k to characterize v_i.
TCOR(Lavelli et al., 2004)	In Term Co-occurrence Representation (TCOR) the semantics of a word is captured by observing its co-occurrences with other words across documents in the corpus. Thus, each word v_i is associated to a vector $\mathbf{t_i} = \langle t_{i,1}, \ldots, t_{i,\|\mathcal{V}\|} \rangle$, where $\|\mathcal{V}\|$ indicates the vocabulary size, and $t_{i,k}$ denotes the contribution of the word v_k to the semantic description of v_i.
PBR (López-Monroy et al., 2015)	Profile Based Representation exploits occurrence-statistics of words over a set of documents in target categories. PBR represents each word $v_i \in V$ with a vector $\mathbf{t_i} = \langle t_{i,1}, \ldots, t_{i,q} \rangle$, where the $t_{i,k}$ is the degree of association between word v_i and category C_k. The target categories can be taken from the task or artificially created by means of clustering such as in (Escalante et al., 2017).
TVT (Errecalde et al., 2017)	Temporal Variation Terms (TVT) is an adapted version of PBR for early scenarios. TVT builds new artificial target classes/labels in the training set simulating a text stream to generate enriched representations of $\mathbf{t_i}$. The idea is to exploit the positive category to create a set of new artificial categories using text-fragments.

Table 1: Word vector representations for early experimentation.

Task	Data set	Training	Test
Sexual predator det.	PAN'12	6588	15329
Depression det.	eRisk'17	486	401

Table 2: Data sets considered for early experimentation. There are only two classes in each dataset.

models will then make predictions incrementally as well. We report f_1 performance when using different amounts of text from test documents. For the proposed MulR representation, we build 5 different resolutions: 10, 50, 100, 500, and 1000. The goal was to generate meta-words at different levels of granularity, and we plan to further explore the impact of these resolutions in our future research.

In the following experiments, we used the word representations in Table 1 to build our proposed MulR document representation. For comparison purposes we also generate an alternative document representation by averaging (Avg) term-vectors of words in each document, which is a popular strategy to build document representations. Finally, we also compare against several traditional baselines such as the Bag-of-Words, LSA, and specialized methods in each collection (Escalante et al., 2017; Errecalde et al., 2017). We evaluate the usefulness of all these different representations in the two early classification tasks mentioned earlier.

6.1 Sexual Predators Detection

In this section we evaluate the performance of the proposed MulR and other reference methodologies for the SPD early detection task (Figure 2). We also show results for MulR and different word representations in Table 3, where several findings can be outlined. First of all, results obtained in early stages (chunk 1 to 4) using the proposed MulR are clearly superior to those obtained averaging word vectors. This is an interesting outcome, since the MulR representation seems to be useful for early scenarios independently of the word vector representation. In the particular case of MulR(TVT), the representation obtains an outstanding performance when having little information (e.g., performance between ≈71% and ≈90% before reading 50% of the text). More important, performance improves as more evidence is available (i.e., see the steady improvement up to ≈97%). These results show that MulR is a robust representation, even in the presence of different amounts of textual evidence, with a clear advantage for early classification stages.

In Figure 2 we can also observe that MulR representation outperformed, by a large margin, the proposed baselines; BoW-TFIDF, LSA, LDA. Furthermore, MulR representation obtains better performance than the work in (Escalante et al., 2017), which consists in averaging the PBRs (same that Avg-PBR) and is the state-of-the-art in early SPD. Note that different than (Escalante et al., 2017), the proposed MulR significantly improves even after reading 40% of the information. The experimental results in Table 3 also show the

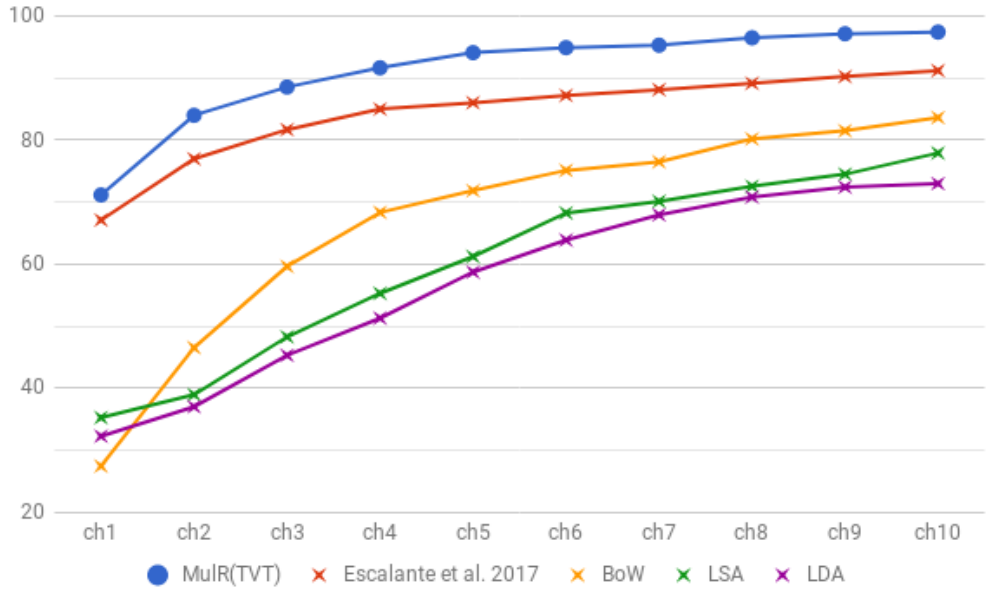


Figure 2: F_1 scores for the chunk by chunk evaluation of the reference methodologies in Sexual Predator Detection.

Method	ch_1	ch_2	ch_3	ch_4	ch_5	ch_6	ch_7	ch_8	ch_9	ch_{10}
BoW-TFIDF	27.40	46.51	59.62	68.33	71.84	75.11	76.49	80.19	81.51	83.63
LSA	35.22	38.93	48.25	55.27	61.21	68.24	70.12	72.54	74.49	77.91
LDA	32.22	36.98	45.27	51.27	58.70	63.87	67.94	70.81	72.41	72.98
Avg(W2V)	55.74	63.87	70.11	82.53	87.24	88.97	88.01	87.45	84.71	83.12
MulR(W2V)	58.97	65.78	71.97	83.09	85.49	87.21	88.46	89.00	89.15	89.49
Avg(DOR)	66.71	76.54	81.01	91.14	92.23	93.91	95.19	95.87	96.47	96.59
MulR(DOR)	68.24	78.77	87.14	**92.07**	**94.18**	94.04	94.84	95.24	95.46	95.97
Avg(TCOR)	60.17	67.97	74.41	78.51	81.24	82.71	83.97	82.51	82.90	82.27
MulR(TCOR)	61.51	69.12	75.43	78.89	80.26	79.97	81.01	81.59	82.14	83.01
Avg(PBR)	67.10	76.97	81.69	85.00	86.03	87.21	88.14	89.16	90.25	91.21
MulR(PBR)	69.16	77.41	83.01	87.05	88.07	89.27	90.14	91.51	92.01	92.41
Avg(TVT)	65.74	80.24	86.19	90.25	92.02	93.13	94.39	95.23	95.89	96.58
MulR(TVT)	**71.15**	**84.00**	**88.56**	91.66	94.11	**94.92**	**95.31**	**96.50**	**97.16**	**97.43**
(Escalante et al., 2017)	67.10	76.97	81.69	85.00	86.03	87.21	88.14	89.16	90.25	91.21

Table 3: F_1 results for the chunk by chunk evaluation of different approaches in Sexual Predator Detection. The proposed MulR is evaluated using different word vector representations in the literature.

following interesting findings:

1. The most useful word vector representation is TVT (Errecalde et al., 2017). This is not surprising, since TVT is a specialized distribution term representation for early prediction scenarios.

2. Word2Vec [4] representations obtained moderate performance in all experiments. We infer that much more data of these specific social media domains are needed in order to build suitable models.

3. MulR representation is an effective solution for all early chunks, but as more text is available, the other methodologies significantly increase their discriminative power, as seen in results for later chunks. In fact, some representations such as Avg(DOR) can outperform MulR(DOR) representation in late stages. However, even under these conditions MulR(TVT) and MulR(DOR) outperform all reference methodologies.

6.2 Depression Detection

In Table 4 we show the experimental results for early depression detection. In Figure 3 we highlight the performance of the proposal and the reference methodologies. From these results we point

[4]Embeddings were trained in each dataset. We tested pretrained word embeddings for wikipedia/twitter, but the performance was worse.

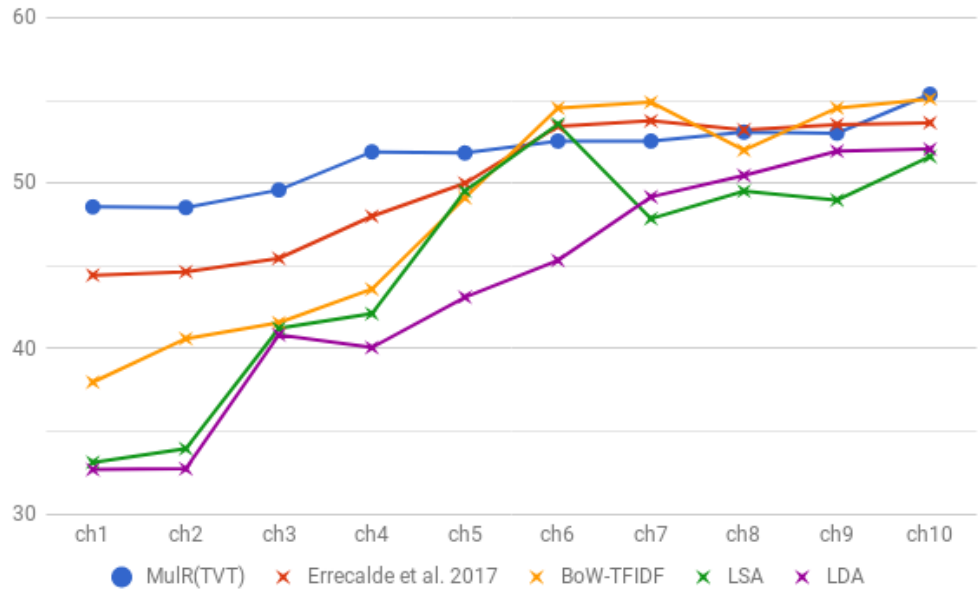


Figure 3: F_1 scores for the chunk by chunk evaluation of the reference methodologies in Depression Detection.

Method	ch_1	ch_2	ch_3	ch_4	ch_5	ch_6	ch_7	ch_8	ch_9	ch_{10}
BoW-TFIDF	37.97	40.60	41.56	43.59	49.12	54.55	54.90	52.00	54.55	55.10
LSA	33.12	33.94	41.22	42.11	49.52	53.57	47.86	49.52	48.98	51.61
LDA	32.68	32.73	40.83	40.06	43.11	45.33	49.17	50.47	51.94	52.06
Avg(W2V)	35.86	41.03	47.06	48.25	44.93	51.80	54.38	56.14	55.28	55.14
MulR(W2V)	41.34	43.79	47.20	48.44	47.67	52.00	53.91	54.18	54.29	54.39
Avg(DOR)	46.06	47.23	48.02	50.54	54.26	58.27	**57.81**	**58.73**	**59.84**	**66.12**
MulR(DOR)	47.55	48.12	48.38	51.83	**55.56**	**58.49**	52.43	53.06	57.73	54.35
Avg(TCOR)	37.42	44.44	44.60	48.64	49.64	53.33	52.94	53.44	52.46	58.32
MulR(TCOR)	44.76	47.95	46.81	48.32	51.47	52.11	54.01	54.55	56.30	57.06
Avg(PBR)	36.70	45.71	44.00	47.83	46.67	51.61	51.69	52.27	49.41	51.76
MulR(PBR)	40.98	46.15	44.83	48.70	50.00	53.10	57.39	54.55	54.55	55.86
Avg(TVT)	39.18	44.21	45.83	46.94	48.42	51.02	48.94	46.15	48.35	51.11
MulR(TVT)	**48.57**	**48.53**	**49.59**	**51.90**	51.83	52.55	52.55	53.09	53.03	55.38
(Errecalde et al., 2017)	44.44	44.64	45.45	48.00	50.00	53.44	53.77	53.23	53.55	53.66

Table 4: F_1 results for the chunk by chunk evaluation of different approaches in Depression Detection. The proposed MulR is evaluated using different word vector representations in the literature.

out several interesting findings. The first one is that for this collection, results obtained by the proposal are clearly superior to others in early stages. In general, we can observe the following:

1. The most useful representation in early stages was MulR(TVT), which have considerable improvements between ≈5% and ≈2% in chunks 1 to 4.

2. Word Embeddings and DOR showed a similar behavior than in SPD. But in late stages, the best representation was Avg(DOR).

3. Depression Detection problem is a much harder problem than SPD. The F_1 measure is under ≈60% in most of the results. This could be due to the highly unbalanced dataset in two ways: i) the number of instances in each class, and ii) the amount of text contained in documents.

6.3 The Relevance of Individual Resolutions

In this section we aim to study the role of the different resolutions in early scenarios.[5] The purpose of the first analysis is to observe the performance of each individual resolution in MulR. In Table 5 we show the results of MulR(TVT) under each of the five resolutions ($R_1 = 10, R_2 = 50, R_3 = 100, R_4 = 500, R_5 = 1000$) and each chunk.

[5]The number and size of resolutions, could improve the performance, but it is a future research path to enhance the characterization of specific data sets.

Sexual Predator Detection										
Method	ch_1	ch_2	ch_3	ch_4	ch_5	ch_6	ch_7	ch_8	ch_9	ch_{10}
MulR(TVT)-R1	59.88	74.71	82.17	85.82	89.19	90.19	91.51	92.61	92.42	92.51
MulR(TVT)-R2	70.03	83.03	88.04	90.49	93.00	94.01	94.85	94.70	94.90	95.02
MulR(TVT)-R3	67.87	82.41	87.23	90.17	92.07	92.88	93.91	94.89	95.49	96.20
MulR(TVT)-R4	66.10	80.33	85.89	88.76	90.48	91.86	93.14	93.85	94.73	95.38
MulR(TVT)-R5	62.34	77.73	83.05	86.72	88.64	90.32	92.16	93.04	93.00	93.76
MulR(TVT)	**71.15**	**84.00**	**88.56**	**91.66**	**94.11**	**94.92**	**95.31**	**96.50**	**97.16**	**97.43**

Table 5: F_1 results for the chunk by chunk evaluation of different approaches in Sexual Predator Detection. The best MulR(TVT) is separately evaluated under each resolution.

For early SPD the evidence is clear; as the resolution increases the performance in early stages decrease.[6] Also note that the higher the resolution, the more chunks needed to outperform the result of the previous resolution. For example, resolution R_3 outperforms R_2 in chunk 8. Also note that R_4 and R_5 needed more chunks to obtain comparable performance than R_2. Our experimental results excluding one resolution at the time showed worse performance, therefore all of them are essential in the overall classification. Clearly, this evidence shows that the MulR representation is in fact very useful.

Sexual Predator Detection					
Test set	R_1	R_2	R_3	R_4	R_5
chunk-1	4	3	2	1	0
chunk-2	3	2	3	2	0
chunk-3	3	2	2	2	1
chunk-4	3	2	3	1	1
chunk-5	3	2	3	1	1
chunk-6	3	1	4	1	1
chunk-7	3	1	3	2	1
chunk-8	2	2	1	2	3
chunk-9	3	1	1	2	3
chunk-10	3	0	2	2	3

Table 6: Post-analysis in test dataset. Distribution of the top ten meta-words according to each resolution R_i at different chunks. We used Information Gain (Hall et al., 2009) to rank meta-words in MulR(TVT).

In Table 6 we provide further evidence about the role of different resolutions. In this complementary analysis we study each chunk at test data. For this we use the MulR learned in training to represent test documents, then we compute the Information Gain using Weka (Hall et al., 2009) at each test chunk. In Table 6 we show the number of features in each resolution R_i that are present in the top ten meta-words of the MulR(TVT). The analysis complements the evidence, lower resolutions have higher IG at early chunks, whereas higher resolutions are more necessary in late chunks.

[6]The only exception to this is R_1, which has the lowest overall performance. This is somewhat expected since this space only has 10 features to represent documents.

7 Conclusions

In this paper we proposed a multi resolution representation that allows to generate multiple "views" of the document. Intuitively these views expose different semantic meanings for words and documents along different resolutions. The different resolutions allow to effectively represent the content of short and large texts at different early stages. The MulR obtained the best results reported so far on the early Sexual Predator Detection task dataset (Inches and Crestani, 2012). For Depression Detection the chunk by chunk evaluation shows promising results for MulR in early stages. What is more, it was shown that the MulR further improves the early recognition performance in the two tasks using different word representations. The relevance of the resolutions in these results is a key factor to understand the proposed MulR and future extensions. These results provide solid evidence to further research on this topic and encourage researchers to apply and evaluate the usefulness of multi-resolution features for other related *early* tasks.

Acknowledgments

This research was partially supported by CONACYT-México (project FC-2016/2410).

References

Douglas Baker and Andrew Kachites McCallum. 1998. Distributional Clustering of Words for Text Classification. In *Proceedings of the 21st ACM International Conference on Research and Development in Information Retrieval*. pages 96–103.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research* 3:993–1022.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai.

1992. Class-based n-gram models of natural language. *Comput. Linguist.* 18(4):467–479.

S. Deerwester, S. Dumais, T. Landauer, G. Furnas, and R. Harshman. 1990. Indexing by Latent Semantic Analysis. *Journal of the American Society of Information Science* 41(6):391–407.

G. Dulac-Arnold, L. Denoyer, and P. Gallinari. 2011. Text classification: A sequential reading approach. In *Advances in Information Retrieval, Proc. of 33rd European Conference on IR Research, (ECIR'11)*. Springers, volume 6611 of *LNCS*, pages 411–423.

Marcelo L. Errecalde, Ma. Paula Villegas, Dario G. Funez, Ma. José Garciarena Ucelay, and Leticia C. Cagnina. 2017. Temporal variation of terms as concept space for early risk prediction. In *CLEF (Working Notes)*.

H. J. Escalante, A. Juarez, E. Villatoro, M. Montes-y-Gómez, and L. Villasenor. 2013. Sexual predator detection in chats with chained classifiers. In *Proceedings of NAACL-HLT 2013, 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. page 46.

H. J. Escalante, M. Montes-y-Gómez, L. Villasenor, and M. L. Errecalde. 2016. Early text classification: a naive solution. In *Proceedings of NAACL-HLT 2016, 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. pages 91–99.

H. J. Escalante, E. Villatoro-Tello, S. E. Garza, A. P. López-Monroy, M. Montes-y-Gómez, and L. Villaseñor-Pineda. 2017. Early detection of deception and aggressiveness using profile-based representations. *Expert Systems with Applications* 89(Supplement C):99 – 111.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P Reutemann, and I. Witten. 2009. The weka data mining software: An update. *SIGKDD Explorations* 11.

G. Inches and F. Crestani. 2012. Overview of the international sexual predator identification competition at pan-2012. In *CEUR Workshop Proceedings, Working Notes for CLEF 2012 Conference*. CEUR, volume 1178.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Machine Learning: ECML-98*, Springer Berlin Heidelberg, volume 1398 of *Lecture Notes in Computer Science*, pages 137–142.

Alberto Lavelli, Fabrizio Sebastiani, and Roberto Zanoli. 2004. Distributional term representations: an experimental comparison. In *CIKM*. ACM, pages 615–624.

Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. 2006. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. volume 2, pages 2169–2178.

David D. Lewis. 1992. An Evaluation of Phrasal and Clustered Representations on a Text Categorization Task. In *Proceedings of the 15th International ACM/SIGIR Conference on Research & Development in Information Retrieval*. June, pages 37–50.

Yong Li and Anil Jain. 1998. Classification of Text Documents. *The Computer Journal* 41(8):537 –546.

A. Pastor López-Monroy, Manuel Montes-y-Gómez, Hugo Jair Escalante, Luis Villasenor-Pineda, and Efstathios Stamatatos. 2015. Discriminative subprofile-specific representations for author profiling in social media. *Knowledge-based Systems* 89:134–147.

David E. Losada, Fabio Crestani, and Javier Parapar. 2017. *eRISK 2017: CLEF Lab on Early Risk Prediction on the Internet: Experimental Foundations*, Springer International Publishing, Cham, pages 346–360.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of Advances in Neural Information Processing Systems 26 (NIPS 2013)*. pages 1–9.

J. Sivic and A. Zisserman. 2004. Video data mining using configurations of viewpoint invariant regions. In *CVPR*. IEEE, volume 1, pages I–488.

Noam Slonim and Naftali Tishby. 2001. The Power of Word Clusters for Text Classification. In *Proceedings of the 23rd European Colloquium on Information Retrieval Research*. volume 1, pages 1–12.

Esaú Villatoro-Tello, Antonio Juárez-González, Hugo Jair Escalante, Manuel Montes-y Gómez, and Luis Villaseñor Pineda. 2012. A two-step approach for effective detection of misbehaving users in chats. In *CLEF (Online Working Notes/Labs/Workshop)*.

Multinomial Adversarial Networks for Multi-Domain Text Classification

Xilun Chen
Department of Computer Science
Cornell Unversity
Ithaca, NY, 14853, USA
xlchen@cs.cornell.edu

Claire Cardie
Department of Computer Science
Cornell Unversity
Ithaca, NY, 14853, USA
cardie@cs.cornell.edu

Abstract

Many text classification tasks are known to be highly domain-dependent. Unfortunately, the availability of training data can vary drastically across domains. Worse still, for some domains there may not be any annotated data at all. In this work, we propose a *multinomial adversarial network*[1] (MAN) to tackle this real-world problem of multi-domain text classification (MDTC) in which labeled data may exist for multiple domains, but in insufficient amounts to train effective classifiers for one or more of the domains. We provide theoretical justifications for the MAN framework, proving that different instances of MANs are essentially minimizers of various f-divergence metrics (Ali and Silvey, 1966) among *multiple* probability distributions. MANs are thus a theoretically sound generalization of traditional adversarial networks that discriminate over *two* distributions. More specifically, for the MDTC task, MAN learns features that are invariant across multiple domains by resorting to its ability to reduce the divergence among the feature distributions of each domain. We present experimental results showing that MANs significantly outperform the prior art on the MDTC task. We also show that MANs achieve state-of-the-art performance for domains with no labeled data.

1 Introduction

Text classification is one of the most fundamental tasks in Natural Language Processing, and has found its way into a wide spectrum of NLP applications, ranging from email spam detection and social media analytics to sentiment analysis and data mining. Over the past couple of decades, supervised statistical learning methods have become the dominant approach for text classification (e.g. McCallum et al. (1998); Kim (2014); Iyyer et al. (2015)). Unfortunately, many text classification tasks are highly domain-dependent in that a text classifier trained using labeled data from one domain is likely to perform poorly on another. In the task of sentiment classification, for example, the phrase "runs fast" is usually associated with positive sentiment in the sports domain; not so when a user is reviewing the battery of an electronic device. In real applications, therefore, an adequate amount of training data from each domain of interest is typically required, and this is expensive to obtain.

Two major lines of work attempt to tackle this challenge: **domain adaptation** (Blitzer et al., 2007) and **multi-domain text classification** (MDTC) (Li and Zong, 2008). In domain adaptation, the assumption is that there is some domain with abundant training data (the source domain), and the goal is to utilize knowledge learned from the source domain to help perform classifications on another lower-resourced target domain.[2] The focus of this work, MDTC, instead simulates an arguably more realistic scenario, where labeled data may exist for multiple domains, but in insufficient amounts to train an effective classifier for one or more of the domains. Worse still, some domains may have *no* labeled data at all. The objective of MDTC is to leverage *all* the available resources in order to improve the system performance over all domains simultaneously.

One state-of-the-art system for MDTC, the CMSC system of Wu and Huang (2015), combines a classifier that is shared across all domains (for learning domain-invariant knowledge) with a set of classifiers, one per domain, each of which captures domain-specific text classification knowledge. This paradigm is sometimes known

[1] The source code of MAN is available at https://github.com/ccsasuke/man.

[2] See §6 for other variants of domain adaptation.

Proceedings of NAACL-HLT 2018, pages 1226–1240
New Orleans, Louisiana, June 1 - 6, 2018.

as the Shared-Private model (Bousmalis et al., 2016). CMSC, however, lacks an explicit mechanism to ensure that the shared classifier captures only domain-independent knowledge: the shared classifier may well also acquire some domain-specific features that are useful for a subset of the domains. We hypothesize that better performance can be obtained if this constraint were explicitly enforced.

In this paper, we thus propose Multinomial Adversarial Networks (henceforth, MANs) for the task of multi-domain text classification. In contrast to standard adversarial networks (Goodfellow et al., 2014), which serve as a tool for minimizing the divergence between *two* distributions (Nowozin et al., 2016), MANs represent a family of theoretically sound adversarial networks that, in contrast, leverage a *multinomial discriminator* to directly minimize the divergence among multiple probability distributions. And just as binomial adversarial networks have been applied to numerous tasks (e.g. image generation (Goodfellow et al., 2014), domain adaptation (Ganin et al., 2016), cross-lingual text classification (Chen et al., 2016)), we anticipate that MANs will make a versatile machine learning framework with applications beyond the MDTC task studied in this work.

We introduce the MAN architecture in §2 and prove in §3 that it directly minimizes the (generalized) f-divergence among multiple distributions so that they are indistinguishable upon successful training. Specifically for MDTC, MAN is used to overcome the aforementioned limitation in prior art where domain-specific features may sneak into the shared model. This is accomplished by relying on MAN's power of minimizing the divergence among the feature distributions of each domain. The high-level idea is that MAN will make the extracted feature distributions of each domain indistinguishable from one another, thus learning general features that are invariant across domains.

We then validate the effectiveness of MAN in experiments on two MDTC data sets. We find first that MAN significantly outperforms the state-of-the-art CMSC method (Wu and Huang, 2015) on the widely used multi-domain Amazon review dataset, and does so without relying on external resources such as sentiment lexica (§4.1). When applied to the second dataset, FDU-MTL (§4.3), we obtain similar results: MAN achieves substantially higher accuracy than the previous top-performing method, ASP-MTL (Liu et al., 2017). ASP-MTL is the first empirical attempt to use a multinomial adversarial network for multi-task learning, but is more restricted and can be viewed as a special case of MAN. In addition, we provide the first theoretical guarantees for multinomial adversarial networks (§3). Finally, while many MDTC methods such as CMSC require labeled data for each domain, MANs can be applied in cases where no labeled data exists for a subset of domains. To evaluate MAN in this semi-supervised setting, we compare MAN to a method that can accommodate unlabeled data for (only) one domain (Zhao et al., 2017), and show that MAN achieves performance comparable to the state of the art (§4.2).

2 Model

In this paper, we strive to tackle the text classification problem in the real-world setting in which texts come from a variety of domains, each with a varying amount of labeled data. Specifically, assume we have a total of N domains, N_1 *labeled domains* (denoted as Δ_L) for which there is some labeled data, and N_2 *unlabeled domains* (Δ_U) for which no annotated training instances are available. Denote $\Delta = \Delta_L \cup \Delta_U$ as the collection of all domains, with $N = N_1 + N_2$. The goal of this work, and of MDTC in general, is to improve the overall classification performance across all N domains, measured in this paper as the average[3] classification accuracy across the N domains in Δ.

2.1 Model Architecture

As shown in Figure 1, the Multinomial Adversarial Network (MAN) adopts the Shared-Private paradigm of Bousmalis et al. (2016) and consists of four components: a *shared feature extractor* $\mathcal{F}_s$, a *domain feature extractor* $\mathcal{F}_{d_i}$ for each labeled domain $d_i \in \Delta_L$, a *text classifier* $\mathcal{C}$, and a *domain discriminator* $\mathcal{D}$. The main idea of MAN is to explicitly model the domain-invariant features that are beneficial to the main classification task across all domains (i.e. the *shared features*, extracted by $\mathcal{F}_s$), as well as the domain-specific features that mainly contribute to the classification in its own domain (the *domain features*, extracted by $\mathcal{F}_d$). Here, the adversarial domain discriminator $\mathcal{D}$ has a multinomial output that takes a shared feature

[3] In this work, we use macro-average over domains, but MAN can be readily adapted for micro-average or other (weighted) averaging schemes.

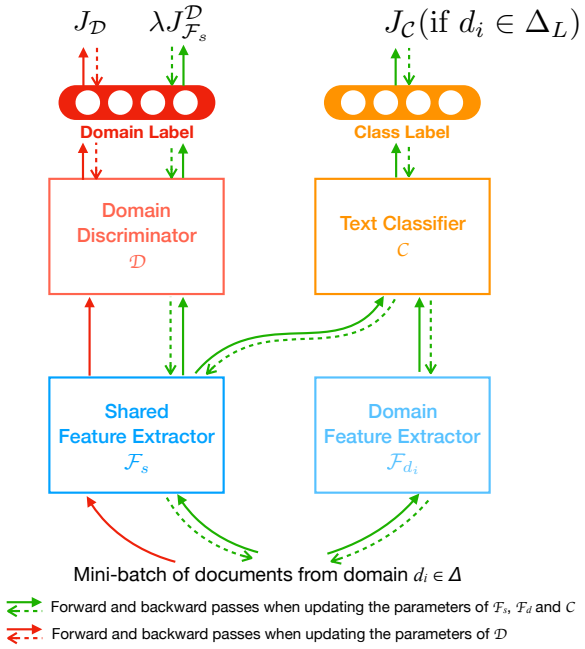


Figure 1: MAN for MDTC. The figure demonstrates the training on a mini-batch of data from one domain. One training iteration consists of one such mini-batch training from each domain. The parameters of $\mathcal{F}_s, \mathcal{F}_d, \mathcal{C}$ are updated together, and the training flows are illustrated by the green arrows. The parameters of $\mathcal{D}$ are updated separately, shown in red arrows. Solid lines indicate forward passes while dotted lines are backward passes. $J^{\mathcal{D}}_{\mathcal{F}_s}$ is the domain loss for $\mathcal{F}_s$, which is anticorrelated with $J_{\mathcal{D}}$ (e.g. $J^{\mathcal{D}}_{\mathcal{F}_s} = -J_{\mathcal{D}}$). (See §2,§3)

vector and predicts the likelihood of that sample coming from each domain. As seen in Figure 1, during the training of $\mathcal{F}_s$ (green arrows denote the training flow), $\mathcal{F}_s$ aims to confuse $\mathcal{D}$ by minimizing $J^{\mathcal{D}}_{\mathcal{F}_s}$, which is anticorrelated to $J_{\mathcal{D}}$ (detailed in §2.2), so that $\mathcal{D}$ cannot predict the domain of a sample given its shared features. The intuition is that if even a strong discriminator $\mathcal{D}$ cannot tell the domain of a sample from the extracted features, those features $\mathcal{F}_s$ learned are essentially domain invariant. By enforcing domain-invariant features to be learned by $\mathcal{F}_s$, when trained jointly via backpropagation, the set of domain feature extractors $\mathcal{F}_d$ will each learn domain-specific features beneficial within its own domain.

The architecture of each component is relatively flexible, and can be decided by the practitioners to suit their particular classification tasks. For instance, the feature extractors can adopt the form of Convolutional Neural Nets (CNN), Recurrent Neural Nets (RNN), or a Multi-Layer Perceptron (MLP), depending on the input data (see §4). The input of MAN will also be dependent on the feature

Algorithm 1 MAN Training

Require: labeled corpus $\mathbb{X}$; unlabeled corpus $\mathbb{U}$; Hyperparameter $\lambda > 0$, $k \in \mathbb{N}$

repeat
  ▷ $\mathcal{D}$ iterations
  for $diter = 1$ to k **do**
    $l_{\mathcal{D}} = 0$
    for all $d \in \Delta$ **do** ▷ For all N domains
      Sample a mini-batch $\boldsymbol{x} \sim \mathbb{U}_d$
      $\boldsymbol{f}_s = \mathcal{F}_s(\boldsymbol{x})$ ▷ Shared feature vector
      $l_{\mathcal{D}}$ += $J_{\mathcal{D}}(\mathcal{D}(\boldsymbol{f}_s); d)$ ▷ Accumulate $\mathcal{D}$ loss
    Update $\mathcal{D}$ parameters using $\nabla l_{\mathcal{D}}$
  ▷ Main iteration
  $loss = 0$
  for all $d \in \Delta_L$ **do** ▷ For all labeled domains
    Sample a mini-batch $(\boldsymbol{x}, \boldsymbol{y}) \sim \mathbb{X}_d$
    $\boldsymbol{f}_s = \mathcal{F}_s(\boldsymbol{x})$
    $\boldsymbol{f}_d = \mathcal{F}_d(\boldsymbol{x})$ ▷ Domain feature vector
    $loss$ += $J_{\mathcal{C}}(\mathcal{C}(\boldsymbol{f}_s, \boldsymbol{f}_d); \boldsymbol{y})$ ▷ Compute $\mathcal{C}$ loss
  for all $d \in \Delta$ **do** ▷ For all N domains
    Sample a mini-batch $\boldsymbol{x} \sim \mathbb{U}_d$
    $\boldsymbol{f}_s = \mathcal{F}_s(\boldsymbol{x})$
    $loss$ += $\lambda \cdot J^{\mathcal{D}}_{\mathcal{F}_s}(\mathcal{D}(\boldsymbol{f}_s); d)$ ▷ Domain loss of $\mathcal{F}_s$
  Update $\mathcal{F}_s, \mathcal{F}_d, \mathcal{C}$ parameters using $\nabla loss$
until convergence

extractor choice. The output of a (shared/domain) feature extractor is a fixed-length vector, which is considered the (shared/domain) hidden features of some given input text. On the other hand, the outputs of $\mathcal{C}$ and $\mathcal{D}$ are label probabilities for class and domain prediction, respectively. For example, both $\mathcal{C}$ and $\mathcal{D}$ can be MLPs with a softmax layer on top. In §3, we provide alternative architectures for $\mathcal{D}$ and their mathematical implications. We now present a detailed description of the MAN training in §2.2 as well as the theoretical grounds in §3.

2.2 Training

Denote the annotated corpus in a labeled domain $d_i \in \Delta_L$ as $\mathbb{X}_i$; and $(x, y) \sim \mathbb{X}_i$ is a sample drawn from the labeled data in domain d_i, where x is the input and y is the task label. On the other hand, for any domain $d_{i'} \in \Delta$, denote the unlabeled corpus as $\mathbb{U}_{i'}$. Note for the choice of unlabeled data of a labeled domain, one can use a separate unlabeled corpus or simply use the labeled data (or use both).

In Figure 1, the arrows illustrate the training flows of various components. Due to the adversarial nature of the domain discriminator $\mathcal{D}$, it is trained with a separate optimizer (red arrows), while the rest of the networks are updated with the main optimizer (green arrows). $\mathcal{C}$ is only trained on the annotated data from labeled domains, and it takes as input the concatenation of the shared and domain feature vectors. At test time, for data from

unlabeled domains with no $\mathcal{F}_d$, the domain features are set to the $\mathbf{0}$ vector for $\mathcal{C}$'s input. On the contrary, $\mathcal{D}$ only takes the shared features as input, for both labeled and unlabeled domains. The MAN training procedure is described in Algorithm 1.

In Algorithm 1, $\mathcal{L}_\mathcal{C}$ and $\mathcal{L}_\mathcal{D}$ are the loss functions of the text classifier $\mathcal{C}$ and the domain discriminator $\mathcal{D}$, respectively. As mentioned in §2.1, $\mathcal{C}$ has a $softmax$ layer on top for classification. We hence adopt the canonical negative log-likelihood (NLL) loss:

$$\mathcal{L}_\mathcal{C}(\hat{y}, y) = -\log P(\hat{y} = y) \tag{1}$$

where y is the true label and $\hat{y}$ is the $softmax$ predictions. For $\mathcal{D}$, we consider two variants of MAN. The first one is to use the NLL loss same as $\mathcal{C}$ which suits the classification task; while another option is to use the Least-Square (L2) loss that was shown to be able to alleviate the gradient vanishing problem when using the NLL loss in the adversarial setting (Mao et al., 2017):

$$\mathcal{L}_\mathcal{D}^{NLL}(\hat{d}, d) = -\log P(\hat{d} = d) \tag{2}$$

$$\mathcal{L}_\mathcal{D}^{L2}(\hat{d}, d) = \sum_{i=1}^{N} (\hat{d}_i - \mathbb{1}_{\{d=i\}})^2 \tag{3}$$

where d is the domain index of some sample and $\hat{d}$ is the prediction. Without loss of generality, we normalize $\hat{d}$ so that $\sum_{i=1}^{N} \hat{d}_i = 1$ and $\forall i : \hat{d}_i \geq 0$.

Therefore, the objectives of $\mathcal{C}$ and $\mathcal{D}$ that we are minimizing are:

$$J_\mathcal{C} = \sum_{i=1}^{N} \mathop{\mathbb{E}}_{(x,y)\sim\mathbb{X}_i} [\mathcal{L}_\mathcal{C}(\mathcal{C}(\mathcal{F}_s(x), \mathcal{F}_d(x)); y)] \tag{4}$$

$$J_\mathcal{D} = \sum_{i=1}^{N} \mathop{\mathbb{E}}_{x\sim\mathbb{U}_i} [\mathcal{L}_\mathcal{D}(\mathcal{D}(\mathcal{F}_s(x)); d)] \tag{5}$$

For the feature extractors, the training of domain feature extractors is straightforward, as their sole objective is to help $\mathcal{C}$ perform better within their own domain. Hence, $J_{\mathcal{F}_d} = J_\mathcal{C}$ for any domain d. Finally, the shared feature extractor $\mathcal{F}_s$ has two objectives: to help $\mathcal{C}$ achieve higher accuracy, and to make the feature distribution invariant across all domains. It thus leads to the following bipartite loss:

$$J_{\mathcal{F}_s} = J_{\mathcal{F}_s}^\mathcal{C} + \lambda \cdot J_{\mathcal{F}_s}^\mathcal{D}$$

where λ is a hyperparameter balancing the two parts. $J_{\mathcal{F}_s}^\mathcal{D}$ is the domain loss of $\mathcal{F}_s$ anticorrelated to $J_\mathcal{D}$:

$$(NLL) J_{\mathcal{F}_s}^\mathcal{D} = -J_\mathcal{D} \tag{6}$$

$$(L2) J_{\mathcal{F}_s}^\mathcal{D} = \sum_{i=1}^{N} \mathop{\mathbb{E}}_{x\sim\mathbb{U}_i} \left[\sum_{j=1}^{N} (\mathcal{D}_j(\mathcal{F}_s(x)) - \frac{1}{N})^2 \right] \tag{7}$$

If $\mathcal{D}$ adopts the NLL loss (6), the domain loss is simply $-J_\mathcal{D}$. For the L2 loss (7), $J_{\mathcal{F}_s}^\mathcal{D}$ intuitively translates to pushing $\mathcal{D}$ to make random predictions. See §3 for theoretical justifications.

3 Theories of Multinomial Adversarial Networks

Binomial adversarial nets are known to have theoretical connections to the minimization of various f-divergences[4] between *two* distributions (Nowozin et al., 2016). However, for adversarial training among multiple distributions, no theoretical justifications have been provided to our best knowledge, despite that this idea has recently been explored empirically (Liu et al., 2017).

In this section, we present a theoretical analysis showing the validity of MAN. In particular, we show that MAN's objective is equivalent to minimizing the total f-divergence between each of the shared feature distributions of the N domains, and the centroid of the N distributions. The choice of loss function will determine which specific f-divergence is minimized. Furthermore, with adequate model capacity, MAN achieves its optimum for either loss function if and only if all N shared feature distributions are identical, hence learning an invariant feature space across all domains.

First, consider the distribution of the shared features $\mathbf{f}$ for instances in each domain $d_i \in \Delta$:

$$P_i(\mathbf{f}) \triangleq P(\mathbf{f} = \mathcal{F}_s(x) | x \in d_i) \tag{8}$$

Combining (5) with the two loss functions (2), (3), the objective of $\mathcal{D}$ can be written as:

$$J_\mathcal{D}^{NLL} = -\sum_{i=1}^{N} \mathop{\mathbb{E}}_{\mathbf{f}\sim P_i} [\log \mathcal{D}_i(\mathbf{f})] \tag{9}$$

$$J_\mathcal{D}^{L2} = \sum_{i=1}^{N} \mathop{\mathbb{E}}_{\mathbf{f}\sim P_i} \left[\sum_{j=1}^{N} (\mathcal{D}_j(\mathbf{f}) - \mathbb{1}_{\{i=j\}})^2 \right] \tag{10}$$

[4] An f-divergence (Ali and Silvey, 1966) is a function that measures the distance between two probability distributions, e.g. the KL or Jensen-Shannon divergence.

where $\mathcal{D}_i(\mathbf{f})$ is the i-th dimension of $\mathcal{D}$'s (normalized) output vector, which conceptually corresponds to the probability of $\mathcal{D}$ predicting that $\mathbf{f}$ is from domain d_i

We first derive the optimal $\mathcal{D}$ for any fixed $\mathcal{F}_s$.

Lemma 1. *For any fixed $\mathcal{F}_s$, with either NLL or L2 loss, the optimum domain discriminator $\mathcal{D}^*$ is:*

$$\mathcal{D}_i^*(\mathbf{f}) = \frac{P_i(\mathbf{f})}{\sum_{j=1}^{N} P_j(\mathbf{f})} \tag{11}$$

The proof involves an application of the *Lagrangian Multiplier* to solve the minimum value of $J_{\mathcal{D}}$, and the details can be found in the Appendix. We then have the following main theorems for the domain loss for $\mathcal{F}_s$:

Theorem 1. *Let $\overline{P} = \frac{\sum_{i=1}^{N} P_i}{N}$. When $\mathcal{D}$ is trained to its optimality, if $\mathcal{D}$ adopts the* NLL loss*:*

$$\begin{aligned} J_{\mathcal{F}_s}^{\mathcal{D}} &= -\min_{\theta_{\mathcal{D}}} J_{\mathcal{D}} = -J_{\mathcal{D}^*} \\ &= -N\log N + N \cdot JSD(P_1, P_2, \ldots, P_N) \\ &= -N\log N + \sum_{i=1}^{N} KL(P_i \| \overline{P}) \end{aligned}$$

where $JSD(\cdot)$ is the generalized Jensen-Shannon Divergence (Lin, 1991) among multiple distributions, defined as the average Kullback-Leibler divergence of each P_i to the centroid $\overline{P}$ (Aslam and Pavlu, 2007).

Theorem 2. *If $\mathcal{D}$ uses the* L2 loss*:*

$$\begin{aligned} J_{\mathcal{F}_s}^{\mathcal{D}} &= \sum_{i=1}^{N} \mathop{\mathbb{E}}_{\mathbf{f}\sim P_i} \left[\sum_{j=1}^{N} (\mathcal{D}_j^*(\mathbf{f}) - \frac{1}{N})^2 \right] \\ &= \frac{1}{N} \sum_{i=1}^{N} \chi^2_{Neyman}(P_i \| \overline{P}) \end{aligned}$$

where $\chi^2_{Neyman}(\cdot\|\cdot)$ is the Neyman χ^2 divergence (Nielsen and Nock, 2014). The proof of both theorems can be found in the Appendix.

Consequently, by the non-negativity and joint convexity of the f-divergence (Csiszar and Korner, 1982), we have:

Corollary 1. *The optimum of $J_{\mathcal{F}_s}^{\mathcal{D}}$ is $-N\log N$ when using NLL loss, and 0 for the L2 loss. The optimum value above is achieved if and only if $P_1 = P_2 = \cdots = P_N = \overline{P}$ for either loss.*

Therefore, the loss of $\mathcal{F}_s$ can be interpreted as simultaneously minimizing the classification loss $J_{\mathcal{C}}$ as well as the divergence among feature distributions of all domains. It can thus learn a shared feature mapping that is invariant across domains upon successful training while being beneficial to the main classification task.

	Book	DVD	Elec.	Kit.	Avg.
Domain-Specific Models Only					
LS	77.80	77.88	81.63	84.33	80.41
SVM	78.56	78.66	83.03	84.74	81.25
LR	79.73	80.14	84.54	86.10	82.63
MLP	81.70	81.65	85.45	85.95	83.69
Shared Model Only					
LS	78.40	79.76	84.67	85.73	82.14
SVM	79.16	80.97	85.15	86.06	82.83
LR	80.05	81.88	85.19	86.56	83.42
MLP	82.40	82.15	85.90	88.20	84.66
MAN-L2-MLP	82.05	83.45	86.45	88.85	85.20
MAN-NLL-MLP	81.85	83.10	85.75	89.10	84.95
Shared-Private Models					
RMTL[1]	81.33	82.18	85.49	87.02	84.01
MTLGraph[2]	79.66	81.84	83.69	87.06	83.06
CMSC-LS[3]	82.10	82.40	86.12	87.56	84.55
CMSC-SVM[3]	82.26	83.48	86.76	88.20	85.18
CMSC-LR[3]	81.81	83.73	86.67	88.23	85.11
SP-MLP	82.00	**84.05**	86.85	87.30	85.05
MAN-L2-SP-MLP	82.46 (±0.25)	83.98 (±0.17)	**87.22*** (±0.04)	88.53 (±0.19)	85.55* (±0.07)
MAN-NLL-SP-MLP	**82.98*** (±0.28)	84.03 (±0.16)	87.06 (±0.23)	**88.57*** (±0.15)	**85.66*** (±0.14)

[1] Evgeniou and Pontil (2004)
[2] Zhou et al. (2011)
[3] Wu and Huang (2015)

Table 1: MDTC results on the Amazon dataset. Models in bold are ours while the performance of the rest is taken from Wu and Huang (2015). Numbers in parentheses indicate standard errors, calculated based on 5 runs. Bold numbers indicate the highest performance in each domain, and $*$ shows statistical significance ($p < 0.05$) over CMSC under a one-sample T-Test.

4 Experiments

4.1 Multi-Domain Text Classification

In this experiment, we compare MAN to state-of-the-art MDTC systems on the multi-domain Amazon review dataset (Blitzer et al., 2007), which is one of the most widely used MDTC datasets. Note that this dataset was already preprocessed into a bag of features (unigrams and bigrams), losing all word order information. This prohibits the use of CNNs or RNNs as feature extractors, limiting the potential performance of the system. Nonetheless, we adopt the same dataset for fair comparison and employ a MLP as our feature extractor. In particular, we take the 5000 most frequent features and represent each review as a 5000d feature vector, where feature values are raw counts of the fea-

tures. Our MLP feature extractor would then have an input size of 5000 in order to process the reviews.

The Amazon dataset contains 2000 samples for each of the four domains: *book, DVD, electronics*, and *kitchen*, with binary labels (positive, negative). Following Wu and Huang (2015), we conduct 5-way cross validation. Three out of the five folds are treated as the training set, one serves as the validation set, while the remaining is the test set. The 5-fold average test accuracy is reported.

Table 1 shows the main results. Three types of models are shown: *Domain-Specific Models Only*, where only in-domain models are trained[5]; *Shared Model Only*, where a single model is trained with all data; and *Shared-Private Models*, a combination of the previous two. Within each category, various architectures are examined, such as Least Square (LS), SVM, and Logistic Regression (LR). As explained before, we use MLP as our feature extractors for all our models (bold ones). Among our models, the ones with the MAN prefix use adversarial training, and MAN-L2 and MAN-NLL indicate MAN with the L2 loss and the NLL loss, respectively.

From Table 1, we can see that by adopting modern deep neural networks, our methods achieve superior performance within the first two model categories even without adversarial training. This is corroborated by the fact that our SP-MLP model performs comparably to CMSC, while the latter relies on external resources such as sentiment lexica. Moreover, when our multinomial adversarial nets are introduced, further improvement is observed. With both loss functions, MAN outperforms all Shared-Private baseline systems on each domain, and achieves statistically significantly higher overall performance. For our MAN-SP models, we provide the mean accuracy as well as the standard errors over five runs, to illustrate the performance variance and conduct significance tests. It can be seen that MAN's performance is relatively stable, and consistently outperforms CMSC.

4.2 Experiments for Unlabeled Domains

As CMSC requires labeled data for each domain, their experiments were naturally designed this way. In reality, however, many domains may not have any annotated corpora available. It is therefore also important to look at the performance in these unlabeled domains for a MDTC system. Fortunately, as depicted before, MAN's adversarial training only utilizes unlabeled data from each domain to learn the domain-invariant features, and can thus be used on unlabeled domains as well. During testing, only the shared feature vector is fed into $\mathcal{C}$, while the domain feature vector is set to $\mathbf{0}$.

Target Domain	Book	DVD	Elec.	Kit.	Avg.
MLP	76.55	75.88	84.60	85.45	80.46
mSDA[1]	76.98	78.61	81.98	84.26	80.46
DANN[2]	77.89	78.86	84.91	86.39	82.01
MDAN (H-MAX)[3]	78.45	77.97	84.83	85.80	81.76
MDAN (S-MAX)[3]	**78.63**	80.65	**85.34**	86.26	**82.72**
MAN-L2-SP-MLP	78.45	81.57	83.37	85.57	82.24
MAN-NLL-SP-MLP	77.78	**82.74**	83.75	**86.41**	82.67

[1] Chen et al. (2012)
[2] Ganin et al. (2016)
[3] Zhao et al. (2017)

Table 2: Results on unlabeled domains. Models in bold are our models while the rest is taken from Zhao et al. (2017). Highest domain performance is shown in bold.

In order to validate MAN's effectiveness, we compare to state-of-the-art *multi-source domain adaptation* (MS-DA) methods (see §6). Compared to standard domain adaptation methods with one source and one target domain, MS-DA allows the adaptation from multiple source domains to a single target domain. Analogically, MDTC can be viewed as *multi-source multi-target* domain adaptation, which is superior when multiple target domains exist. With multiple target domains, MS-DA will need to treat each one as an independent task, which is more expensive and cannot utilize the unlabeled data in other target domains.

In this work, we compare MAN with one recent MS-DA method, MDAN (Zhao et al., 2017). Their experiments only have one target domain to suit their approach, and we follow this setting for fair comparison. However, it is worth noting that MAN is designed for the MDTC setting, and can deal with multiple target domains at the same time, which can potentially improve the performance by taking advantage of more unlabeled data from multiple target domains during adversarial training. We adopt the same setting as Zhao et al. (2017), which is based on the same multi-domain Amazon review dataset. Each of the four domains in the dataset is treated as the target domain in four separate experiments, while the re-

[5]For our models, it means $\mathcal{F}_s$ is disabled. Similarly, for Shared Model Only, no $\mathcal{F}_d$ is used.

	books	elec.	dvd	kitchen	apparel	camera	health	music	toys	video	baby	magaz.	softw.	sports	IMDb	MR	Avg.
Domain-Specific Models Only																	
BiLSTM	81.0	78.5	80.5	81.2	86.0	86.0	78.7	77.2	84.7	83.7	83.5	91.5	85.7	84.0	85.0	74.7	82.6
CNN	85.3	87.8	76.3	84.5	86.3	89.0	87.5	81.5	87.0	82.3	82.5	86.8	87.5	85.3	83.3	75.5	84.3
Shared Model Only																	
FS-MTL	82.5	85.7	83.5	86.0	84.5	86.5	88.0	81.2	84.5	83.7	88.0	92.5	86.2	85.5	82.5	74.7	84.7
MAN-L2-CNN	**88.3**	88.3	87.8	88.5	85.3	90.5	**90.8**	85.3	89.5	89.0	89.5	91.3	88.3	89.5	**88.5**	73.8	87.7
MAN-NLL-CNN	88.0	87.8	87.3	88.5	86.3	90.8	89.8	84.8	89.3	89.3	87.8	91.8	90.0	**90.3**	87.3	73.5	87.6
Shared-Private Models																	
ASP-MTL	84.0	86.8	85.5	86.2	87.0	89.2	88.2	82.5	88.0	84.5	88.2	92.2	87.2	85.7	85.5	**76.7**	86.1
MAN-L2-SP-CNN	87.6* (0.2)	87.4 (1.0)	88.1* (0.4)	89.8* (0.4)	**87.6** (0.7)	**91.4*** (0.4)	89.8* (0.3)	**85.9*** (0.1)	90.0* (0.1)	89.5* (0.2)	90.0 (0.6)	92.5 (0.5)	90.4* (0.4)	89.0* (0.4)	86.6 (0.5)	76.1 (0.5)	88.2* (0.1)
MAN-NLL-SP-CNN	86.8* (0.4)	**88.8** (0.6)	**88.6*** (0.4)	**89.9*** (0.4)	**87.6** (0.4)	90.7 (0.4)	89.4 (0.3)	85.5* (0.1)	**90.4*** (0.2)	**89.6*** (0.3)	**90.2** (0.6)	**92.9** (0.4)	**90.9*** (0.7)	89.0* (0.2)	87.0* (0.1)	**76.7** (0.8)	**88.4*** (0.1)

Table 3: Results on the FDU-MTL dataset. Bolded models are ours, while the rest is from Liu et al. (2017). Highest performance is each domain is highlighted. For our full MAN models, standard errors are shown in parenthese and statistical significance ($p < 0.01$) over ASP-MTL is indicated by *.

maining three are used as source domains.

In Table 2, the target domain is shown on top, and the test set accuracy is reported for various systems. It shows that MAN outperforms several baseline systems, such as a MLP trained on the source-domains, as well as single-source domain adaptation methods such as mSDA (Chen et al., 2012) and DANN (Ganin et al., 2016), where the training data in the multiple source domains are combined and viewed as a single domain. Finally, when compared to MDAN, MAN and MDAN each achieves higher accuracy on two out of the four target domains, and the average accuracy of MAN is similar to MDAN. Therefore, MAN achieves competitive performance for the domains without annotated corpus. Nevertheless, unlike MS-DA methods, MAN can handle multiple target domains at one time.

4.3 Experiments on the MTL Dataset

To make fair comparisons, the previous experiments follow the standard settings in the literature, where the widely adopted Amazon review dataset is used. However, this dataset has a few limitations. First, it has only four domains. In addition, the reviews are already tokenized and converted to a bag of features consisting of unigrams and bigrams. Raw review texts are hence not available in this dataset, making it impossible to use certain modern neural architectures such as CNNs and RNNs. To provide more insights on how well MAN works with other feature extractor architectures, we provide a third set of experiments on the FDU-MTL dataset (Liu et al., 2017). This dataset is created as a multi-task learning dataset with 16 *tasks*, where each task is essentially a different domain of reviews. It has 14 Amazon domains: books, electronics, DVD, kitchen, apparel, camera, health, music, toys, video, baby, magazine, software, and sports, in addition to two movie review domains from the IMDb and the MR datasets. Each domain has a development set of 200 samples, and a test set of 400 samples. The amount of training and unlabeled data vary across domains but are roughly 1400 and 2000, respectively.

We compare MAN with ASP-MTL (Liu et al., 2017) on this FDU-MTL dataset. ASP-MTL also adopts adversarial training for learning a shared feature space, and can be viewed as a special case of MAN that adopts the NLL loss (MAN-NLL) and chooses LSTM as their feature extractor. In contrast, we found a CNN-based feature extractor (Kim, 2014) achieves much better accuracy while being ~ 10 times faster. Indeed, as shown in Table 3, with or without adversarial training, our CNN models outperform LSTM ones by a large margin. When used in our MAN framework, we attain the state-of-the-art performance on every domain with a 88.4% overall accuracy, surpassing ASP-MTL by a significant margin of 2.3%.

We hypothesize the reason a LSTM performs much worse than a CNN is its lack of an attention mechanism. In ASP-MTL, only the last hidden unit is taken as the extracted features. While LSTMs are effective for representing the context for each token, it might not be powerful enough for directly encoding the entire document (Bahdanau et al., 2015). Therefore, various attention mechanisms have been introduced on top of the vanilla LSTM to select words (and contexts) most relevant for making the predictions. In our preliminary experiments, we find that a Bi-directional LSTM with the dot-product attention (Luong et al., 2015) yields better performance

than the vanilla LSTM in ASP-MTL. However, it still does not outperform CNN and is much slower. As a result, we conclude that, for text classification tasks, CNN is both effective and efficient in extracting local and higher-level features for making a single categorization.

Finally, we observe that MAN-NLL achieves slightly higher overall performance compared to MAN-L2, providing evidence for the claim in a recent study (Lucic et al., 2017) that the original GAN loss (NLL) may not be inherently inferior to the L2 loss. Moreover, the two variants excel in different domains, suggesting the possibility of further performance gain when using ensemble.

5 Implementation Details

For all three of our experiments, we use $\lambda = 0.05$ and $k = 5$ (See Algorithm 1). For both optimizers, Adam (Kingma and Ba, 2015) is used with learning rate 0.0001. The size of the shared feature vector is set to 128 while that of the domain feature vector is 64. Dropout of $p = 0.4$ is used in all components. $\mathcal{C}$ and $\mathcal{D}$ each has one hidden layer of the same size as their input ($128 + 64$ for $\mathcal{C}$ and 128 for $\mathcal{D}$). ReLU is used as the activation function. Batch normalization (Ioffe and Szegedy, 2015) is used in both $\mathcal{C}$ and $\mathcal{D}$ but not $\mathcal{F}$. We use a batch size of 8.

For our first two experiments on the Amazon review dataset, the MLP feature extractor is used. As described in §4.1, it has an input size of 5000. Two hidden layers are used, with size 1000 and 500, respectively.

For the CNN feature extractor used in the FDU-MTL experiment, a single convolution layer is used. The kernel sizes are 3, 4, and 5, and the number of kernels are 200. The convolution layers take as input the 100d word embeddings of each word in the input sequence. We use *word2vec* word embeddings (Mikolov et al., 2013) trained on a bunch of unlabeled raw Amazon reviews (Blitzer et al., 2007). After convolution, the outputs go through a ReLU layer before fed into a max pooling layer. The pooled output is then fed into a single fully connected layer to be converted into a feature vector of size either 128 or 64. More details of using CNN for text classification can be found in the original paper (Kim, 2014). MAN is implemented using PyTorch (Paszke et al., 2017).

6 Related Work

Multi-Domain Text Classification The MDTC task was first examined by Li and Zong (2008), who proposed to fuse the training data from multiple domains either at the feature level or the classifier level. The prior art of MDTC (Wu and Huang, 2015) decomposes the text classifier into a general one and a set of domain-specific ones. However, the general classifier is learned by parameter sharing and domain-specific knowledge may sneak into it. They also require external resources to help improve accuracy and compute domain similarities.

Domain Adaptation Domain Adaptation attempts to transfer the knowledge from a source domain to a target one, and the traditional form is the *single-source, single-target* (**SS,ST**) adaptation (Blitzer et al., 2006). Another variant is the **SS,MT** adaptation (Yang and Eisenstein, 2015), which tries to simultaneously transfer the knowledge to multiple target domains from a single source. However, it cannot fully take advantage the training data if it comes from multiple source domains. **MS,ST** adaptation (Mansour et al., 2009; Zhao et al., 2017) can deal with multiple source domains but only transfers to a single target domain. Therefore, when multiple target domains exist, they need to treat them as independent problems, which is more expensive and cannot utilize the additional unlabeled data in these domains. Finally, MDTC can be viewed as **MS,MT** adaptation, which is arguably more general and realistic.

Adversarial Networks The idea of adversarial networks was proposed by Goodfellow et al. (2014) for image generation, and has been applied to various NLP tasks as well (Chen et al., 2016; Yu et al., 2017). Ganin et al. (2016) first used it for the **SS,ST** domain adaptation followed by many others. Bousmalis et al. (2016) utilized adversarial training in a shared-private model for domain adaptation to learn domain-invariant features, but still focused on the **SS,ST** setting. Finally, the idea of using adversarial nets to discriminate over multiple distributions was empirically explored by a very recent work (Liu et al., 2017) under the multi-task learning setting, and can be considered as a special case of our MAN framework with the NLL domain loss. We propose MAN as a more general framework with alternative architectures for the adversarial component, and for the first time

provide theoretical justifications the multinomial adversarial nets. Moreover, Liu et al. (2017) used a LSTM without attention as their feature extractor, which we found to perform sub-optimal in the experiments. We instead chose Convolutional Neural Nets as our feature extractor that achieves higher accuracy while running an order of magnitude faster (see §4.3).

7 Conclusion

In this work, we propose a family of Multinomial Adversarial Networks (MANs) that generalize the traditional binomial adversarial nets in the sense that MAN can simultaneously minimize the difference among multiple probability distributions instead of just two. We provide theoretical justifications for two instances of MAN, MAN-NLL and MAN-L2, showing they are minimizers of two different f-divergence metrics among multiple distributions, respectively. This indicates MAN can be used to make multiple distributions indistinguishable from one another. It can hence be applied to a variety of tasks, similar to the versatile binomial adversarial nets, which have been used in many areas for making *two* distributions alike.

In this paper, we design a MAN model for the MDTC task, following the shared-private paradigm that has a shared feature extractor to learn domain-invariant features and domain feature extractors to learn domain-specific ones. MAN is used to enforce the shared feature extractor to learn only domain-invariant knowledge, by resorting to MAN's power of making indistinguishable the shared feature distributions of samples from each domain. We conduct extensive experiments, demonstrating our MAN model outperforms the prior art systems in MDTC, and achieves state-of-the-art performance on domains without labeled data when compared to multi-source domain adaptation methods.

Acknowledgments

This work was supported in part by NSF grant SES-1741441 and DARPA DEFT Grant FA8750-13-2-0015. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF, DARPA or the U.S. Government. We also thank Yun Liu, Tianze Shi, Xun Huang, and the anonymous reviewers for their helpful feedback and/or discussions.

References

S. M. Ali and S. D. Silvey. 1966. A general class of coefficients of divergence of one distribution from another. *Journal of the Royal Statistical Society. Series B (Methodological)* 28(1):131–142. http://www.jstor.org/stable/2984279.

Javed A. Aslam and Virgil Pavlu. 2007. Query hardness estimation using jensen-shannon divergence among multiple scoring functions. In *Advances in Information Retrieval*. Springer Berlin Heidelberg, Berlin, Heidelberg, pages 198–209. https://doi.org/10.1007/978-3-540-71496-5_20.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations (ICLR 2015)*. http://arxiv.org/abs/1409.0473.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Association for Computational Linguistics, Prague, Czech Republic, pages 440–447. http://www.aclweb.org/anthology/P07-1056.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Sydney, Australia, pages 120–128. http://www.aclweb.org/anthology/W/W06/W06-1615.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain separation networks. In *Advances in Neural Information Processing Systems 29*, Curran Associates, Inc., pages 343–351. http://papers.nips.cc/paper/6254-domain-separation-networks.pdf.

Minmin Chen, Zhixiang Xu, Kilian Q. Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, Omnipress, USA, ICML'12, pages 1627–1634. http://dl.acm.org/citation.cfm?id=3042573.3042781.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2016. Adversarial deep averaging networks for cross-lingual sentiment classification. *Computing Research Repository* arXiv:1606.01614. https://arxiv.org/abs/1606.01614.

Imre Csiszar and Janos Korner. 1982. *Information Theory: Coding Theorems for Discrete Memoryless Systems*. Academic Press, Inc., Orlando, FL, USA. https://dl.acm.org/citation.cfm?id=601016.

Theodoros Evgeniou and Massimiliano Pontil. 2004. Regularized multi–task learning. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA, KDD '04, pages 109–117. https://doi.org/10.1145/1014052.1014067.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research* 17:1–35. http://dl.acm.org/citation.cfm?id=2946645.2946704.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, Curran Associates, Inc., pages 2672–2680. http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*. pages 448–456. http://jmlr.org/proceedings/papers/v37/ioffe15.pdf.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 1681–1691. https://doi.org/10.3115/v1/P15-1162.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 1746–1751. https://doi.org/10.3115/v1/D14-1181.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations 2015*. https://arxiv.org/abs/1412.6980.

Shoushan Li and Chengqing Zong. 2008. Multi-domain sentiment classification. In *Proceedings of ACL-08: HLT, Short Papers*. Association for Computational Linguistics, Columbus, Ohio, pages 257–260. http://www.aclweb.org/anthology/P/P08/P08-2065.

Jianhua Lin. 1991. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory* 37(1):145–151. https://doi.org/10.1109/18.61115.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1–10. https://doi.org/10.18653/v1/P17-1001.

Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. 2017. Are GANs Created Equal? A Large-Scale Study. *Computing Research Repository* arXiv:1711.10337. http://arxiv.org/abs/1711.10337.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1412–1421. https://doi.org/10.18653/v1/D15-1166.

Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. 2009. Domain adaptation with multiple sources. In *Advances in Neural Information Processing Systems 21*, Curran Associates, Inc., pages 1041–1048. http://papers.nips.cc/paper/3550-domain-adaptation-with-multiple-sources.pdf.

Xudong Mao, Qing Li, Haoran Xie, Raymond Y.K. Lau, Zhen Wang, and Stephen Paul Smolley. 2017. Least squares generative adversarial networks. In *IEEE International Conference on Computer Vision (ICCV)*. pages 2813–2821. https://doi.org/10.1109/ICCV.2017.304.

Andrew McCallum, Kamal Nigam, et al. 1998. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*. Madison, WI, USA, volume 752, pages 41–48. http://www.aaai.org/Papers/Workshops/1998/WS-98-05/WS98-05-007.pdf.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations 2013 Workshop*. https://arxiv.org/abs/1301.3781.

Frank Nielsen and Richard Nock. 2014. On the chi square and higher-order chi distances for approximating f-divergences. *IEEE Signal Processing Letters* 21(1):10–13. https://doi.org/10.1109/LSP.2013.2288355.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. 2016. f-GAN: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc., pages 271–279. http://papers.nips.cc/paper/6066-f-gan-training-generative-neural-samplers-using-variational-divergence-minimization.pdf.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. *NIPS 2017 Autodiff Workshop* https://openreview.net/pdf?id=BJJsrmfCZ.

Fangzhao Wu and Yongfeng Huang. 2015. Collaborative multi-domain sentiment classification. In *2015 IEEE International Conference on Data Mining*. pages 459–468. https://doi.org/10.1109/ICDM.2015.68.

Yi Yang and Jacob Eisenstein. 2015. Unsupervised multi-domain adaptation with feature embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 672–682. https://doi.org/10.3115/v1/N15-1069.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI Conference on Artificial Intelligence*. https://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14344.

Han Zhao, Shanghang Zhang, Guanhang Wu, João P. Costeira, José M. F. Moura, and Geoffrey J. Gordon. 2017. Multiple source domain adaptation with adversarial training of neural networks. *Computing Research Repository* arXiv:1705.09684. http://arxiv.org/abs/1705.09684.

J. Zhou, J. Chen, and J. Ye. 2011. *MALSAR: Multi-tAsk Learning via StructurAl Regularization*. Arizona State University. http://www.public.asu.edu/~jye02/Software/MALSAR.

Appendix A Proofs

A.1 Proofs for MAN-NLL

Assume we have N domains, consider the distribution of the shared features $\mathcal{F}_s$ for instances in each domain d_i:

$$P_i(\mathbf{f}) \triangleq P(\mathbf{f} = \mathcal{F}_s(x) | x \in d_i)$$

The objective that $\mathcal{D}$ attempts to minimize is:

$$J_{\mathcal{D}} = -\sum_{i=1}^{N} \mathop{\mathbb{E}}_{\mathbf{f} \sim P_i} [\log \mathcal{D}_i(\mathbf{f})] \tag{12}$$

where $\mathcal{D}_i(\mathbf{f})$ is the i-th dimension of $\mathcal{D}$'s output vector, which conceptually corresponds to the softmax probability of $\mathcal{D}$ predicting that $\mathbf{f}$ is from domain d_i. We therefore have property that for any $\mathbf{f}$:

$$\sum_{i=1}^{N} \mathcal{D}_i(\mathbf{f}) = 1 \tag{13}$$

Lemma 1. *For any fixed $\mathcal{F}_s$, the optimum domain discriminator $\mathcal{D}^*$ is:*

$$\mathcal{D}_i^*(\mathbf{f}) = \frac{P_i(\mathbf{f})}{\sum_{j=1}^{N} P_j(\mathbf{f})} \tag{14}$$

Proof. For a fixed $\mathcal{F}_s$, the optimum

$$\begin{aligned}
\mathcal{D}^* = \arg\min_{\mathcal{D}} J_{\mathcal{D}} &= \arg\min_{\mathcal{D}} -\sum_{i=1}^{N} \mathop{\mathbb{E}}_{\mathbf{f} \sim P_i} [\log \mathcal{D}_i(\mathbf{f})] \\
&= \arg\max_{\mathcal{D}} \sum_{i=1}^{N} \int_{\mathbf{f}} P_i(\mathbf{f}) \log \mathcal{D}_i(\mathbf{f}) d\mathbf{f} \\
&= \arg\max_{\mathcal{D}} \int_{\mathbf{f}} \sum_{i=1}^{N} P_i(\mathbf{f}) \log \mathcal{D}_i(\mathbf{f}) d\mathbf{f}
\end{aligned}$$

We employ the Lagrangian Multiplier to derive $\arg\max_{\mathcal{D}} \sum_{i=1}^{N} P_i(\mathbf{f}) \log \mathcal{D}_i(\mathbf{f})$ under the constraint of (13). Let

$$L(\mathcal{D}_1, \ldots, \mathcal{D}_N, \lambda) = \sum_{i=1}^{N} P_i \log \mathcal{D}_i - \lambda(\sum_{i=1}^{N} \mathcal{D}_i - 1)$$

Let $\nabla L = 0$:

$$\begin{cases} \nabla_{\mathcal{D}_i} \sum_{j=1}^{N} P_j \log \mathcal{D}_j = \lambda \nabla_{\mathcal{D}_i} (\sum_{j=1}^{N} \mathcal{D}_j - 1) \quad (\forall i) \\ \sum_{i=1}^{N} \mathcal{D}_i - 1 = 0 \end{cases}$$

Solving the two equations, we have:

$$\mathcal{D}_i^*(\mathbf{f}) = \frac{P_i(\mathbf{f})}{\sum_{j=1}^{N} P_j(\mathbf{f})}$$

□

On the other hand, the loss function of the shared feature extractor $\mathcal{F}_s$ consists of two additive components, the loss from the text classifier $\mathcal{C}$, and the loss from the domain discriminator $\mathcal{D}$:

$$J_{\mathcal{F}_s} = J_{\mathcal{F}_s}^{\mathcal{C}} + \lambda J_{\mathcal{F}_s}^{\mathcal{D}} \triangleq J_{\mathcal{C}} - \lambda J_{\mathcal{D}} \tag{15}$$

We have the following theorem for the domain loss for $\mathcal{F}_s$:

Theorem 1. *When $\mathcal{D}$ is trained to its optimality:*

$$J_{\mathcal{F}_s}^{D} = -J_{\mathcal{D}^*} = -N \log N + N \cdot JSD(P_1, P_2, \ldots, P_N) \tag{16}$$

where $JSD(\cdot)$ is the generalized Jensen-Shannon Divergence (Lin, 1991) among multiple distributions.

Proof. Let $\overline{P} = \frac{\sum_{i=1}^{N} P_i}{N}$.

There are two equivalent definitions of the generalized Jensen-Shannon divergence: the original definition based on Shannon entropy (Lin, 1991), and a reshaped one expressed as the average Kullback-Leibler divergence of each P_i to the centroid $\overline{P}$ (Aslam and Pavlu, 2007). We adopt the latter one here:

$$JSD(P_1, P_2, \ldots, P_N) \triangleq \frac{1}{N} \sum_{i=1}^{N} KL(P_i \| \overline{P}) = \frac{1}{N} \sum_{i=1}^{N} \mathop{\mathbb{E}}_{\mathbf{f} \sim P_i} \left[\log \frac{P_i(\mathbf{f})}{\overline{P}(\mathbf{f})} \right] \tag{17}$$

Now substituting $\mathcal{D}^*$ into $J_{\mathcal{F}_s}^{\mathcal{D}}$:

$$\begin{aligned}
J_{\mathcal{F}_s}^{\mathcal{D}} = -J_{\mathcal{D}^*} &= \sum_{i=1}^{N} \mathop{\mathbb{E}}_{\mathbf{f} \sim P_i} \left[\log \mathcal{D}_i^*(\mathbf{f}) \right] \\
&= \sum_{i=1}^{N} \mathop{\mathbb{E}}_{\mathbf{f} \sim P_i} \left[\log \frac{P_i(\mathbf{f})}{\sum_{j=1}^{N} P_j(\mathbf{f})} \right] \\
&= -N \log N + \sum_{i=1}^{N} \mathop{\mathbb{E}}_{\mathbf{f} \sim P_i} \left[\log \frac{P_i(\mathbf{f})}{\sum_{j=1}^{N} P_j(\mathbf{f})} + \log N \right] \\
&= -N \log N + \sum_{i=1}^{N} \mathop{\mathbb{E}}_{\mathbf{f} \sim P_i} \left[\log \frac{P_i(\mathbf{f})}{\frac{\sum_{j=1}^{N} P_j(\mathbf{f})}{N}} \right] \\
&= -N \log N + \sum_{i=1}^{N} \mathop{\mathbb{E}}_{\mathbf{f} \sim P_i} \left[\log \frac{P_i(\mathbf{f})}{\overline{P}} \right] \\
&= -N \log N + \sum_{i=1}^{N} KL(P_i \| \overline{P}) \\
&= -N \log N + N \cdot JSD(P_1, P_2, \ldots, P_N)
\end{aligned}$$

□

Consequently, by the non-negativity of JSD (Lin, 1991), we have the following corollary:

Corollary 1. *The optimum of $J_{\mathcal{F}_s}^{\mathcal{D}}$ is $-N \log N$, and is achieved if and only if $P_1 = P_2 = \cdots = P_N = \overline{P}$.*

A.2 Proofs for MAN-L2

The proof is similar for MAN with the L2 loss. The loss function used by $\mathcal{D}$ is, for a sample from domain d_i with shared feature vector $\mathbf{f}$:

$$\mathcal{L}_{\mathcal{D}}(\mathcal{D}(\mathbf{f}), i) = \sum_{j=1}^{N} (\mathcal{D}_j(\mathbf{f}) - \mathbb{1}_{\{i=j\}})^2 \tag{18}$$

So the objective that $\mathcal{D}$ minimizes is:

$$J_{\mathcal{D}} = \sum_{i=1}^{N} \mathop{\mathbb{E}}_{\mathbf{f} \sim P_i} \left[\sum_{j=1}^{N} (\mathcal{D}_j(\mathbf{f}) - \mathbb{1}_{\{i=j\}})^2 \right] \tag{19}$$

For simplicity, we further constrain $\mathcal{D}$'s outputs to be on a simplex:

$$\sum_{i=1}^{N} \mathcal{D}_i(\mathbf{f}) = 1 \quad (\forall \mathbf{f}) \tag{20}$$

Lemma 2. *For any fixed $\mathcal{F}_s$, the optimum domain discriminator $\mathcal{D}^*$ is:*

$$\mathcal{D}_i^*(\mathbf{f}) = \frac{P_i(\mathbf{f})}{\sum_{j=1}^{N} P_j(\mathbf{f})} \tag{21}$$

Proof. For a fixed $\mathcal{F}_s$, the optimum

$$\begin{aligned}
\mathcal{D}^* = \arg\min_{\mathcal{D}} J_{\mathcal{D}} &= \arg\min_{\mathcal{D}} \sum_{i=1}^{N} \mathop{\mathbb{E}}_{\mathbf{f}\sim P_i} \left[\mathcal{L}_{\mathcal{D}}(\mathcal{D}(\mathbf{f}), i)\right] \\
&= \arg\min_{\mathcal{D}} \sum_{i=1}^{N} \int_{\mathbf{f}} P_i(\mathbf{f}) \mathcal{L}_{\mathcal{D}}(\mathcal{D}(\mathbf{f}), i) d\mathbf{f} \\
&= \arg\min_{\mathcal{D}} \int_{\mathbf{f}} \sum_{i=1}^{N} P_i(\mathbf{f}) \sum_{j=1}^{N} (\mathcal{D}_j(\mathbf{f}) - \mathbb{1}_{\{i=j\}})^2 d\mathbf{f}
\end{aligned}$$

Similar to MAN-NLL, we employ the Lagrangian Multiplier to derive $\arg\max_{\mathcal{D}} \sum_{i=1}^{N} P_i(\mathbf{f}) \sum_{j=1}^{N} (\mathcal{D}_j(\mathbf{f}) - \mathbb{1}_{\{i=j\}})^2$ under the constraint of (20). Let $\nabla L = 0$:

$$\begin{cases} 2((\sum_{j=1}^{N} P_j)\mathcal{D}_i - P_i) = \lambda \quad (\forall i) \\ \sum_{i=1}^{N} \mathcal{D}_i - 1 = 0 \end{cases}$$

Solving the two equations, we have $\lambda = 0$ and:

$$\mathcal{D}_i^*(\mathbf{f}) = \frac{P_i(\mathbf{f})}{\sum_{j=1}^{N} P_j(\mathbf{f})}$$

□

For the domain loss of $\mathcal{F}_s$:

Theorem 2. *Let $\overline{P} = \frac{\sum_{i=1}^{N} P_i}{N}$. When $\mathcal{D}$ is trained to its optimality:*

$$\begin{aligned}
J_{\mathcal{F}_s}^{\mathcal{D}} &= \sum_{i=1}^{N} \mathop{\mathbb{E}}_{\mathbf{f}\sim P_i} \left[\sum_{j=1}^{N} (D_j(\mathbf{f}) - \frac{1}{N})^2\right] \\
&= \frac{1}{N} \sum_{i=1}^{N} \chi^2_{Neyman}(P_i \| \overline{P})
\end{aligned} \tag{22}$$

where $\chi^2_{Neyman}(\cdot\|\cdot)$ is the Neyman χ^2 divergence (Nielsen and Nock, 2014).

Proof. Substituting $\mathcal{D}^*$ into $\mathcal{L}^{\mathcal{D}}_{\mathcal{F}_s}$:

$$
\begin{aligned}
J^{\mathcal{D}}_{\mathcal{F}_s} &= \sum_{i=1}^{N} \mathop{\mathbb{E}}_{\mathbf{f}\sim P_i} \left[\sum_{j=1}^{N} (D^*_j(\mathbf{f}) - \frac{1}{N})^2 \right] \\
&= \sum_{i=1}^{N} \int_{\mathbf{f}} P_i \sum_{j=1}^{N} (\frac{P_j}{N\overline{P}} - \frac{1}{N})^2 d\mathbf{f} \\
&= \int_{\mathbf{f}} \sum_{i=1}^{N} \sum_{j=1}^{N} P_i (\frac{P_j}{N\overline{P}} - \frac{1}{N})^2 d\mathbf{f} \\
&= \frac{1}{N^2} \sum_{j=1}^{N} \int_{\mathbf{f}} \sum_{i=1}^{N} P_i (\frac{P_j}{\overline{P}} - 1)^2 d\mathbf{f} \\
&= \frac{1}{N^2} \sum_{j=1}^{N} \int_{\mathbf{f}} N\overline{P} (\frac{P_j}{\overline{P}} - 1)^2 d\mathbf{f} \\
&= \frac{1}{N} \sum_{j=1}^{N} \int_{\mathbf{f}} \frac{(P_j - \overline{P})^2}{\overline{P}} d\mathbf{f} \\
&= \frac{1}{N} \sum_{i=1}^{N} \chi^2_{Neyman}(P_i \| \overline{P})
\end{aligned}
$$

□

Finally, by the joint convexity of f-divergence, we have the following corollary:

Corollary 2.

$$
\begin{aligned}
\mathcal{L}^{\mathcal{D}}_{\mathcal{F}_s} &= \frac{1}{N} \sum_{i=1}^{N} \chi^2_{Neyman}(P_i \| \overline{P}) \\
&\geq \chi^2_{Neyman}(\frac{1}{N} \sum_{i=1}^{N} P_i \| \frac{1}{N} \sum_{i=1}^{N} \overline{P}) \\
&= \chi^2_{Neyman}(\overline{P} \| \overline{P}) = 0
\end{aligned}
$$

and the equality is attained if and only if $P_1 = P_2 = \cdots = P_N = \overline{P}$.

Pivot Based Language Modeling for Improved Neural Domain Adaptation

Yftah Ziser and **Roi Reichart**
Faculty of Industrial Engineering and Management, Technion, IIT
syftah@campus.technion.ac.il, roiri@ie.technion.ac.il

Abstract

Representation learning with pivot-based methods and with Neural Networks (NNs) have lead to significant progress in domain adaptation for Natural Language Processing. However, most previous work that follows these approaches does not explicitly exploit the structure of the input text, and its output is most often a single representation vector for the entire text. In this paper we present the *Pivot Based Language Model (PBLM)*, a representation learning model that marries together pivot-based and NN modeling in a structure aware manner. Particularly, our model processes the information in the text with a sequential NN (LSTM) and its output consists of a context-dependent representation vector for every input word. Unlike most previous representation learning models in domain adaptation, PBLM can naturally feed structure aware text classifiers such as LSTM and CNN. We experiment with the task of cross-domain sentiment classification on 20 domain pairs and show substantial improvements over strong baselines.[1]

1 Introduction

Domain adaptation (*DA*, (Daumé III, 2007; Ben-David et al., 2010)) is a fundamental challenge in NLP, due to the reliance of many algorithms on costly labeled data which is scarce in many domains. To save annotation efforts, DA aims to import algorithms trained with labeled data from one or several domains to new ones. While DA algorithms have long been developed for many tasks and domains (e.g. (Jiang and Zhai, 2007; McClosky et al., 2010; Titov, 2011; Bollegala et al., 2011; Rush et al., 2012; Schnabel and Schütze, 2014)), the unprecedented growth of heterogeneous online content calls for more progress.

DA through Representation Learning (DReL), where the DA method induces shared representations for the examples in the source and the target domains, has become prominent in the Neural Network (NN) era. A seminal (non-NN) DReL work is structural correspondence learning (SCL) (Blitzer et al., 2006, 2007) which models the connections between pivot features – features that are frequent in the source and the target domains and are highly correlated with the task label in the source domain – and the other, non-pivot, features. While this approach explicitly models the correspondence between the source and the target domains, it has been outperformed by NN-based models, particularly those based on autoencoders (AEs, (Glorot et al., 2011; Chen et al., 2012)) which employ compress-based noise reduction to extract features that empirically support domain adaptation. Recently, Ziser and Reichart (2017) (ZR17) proposed to marry these approaches. They have presented the autoencoder-SCL models and demonstrated their superiority over a large number of previous approaches, particularly those that employ pivot-based ideas only or NNs only.

Current DReL methods, however, suffer from a fundamental limitation: they ignore the structure of their input text (usually sentence or document). This is reflected both in the way they represent their input text, typically with a single vector whose coordinates correspond to word counts or indicators across the text, and in their output which typically consists of a single vector representation. This structure-indifferent approach stands in a sharp contrast to numerous NLP algorithms where text structure plays a key role.

Moreover, learning a single feature vector per

[1] Our code is publicly available at: https://github.com/yftah89/PBLM-Domain-Adaptation.

Proceedings of NAACL-HLT 2018, pages 1241–1251
New Orleans, Louisiana, June 1 - 6, 2018.

input example, these methods can feed only task classifiers such as SVM and feed-forward NNs that take a single vector as input, but cannot feed sequential (e.g. RNNs and LSTMs (Hochreiter and Schmidhuber, 1997)) or convolution (CNNs (LeCun et al., 1998)) networks that require an input vector per word or sentence in their input. This may be a serious limitation given the excellent performance of structure aware models in a large variety of NLP tasks, including sentiment analysis and text classification (e.g.(Kim, 2014; Yogatama et al., 2017)) - prominent DA evaluation tasks.

Fig. 1 demonstrates the limitation of structure-indifferent modeling in DA for sentiment analysis. While the example review contains more positive pivot features (see definition in Sec. 2), the sentiment expressed in the review is negative. A representation learning method should encode the review structure (e.g. the role of the terms *at first* and *However*) in order to uncover the sentiment.[2]

In this paper we overcome these limitations. We present (Section 3) the *Pivot Based Language Model (PBLM)* - a domain adaptation model that (a) is aware of the structure of its input text; and (b) outputs a representation vector for every input word. Particularly, the model is a sequential NN (LSTM) that operates very similarly to LSTM language models (LSTM-LMs). The fundamental difference is that while for every input word LSTM-LMs output a hidden vector and a prediction of the next word, the output of PBLM is a hidden vector and a prediction of the next word if that word is a pivot feature or else, a generic NONE tag. Hence, PBLM not only exploits the sequential nature of its input text, but its output states can naturally feed LSTM and CNN task classifiers. Notice that PBLM is very flexible: instead of pivot based unigram prediction it can be defined to predict pivots of arbitrary length (e.g. the next bigram or trigram), or, alternatively, it can be defined over sentences or other textual units instead of words.

Following a large body of DA work, we experiment (Section 5) with the task of binary sentiment classification. We consider adaptation between each domain pair in the four product review domains of Blitzer et al. (2007) (12 domain pairs) as well as between these domains and an airline review domain (Nguyen, 2015) and vice versa (8 domain pairs). The latter 8 setups are particularly challenging as the airline reviews tend to be more negative than the product reviews (see Section 4).

[2]Pivots are defined with respect to a (source, target) domain pair. The pivots highlighted in the figure are the pivots for this review in all the setups we explored.

> I was *at first* **very excited** with my new Zyliss salad spinner - it is easy to spin and looks **great** ... . *However*, ... it doesn't get your greens very dry. I've been surprised and **disappointed** by the amount of water left on lettuce after spinning, and spinning, and spinning.

Figure 1: Example review from the kitchen appliances domain of Blitzer et al. (2007). Positive pivot features are underlined with a wavy line. Negative pivot features are underlined with a straight line. Although there are more positive pivots than negative ones, the review is negative.

We implement PBLM with two task classifiers, LSTM and CNN, and compare them to strong previous models, among which are: SCL (pivot based, no NN), the marginalized stacked denoising autoencoder model (MSDA, (Chen et al., 2012) - AE based, no pivots), the MSDA-DAN model ((Ganin et al., 2016) - AE with a Domain Adversarial Network (DAN) enhancement) and AE-SCL-SR (the best performing model of ZR17, combining AEs, pivot information and pre-trained word vectors). PBLM-LSTM and PBLM-CNN perform very similarly to each other and strongly outperform previous models. For example, PBLM-CNN achieves averaged accuracies of 80.4%, 84% and 76.2% in the 12 product domain setups, 4 product to airline setups and 4 airline to product setups, respectively, while AE-SCL-SR, the best baseline, achieves averaged accuracies of 78.1%, 78.7% and 68.1%, respectively.

2 Background and Previous Work

DA is an established challenge in machine learning in general and in NLP in particular (e.g. (Roark and Bacchiani, 2003; Chelba and Acero, 2004; Daumé III and Marcu, 2006)). While DA has several setups, the focus of this work is on unsupervised DA. In this setup we have access to unlabeled data from the the source and the target domains, but labeled data is available in the source domain only. We believe that in the current web era with the abundance of text from numerous domains, this is the most realistic setup.

Several approaches to DA have been proposed, for example: instance reweighting (Huang et al., 2007; Mansour et al., 2009), sub-sampling from

both domains (Chen et al., 2011) and learning joint target and source feature representations (DReL), the approach we take here. The rest of this section hence discusses DReL work that is relevant to our ideas, but first we describe our problem setup.

Unsupervised Domain Adaptation with DReL The pipeline of this setup typically consists of two steps: representation learning and classification. In the first step, a representation model is trained on the unlabeled data from the source and target domains. In the second step, a classifier for the supervised task is trained on the source domain labeled data. To facilitate domain adaptation, every example that is fed to the task classifier (second step) is first represented by the representation model of the first step. This is true both when the task classifier is trained and at test time when it is applied to the target domain.

An exception of this pipeline are end-to-end models that jointly learn to represent the data and to perform the classification task, exploiting the unlabeled and labeled data together. A representative member of this class of models (MSDA-DAN, (Ganin et al., 2016)) is one of our baselines.

Pivot Based Domain Adaptation This approach was proposed by Blitzer et al. (2006, 2007), through their SCL method. Its main idea is to divide the shared feature space of the source and the target domains to a set of pivot features that are frequent in both domains and have a strong impact on the source domain task classifier, and a complementary set of non-pivot features.

In SCL, after the original feature set is divided into the pivot and non-pivot subsets, this division is utilized in order to map the original feature space of both domains into a shared, low-dimensional, real-valued feature space. To do so, a binary classifier is defined for each of the pivot features. This classifier takes the non-pivot features of an input example as its representation, and is trained on the unlabeled data from both the source and the target domains, to predict whether its associated pivot feature appears in the example or not. Note that no human annotation is required for the training of these classifiers, the supervision signal is in the unlabeled data. The matrix whose columns are the weight vectors of the classifiers is post-processed with singular value decomposition (SVD) and the derived matrix maps feature vectors from the original space to the new.

Since the presentation of SCL, pivot-based DA has been researched extensively (e.g. (Pan et al., 2010; Gouws et al., 2012; Bollegala et al., 2015; Yu and Jiang, 2016; Ziser and Reichart, 2017)). PBLM is a pivot-based method but, in contrast to previous models, it relies on sequential NNs to exploit the structure of the input text. Even models such as (Bollegala et al., 2015), that embed pivots and non-pivots so that the former can predict if the latter appear in their neighborhood, learn a single representation for all the occurrences of a word in the input corpus. That is, Bollegala et al. (2015), as well as other methods that learn cross-domain word embeddings (Yang et al., 2017), learn word-type representations, rather than context specific representations. In Sec. 3 we show how PBLM's context specific outputs naturally feed structure aware task classifiers such as LSTM and CNN.

AE Based Domain Adaptation The basic elements of an autoencoder are an encoder function e and a decoder function d, and its output is a reconstruction of its input x: $r(x) = d(e(x))$. The parameters of the model are trained to minimize a loss between x and $r(x)$, such as their Kullback-Leibler (KL) divergence or their cross entropy.

Variants of AEs are prominent in recent DA literature. Examples include Stacked Denoising Autoencoders (SDA, (Vincent et al., 2008; Glorot et al., 2011) and marginalized SDA (MSDA, (Chen et al., 2012)) that is more computationally efficient and scalable to high-dimensional feature spaces than SDA, and has been extended in various manners (e.g. (Yang and Eisenstein, 2014; Clinchant et al., 2016)). Finally, models based on variational autoencoders (Kingma and Welling, 2014; Rezende et al., 2014) have recently been applied in DA (e.g. variational fair autoencoder (Louizos et al., 2016)), but in our experiments they were still not competitive with MSDA.

While AE based models have set a new state-of-the-art for DA in NLP, they are mostly based on noise reduction in the representation and do not exploit task specific and linguistic information. This paved the way for ZR17 that integrated pivot-based ideas into domain adaptation with AEs.

Combining Pivots and AEs in Domain Adaptation ZR17 combined AEs and pivot-based modeling for DA. Their basic model (AE-SCL) is a three layer feed-forward network where the non-pivot features are fed to the input layer, encoded

into a hidden representation and this hidden representation is then decoded into the pivot features of the input example. Their advanced model (AE-SCL-SR) has the same architecture but the decoding matrix consists of pre-trained embeddings of the pivot features, which encourages input documents with similar pivots to have similar hidden representations. These embeddings are induced by word2vec (Mikolov et al., 2013) trained with unlabeled data from the source and the target domains.

ZR17 have demonstrated the superiority of their models (especially, AE-SCL-SR) over SCL (pivot-based, no AE), MSDA (AE-based, no pivots) and MSDA-DAN (AE-based with adversarial enhancement, no pivots) in 16 cross-domain sentiment classification setups, including the 12 legacy setups of Blitzer et al. (2007). However, as in previous pivot based methods, AE-SCL and AE-SCL-SR learn a single, structure-indifferent, feature representation of the input text. Our core idea is to implement a pivot-based sequential neural model that exploits the structure of its input text and that its output representations can be smoothly integrated with structure aware classifiers such as LSTM and CNN. Our second goal is motivated by the strong performance of LSTM and CNN in text classification tasks (Yogatama et al., 2017).

3 Domain Adaptation with PBLMs

We now introduce our PBLM model that learns representations for DA. As PBLM is inspired by language modeling, we assume the original feature set of the NLP task classifier consists of word unigrams and bigrams. This choice of features also allows us to directly compare our work to the rich literature on DA for sentiment classification where this is the standard feature set. PBLM, however, is not limited to word n-gram features.

We start with a brief description of LSTM based language modeling (LSTM-LM, (Mikolov et al., 2010)) and then describe how PBLM modifies that model in order to learn pivot-based representations that are aware of the structure of the input text. We then show how to employ these representations in structure aware text classification (with LSTM or CNN) and how to train such PBLM-LSTM and PBLM-CNN classification pipelines.

LSTM Language Modeling LSTMs address the vanishing gradient problem commonly found in RNNs (Elman, 1990) by incorporating gating functions into their state dynamics (Hochreiter and Schmidhuber, 1997). At each time step, an LSTM maintains a hidden vector, h_t, computed in a sequence of non-linear transformations of the input x_t and the previous hidden states $h_1, \ldots, h_{t-1}$.

Given an input word, an LSTM-LM should predict the next word in the sequence. For a lexicon V, the probability of the j-th word is:

$$p(y_t = j) = \frac{e^{h_t \cdot W_j}}{\sum_{k=1}^{|V|} e^{h_t \cdot W_k}}$$

Here, W_i is a parameter vector learned by the network for each of the words in the vocabulary. The loss function we consider in this paper is the cross-entropy loss over these probabilities.

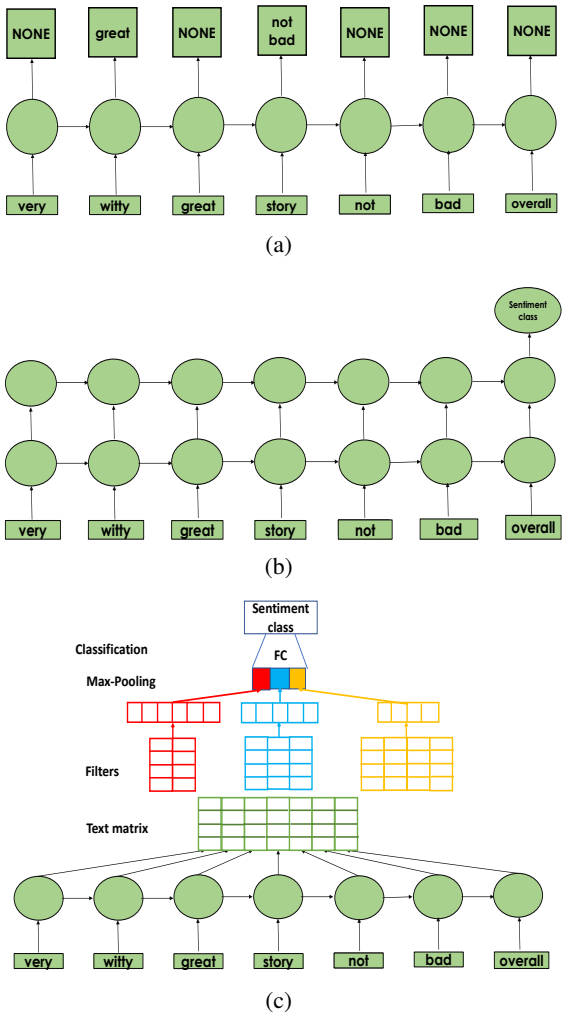


Figure 2: (a) Second order PBLM for representation learning. (b+c) PBLM based models for DA: PBLM-LSTM (b) and PBLM-CNN (c).

Representation Learning with PBLM Figure 2a provides an illustration of the PBLM model. The first (bottom) layer is an embedding layer,

where a 1-hot word vector input is multiplied by a (randomly initialized) parameter matrix before being passed to the next layer. The second layer is an LSTM that predicts the next bigram or unigram if one of these is a pivot (if both are, it predicts the bigram). Otherwise its prediction is NONE.

PBLM operates similarly to LSTM-LM. The basic difference between the models is the prediction they make for a given input word (x_t). While an LSTM-LM aims to predict the next input word, PBLM predicts the next word unigram or bigram if one of these is a pivot, and NONE otherwise.

PBLM is very flexible. It can be of any order: a k-order PBLM predicts the longest prefix of the sequence consisting of the next k words, as long as that prefix forms a pivot. If none of the prefixes forms a pivot then PBLM predicts NONE.[3] Moreover, while PBLM is defined here over word sequences, it can be defined over other sequences, e.g., the sentence sequence of a document.

Intuitively, in the example of fig. 2a a second order model is more informative for sentiment classification than a first-order model (that predicts only the next word unigram in case that word is a pivot) would be. Indeed, "not bad" conveys the relevant sentiment-related information, while "bad" is misleading with respect to that same sentiment. Notice that after the prefix "very witty" the model predicts "great" and not "great story" because in this example "great" is a pivot while "great story" is not, as "great story" is unlikely to be frequent outside the book review domain.

Figures 2a and 1 also demonstrate a major advantage of PBLM over models that learn a single text representation. From the book review example in fig. 2a, PBLM learns the connection between *witty* - an adjective that is often used to describe books, but not kitchen appliances - and *great* - a common positive adjective in both domains, and hence a pivot feature. Likewise, from the example of fig. 1 PBLM learns the connection between *easy* - an adjective that is often used to describe kitchen appliances, but not books - and *great*. That is, PBLM is able to learn the connection between *witty* and *easy* which will facilitate adaptation between the books and kitchen appliances domains. Previous work that learns a single text representation, in contrast, would learn from fig. 1 a connection between *easy* and the three pivots: *very excited*, *great* and *disappointed*. From fig. 2a such a method would learn the connection between *witty* and *great* and *not bad*. The connection between *witty* and *easy* will be much weaker.

Structure Aware Classification with PBLM Representations PBLM not only exploits the sequential nature of its input text, but its output vectors can feed LSTM (PBLM-LSTM, fig. 2b) and CNN (PBLM-CNN, fig. 2c) classifiers.

PBLM-LSTM is a three-layer model. The bottom two layers are the PBLM model of fig. 2a. When PBLM is combined with a classifier, its softmax layer (top layer of fig. 2a) is cut and only its output vectors (h_t) are passed to the next LSTM layer (third layer of fig. 2b). The final hidden vector of that layer feeds the task classifier.

Note that since we cut the PBLM softmax layer when it is combined with the task classifier, PBLM should be trained before this combination is performed. Below we describe how we exploit this modularity to facilitate domain adaptation.

In PBLM-CNN, the combination between the PBLM and the CNN is similar to fig. 2b: the PBLM's softmax layer is cut and a matrix whose columns are the h_t vectors of the PBLM is passed to the CNN. We employ K different filters of size $|h_t| \times d$, each going over the input matrix in a sliding window of d consecutive hidden vectors, and generating a $1 \times (n-d+1)$ size vector, where n is the input text length. A max pooling is performed for each of the k vectors to generate a single $1 \times K$ vector that is fed into the task classifier.

PBLM can feed structure aware classifiers other than LSTM and CNN. Moreover, PBLM can also generate a single text representation as in most previous work. This can be done, e.g., by averaging the PBLM's hidden vectors and feeding the averaged vector into a linear non-structured classifier (e.g. logistic regression) or a feed-forward NN. In Sec. 5 we demonstrate that PBLM's ability to feed structure aware classifiers such as LSTM and CNN provides substantial accuracy gains. To the best of our knowledge, PBLM is unique in its structure aware representation: previous work generated one representation per input example.

Domain Adaptation with PBLM Representations We focus on unsupervised DA where the input consists of a source domain labeled set and a plentiful of unlabeled examples form the source and the target domains. Our goal is to use the unlabeled data as a bridge between the domains.

[3] A word sequence is one of its own prefixes.

Our fundamental idea is to decouple the PBLM training which requires only unlabeled text, from the NLP classification task which is supervised and for which the required labeled example set is available only for the source domain. We hence employ a two step training procedure. First PBLM (figure 2a) is trained with unlabeled data from both the source and the target domains. Then the trained PBLM is combined with the classifier layers (top layer of fig. 2b, CNN layers of fig. 2c) and the final model is trained with the source domain labeled data to perform the classification task. As noted above, in the second step we cut the PBLM's softmax layer, only its h_t vectors are passed to the classifier. Moreover, during this step the parameters of the pre-trained PBLM are held fixed, only the parameters of the classifier layers are trained.

4 Experimental Setup

[4]**Task and Domains** Following a large body of DA work, we experiment with the task of cross-domain sentiment classification. To facilitate comparison with previous work we experiment with the product review domains of (Blitzer et al., 2007) – Books (B), DVDs (D), Electronic items (E) and Kitchen appliances (K) (12 ordered domain pairs) – replicating the experimental setup of ZR17 (including baselines, design, and hyper-parameter details). For each domain there are 2000 labeled reviews, 1000 positive and 1000 negative, and unlabeled reviews: 6000 (B), 34741 (D), 13153 (E) and 16785 (K).

To consider a more challenging setup we experiment with a domain consisting of user reviews on services rather than products. We downloaded an airline review dataset, consisting of reviews labeled by their authors (Nguyen, 2015). We randomly sampled 1000 positive and 1000 negative reviews for our labeled set, the remaining 39396 reviews form our unlabeled set. We hence have 4 product to airline and 4 airline to product setups.

Interestingly, in the product domains unlabeled reviews tend to be much more positive than in the airline domain. Particularly, in the B domain there are 6.43 positive reviews on every negative review; in D the ratio is 7.39 to 1; in E it is 3.65 to 1; and in K it is 4.61 to 1. In the airline domain there are only 1.15 positive reviews for every negative review. We hence expect DA from product to airline reviews and vice versa to be more challenging than DA from one product review domain to another.[5]

Baselines We consider the following baselines: (a) AE-SCL-SR (ZR17). We also experimented with the more basic AE-SCL but, like in ZR17, we got lower results in most cases; (b) SCL with pivot features selected using the mutual information criterion (SCL-MI, (Blitzer et al., 2007)). For this method we used the implementation of ZR17; (c) MSDA (Chen et al., 2012), with code taken from the authors' web page; (d) The MSDA-DAN model (Ganin et al., 2016) which employs a domain adversarial network (DAN) with the MSDA vectors as input. The DAN code is taken from the authors' repository; (e) The no domain adaptation case where the sentiment classifier is trained in the source domain and applied to the target domain without adaptation. For this case we consider three classifiers: logistic regression (denoted NoSt as it is not aware of its input's structure), as well as LSTM and CNN which provide a control for the importance of the structure aware task classifiers in PBLM models. To further control for this effect we compare to the PBLM-NoSt model where the PBLM output vectors (h_t vectors generated after each input word) are averaged and the averaged vector feeds the logistic regression classifier.[6]

In all the participating methods, the input features consist of word unigrams and bigrams. The division of the feature set into pivots and non-pivots is based on the the method of ZR17 that followed the work of Blitzer et al. (2007) (details are in Appendix C). The sentiment classifier employed with the SCL-MI, MSDA and AE-SCL-SR representations is the same logistic regression classifier as in the NoSt condition mentioned above. For these methods we concatenate the representation learned by the model with the original representation and this representation is fed to the classifier. MSDA-DAN jointly learns the feature representation and performs the sentiment classification task. It is hence fed by a concatenation of the original and the MSDA-induced representations.

[4]The URLs of the datasets and the code (previous models and standard packages) we used, are in Appendix A.

[5]While we have the labels for our unlabeled data, we did not use them in our research except in this analysis.

[6]We considered several additional baselines: (1) Variational fair autoencoder (Louizos et al., 2016) which performed substantially worse than the DA baselines ((a)-(d)); (2) We tried to compare to (Bollegala et al., 2015) but, similarly to ZR17, failed to replicate their results; and (3) We replaced PBLM with an LSTM-LM, but the results substantially degraded. We do not report results for these models.

Five Fold CV We employ a 5-fold cross-validation protocol as in (Blitzer et al., 2007; Ziser and Reichart, 2017). In all five folds 1600 source domain examples are randomly selected for training data and 400 for development, such that both the training and the development sets are balanced and have the same number of positive and negative reviews. The results we report are the averaged performance of each model across these 5 folds.

Hyperparameter Tuning For all previous models, we follow the tuning process described in ZR17 (paper and appendices). Hyperparameter tuning for the PBLM models and the non-adapted CNN and LSTM is described in Appendix B.

5 Results

Overall Results Table 1 presents our results. PBLM models with structure aware classifiers (PBLM-LSTM and PBLM-CNN, henceforth denoted together as S-PBLM) outperform all other alternatives in all 20 setups and three averaged evaluations (*All* columns in the tables). The gaps are quite substantial – the average accuracy of PBLM-LSTM and PBLM-CNN compared to the best baseline, AE-SCL-SR, are: 79.6% and 80.4% vs. 78.1% for the product review setups, 85% and 84% vs. 78.7% for the product to airline (service) review setups, and 76.1% and 76.2% vs. 68.1% for the airline to product review setups.

S-PBLM performance in the more challenging product to airline and airline to product setups are particularly impressive. The challenging nature of these setups stems from the presumably larger differences between product and service reviews and from the different distribution of positive and negative reviews in the unlabeled data of both domains (Sec. 4). These differences are reflected by the lower performance of the non-adapted classifiers: an averaged accuracy of 70.6%-73.1% across product domain pairs (three lower lines of the *All* column of the top table), compared to an average of 67.3%-69.9% across product to airline setups and an average of 61.3%-62.4% across airline to product setups. Moreover, while the best previous method (AE-SCL-SR) achieves an averaged accuracy of 78.1% for product domains and an averaged accuracy of 78.7% when adapting from product to airline reviews, when adapting from airline to product reviews its averaged accuracy drops to 68.1%. The S-PBLM models do consistently better in all three setups, with an averaged accuracy of 80.4%, 85% and 76.2% of the best S-PBLM model, respectively.

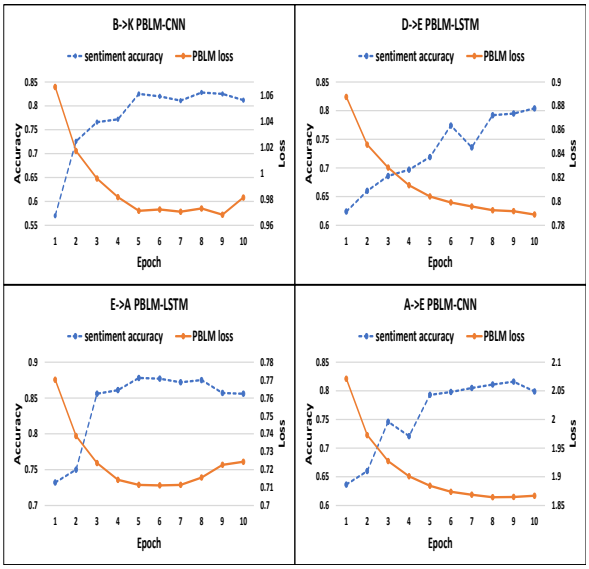


Figure 3: PBLM loss (solid, red line) vs. sentiment accuracy (dashed, blue line) of PBLM-CNN (top) and PBLM-LSTM (bottom) in four representative setups. Patterns in other setups are very similar.

Analysis of S-PBLM Strength The results shed light on the sources of the S-PBLM models success. The accuracy of these models, PBLM-LSTM and PBLM-CNN, is quite similar across setups: their accuracy gap is up to 3.1% in all 20 setups and up to 1% in the three averages (*All* columns). However, the S-PBLM models substantially outperform PBLM-NoSt that employs a structure-indifferent classifier. The averaged gaps are 5.6% (80.4% vs. 74.8%) in the product to product setups, 11.1% in the product to airline setups (85% vs. 73.9%) and 10.9% in the airline to product setups (76.2% vs. 65.3%). Hence, we can safely conclude that while the integration of PBLM with a structured task classifier has a dramatic impact on cross-domain accuracy, it is less important if that classifier is an LSTM or a CNN.

Comparison with non-adapted models reveals that structure aware modeling, as provided by LSTM and CNN, is not sufficient for high performance. Indeed, non-adapted LSTM and CNN do substantially worse than S-PBLM in all setups. Finally, comparison with AE-SCL-SR demonstrates that while the integration of pivot based learning with NNs leads to stronger results than in any other previous work, the structure awareness of the S-PBLM models substantially improves accuracy.

Product Review Domains (Blitzer et al., 2007)													
Source-Target	**D-B**	**E-B**	**K-B**	**B-D**	**E-D**	**K-D**	**B-E**	**D-E**	**K-E**	**B-K**	**D-K**	**E-K**	**All**
PBLM Models													
PBLM-LSTM	80.5	70.8	73.5	82.6	**77.6**	78.6	74.5	**80.4**	85.4	80.9	**83.3**	87.1	79.6
PBLM-CNN	**82.5**	**71.4**	**74.2**	**84.2**	75	**79.8**	**77.6**	79.6	**87.1**	**82.5**	83.2	**87.8**	**80.4**
PBLM-NoSt	74	68.6	67.4	78.3	73.2	73.3	71.3	74.2	82.1	75.5	76.9	83.2	74.8
Previous Work Models													
AE-SCL-SR	77.3	71.1	73	81.1	74.5	76.3	76.8	78.1	84	80.1	80.3	84.6	78.1
MSDA	76.1	71.9	70	78.3	71	71.4	74.6	75	82.4	78.8	77.4	84.5	75.9
MSDA-DAN	75	71	71.2	79.7	73.1	73.8	74.7	74.5	82.1	75.4	77.6	85	76.1
SCL-MI	73.2	68.5	69.3	78.8	70.4	72.2	71.9	71.5	82.2	77.2	74	82.9	74.3
No Domain Adaptation													
NoSt	73.6	67.9	67.6	76	69.1	70.2	70	70.9	81.6	74	73.2	82.4	73.1
LSTM	69.2	67.9	67.5	72.8	68.1	66.2	65.9	68.3	78.2	72.1	70.5	80.6	70.6
CNN	71.2	65.6	66.5	73.6	67.1	70.8	69.6	69.7	79.9	72.7	72.6	80.6	71.6

Product and Airline Review Domains (Blitzer et al., 2007; Nguyen, 2015)										
Source-Target	B-A	D-A	E-A	K-A	All (P-Air)	A-B	A-D	A-E	A-K	All (Air-P)
PBLM Models										
PBLM-LSTM	83.7	**81**	**87.7**	**87.4**	**85**	70.3	71.1	80.5	**82.6**	76.1
PBLM-CNN	**83.8**	78.3	86.5	86.1	84	**70.6**	**71.3**	**81.1**	81.8	**76.2**
PBLM-NoSt	74.2	74.9	72.4	73.9	73.9	62.5	62	69.6	67.3	65.3
Previous Work Models										
AE-SCL-SR	79.1	76.1	82.6	76.9	78.7	60.5	66	74.4	71.7	68.1
MSDA	72.2	73.3	75.1	76.8	74.3	58.5	61	70.6	69	64.8
MSDA-DAN	73.5	73.9	76.3	76.6	75	59.5	60.7	71	71.7	65.7
SCL	70.9	69	80.2	72.3	73	61.7	62.1	72.3	69.7	66.4
No Domain Adaptation										
NoSt	68.5	67.6	74	69.6	69.9	57.5	59.7	67.2	65.2	62.4
LSTM	68.3	65	72.1	68.6	67.3	56.7	57.3	66.2	65	61.3
CNN	67.6	66.7	72	70	69.1	56.3	59	66	66.6	62

Table 1: Accuracy of adaption between product review domains (top table). and between product review domains and the airline (A) review domain (bottom table). All the differences between PBLM-CNN and AE-SCL-SR and between PBLM-LSTM and AE-SCL-SR are statistically significant (except from E-B in the former comparison and E-B and K-B in the latter). Statistical significance is computed with the McNemar paired test for labeling disagreements ((Gillick and Cox, 1989; Blitzer et al., 2006), $p < 0.05$).

Figure 3 further demonstrates the adequacy of the PBLM architecture for domain adaptation. The graphs demonstrate, for both S-PBLM models, a strong correlation between the PBLM cross-entropy loss values and the sentiment accuracy of the resulting PBLM-LSTM and PBLM-CNN models. We show these patterns for two product domain setups and two setups that involve a product domain and the airline domain – the patterns for the other setups of table 1 are very similar.

This analysis highlights our major contribution. We have demonstrated that it is the combination of four components that makes DA for sentiment classification very effective: (a) Neural network modeling; (b) Pivot based modeling; (c) Structure awareness of the pivot-based model; and (d) Structure awareness of the task classifier.

6 Conclusions

We addressed the task of DA in NLP and presented PBLM: a representation learning model that combines pivot-based ideas and NN modeling, in a structure aware manner. Unlike previous work, PBLM exploits the structure of its input, and its output consists of a vector per input word. PBLM-LSTM and PBLM-CNN substantially outperform strong previous models in traditional and newly presented sentiment classification DA setups.

In future we intend to extend PBLM so that it could deal with NLP tasks that require the prediction of a linguistic structure. For example, we believe that PBLM can be smoothly integrated with recent LSTM-based parsers (e.g. (Dyer et al., 2015; Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017)). We also intend to extend the reach of our approach to cross-lingual setups.

References

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Machine learning* 79(1-2):151–175. https://doi.org/10.1007/s10994-009-5152-4.

John Blitzer, Mark Dredze, Fernando Pereira, et al. 2007. Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification. In *Proc. of ACL*. http://aclweb.org/anthology/P07-1056.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proc. of EMNLP*. http://aclweb.org/anthology/W06-1615.

Danushka Bollegala, Takanori Maehara, and Ken-ichi Kawarabayashi. 2015. Unsupervised cross-domain word representation learning. In *Proc. of ACL*. https://doi.org/10.3115/v1/P15-1071.

Danushka Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka. 2011. Relation adaptation: learning to extract novel relations with minimum supervision. In *Proc. of IJCAI*. https://doi.org/10.1109/TKDE.2011.250.

Ciprian Chelba and Alex Acero. 2004. Adaptation of maximum entropy capitalizer: Little data can help a lot. In *Proc. of EMNLP*. http://aclweb.org/anthology/W04-3237.

Minmin Chen, Yixin Chen, and Kilian Q Weinberger. 2011. Automatic feature decomposition for single view co-training. In *Proc. of ICML*. http://dblp.uni-trier.de/rec/bib/conf/icml/ChenWC11.

Minmin Chen, Zhixiang Xu, Kilian Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. In *Proc. of ICML*. http://icml.cc/2012/papers/416.pdf.

Stéphane Clinchant, Gabriela Csurka, and Boris Chidlovskii. 2016. A domain adaptation regularization for denoising autoencoders. In *Proc. of ACL (short papers)*. https://doi.org/10.18653/v1/P16-2005.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proc. of ACL*. http://aclweb.org/anthology/P07-1009.

Hal Daumé III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research* 26:101–126. http://dl.acm.org/citation.cfm?id=1622559.1622562.

Timothy Dozat and Christopher D Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proc. of ICLR*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proc. of ACL*. http://www.aclweb.org/anthology/P15-1033.

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

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research* 17(59):1–35. http://jmlr.org/papers/v17/15-239.html.

Laurence Gillick and Stephen J Cox. 1989. Some statistical issues in the comparison of speech recognition algorithms. In *Proc. of ICASSP*. IEEE. https://doi.org/10.1109/ICASSP.1989.266481.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *In proc. of ICML*. pages 513–520. http://dblp.uni-trier.de/rec/bib/conf/icml/GlorotBB11.

Stephan Gouws, GJ Van Rooyen, MIH Medialab, and Yoshua Bengio. 2012. Learning structural correspondences across different linguistic domains with synchronous neural language models. In *Proc. of the xLite Workshop on Cross-Lingual Technologies, NIPS*.

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

Jiayuan Huang, Arthur Gretton, Karsten M Borgwardt, Bernhard Schölkopf, and Alex J Smola. 2007. Correcting sample selection bias by unlabeled data. In *Proc. of NIPS*.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *Proc. of ACL*. http://aclweb.org/anthology/P07-1034.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *In Proc. of EMNLP*. http://www.aclweb.org/anthology/D14-1181.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proc. of ICLR*.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *Proc. of ICLR*. http://dblp.uni-trier.de/rec/bib/journals/corr/KingmaW13.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the ACL (TACL)* 4:313–327. https://transacl.org/ojs/index.php/tacl/article/view/885.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324. https://doi.org/10.1109/5.726791.

Christos Louizos, Kevin Swersky, Yujia Li, Max Welling, and Richard Zemel. 2016. The variational fair autoencoder http://dblp.uni-trier.de/rec/bib/journals/corr/LouizosSLWZ15.

Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. 2009. Domain adaptation with multiple sources. In *Proc. of NIPS*.

David McClosky, Eugene Charniak, and Mark Johnson. 2010. Automatic domain adaptation for parsing. In *Proc. of NAACL*. http://aclweb.org/anthology/N10-1004.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*. https://doi.org/10.1109/AINL-ISMW-FRUCT.2015.7382966.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proc. of NIPS*.

Quang Nguyen. 2015. The airline review dataset. `https://github.com/quankiquanki/skytrax-reviews-dataset`. Scraped from `www.airlinequality.com`.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th international conference on World wide web*. ACM, pages 751–760. https://doi.org/10.1145/1772690.1772767.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *Proc. of ICML*. http://dblp.uni-trier.de/rec/bib/conf/icml/RezendeMW14.

Brian Roark and Michiel Bacchiani. 2003. Supervised and unsupervised pcfg adaptation to novel domains. In *Proc. of HLT-NAACL*. http://aclweb.org/anthology/N03-1027.

Alexander M Rush, Roi Reichart, Michael Collins, and Amir Globerson. 2012. Improved parsing and pos tagging using inter-sentence consistency constraints. In *Proc. of EMNLP-CoNLL*. http://aclweb.org/anthology/D12-1131.

Tobias Schnabel and Hinrich Schütze. 2014. Flors: Fast and simple domain adaptation for part-of-speech tagging. *Transactions of the Association for Computational Linguistics* 2:15–26. http://aclweb.org/anthology/Q/Q14/Q14-1002.pdf.

Ivan Titov. 2011. Domain adaptation by constraining inter-domain variability of latent feature representation. In *Proc. of ACL*. http://www.aclweb.org/anthology/P11-1007.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proc. of ICML*. https://doi.org/10.1145/1390156.1390294.

Wei Yang, Wei Lu, and Vincent Zheng. 2017. A simple regularization-based algorithm for learning cross-domain word embeddings. In *Proc. of EMNLP*. https://www.aclweb.org/anthology/D17-1312.

Yi Yang and Jacob Eisenstein. 2014. Fast easy unsupervised domain adaptation with marginalized structured dropout. In *Proc. of ACL (short papers)*. https://doi.org/10.3115/v1/P14-2088.

Dani Yogatama, Chris Dyer, Wang Ling, and Phil Blunsom. 2017. Generative and discriminative text classification with recurrent neural networks. *arXiv preprint arXiv:1703.01898* .

Jianfei Yu and Jing Jiang. 2016. Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification. In *Proc. of EMNLP*. http://aclweb.org/anthology/D16-1023.

Yftah Ziser and Roi Reichart. 2017. Neural structural correspondence learning for domain adaptation. In *Proc. of CoNLL*. http://aclweb.org/anthology/K17-1040.

A URLs of Code and Data

As mentioned in section 4 of the paper, we provide here a list of URLs for the code and data we use in the paper. We do that in order to avoid a large number of footnotes in the main paper:

- Blitzer et al. (2007) product review data: `http://www.cs.jhu.edu/~mdredze/datasets/sentiment/index2.html`.
- The airline review data is (Nguyen, 2015).
- Code for the AE-SCL and AE-SCL-SR models of ZR17 (Ziser and Reichart, 2017): `https://github.com/yftah89/Neural-SCLDomain-Adaptation`.
- Code for the SCL-MI method of Blitzer et al. (2007): see footnote [7] (the URL does not fit into the line width).

[7] `https://github.com/yftah89/structural-correspondence-learning-SCL`

- Code for MSDA (Chen et al., 2012): http://www.cse.wustl.edu/~mchen.
- Code for the domain adversarial network used as part of the MSDA-DAN baseline (Ganin et al., 2016): https://github.com/GRAAL-Research/domain_adversarial_neural_network.
- Logistic regression code: http://scikit-learn.org/stable/.

B Hyperparameter Tuning and Experimental Details

Hyperparameter Tuning As discussed in section 4 of the paper, for all previous work models, we follow the experimental setup of ZR17 (paper and appendices) including their hyperparameter estimation protocol. The hyperparameters of the PBLM models and the non-adapted CNN and LSTM are provided here. For PBLM we considered the following hyperparameteres:

- Input word embedding size: $(32, 64, 128, 256)$.
- Number of pivot features: $(100, 200, 300, 400, 500)$.
- $|h_t|$: $(128, 256, 512)$.
- PBLM model order: second order.

For the LSTM in PBLM-LSTM as well as the baseline non-adapted LSTM we considered the same $|h_t|$ and input word embedding size values as for PBLM. For PBLM-CNN and for the baseline, non-adapted, CNN we only experimented with $K = 250$ filters and with a kernel of size $d = 3$.

All the algorithms in the paper that involve a CNN or a LSTM (including the PBLM itself) are trained with the ADAM algorithm (Kingma and Ba, 2015). For this algorithm we used the parameters described in the original ADAM article:

- Learning rate: $lr = 0.001$.
- Exponential decay rate for the 1st moment estimates: $\beta_1 = 0.9$.
- Exponential decay rate for the 2nd moment estimates: $\beta_2 = 0.999$.
- Fuzz factor: $\epsilon = 1e - 08$.
- Learning rate decay over each update: $decay = 0.0$.

Experimental Details All sequential models considered in our experiments are fed with one review example at a time. For all models in the paper, punctuation is first removed from the text before it is processed by the model (sentence boundaries are still encoded). This is the only preprecessing step we employ in the paper.

We considered several alternative implementations of the PBLM-NoSt baseline. In the variant we selected the PBLM output vectors (h_t vectors generated after each word of the input review) are averaged and the averaged vector feeds a non-structured logistic regression classifier. We also tried to take only the final h_t vector of PBLM as an input to the classifier or to sum the h_t vectors instead of taking their average. These alternatives gave worse results.

C Pivot Feature Selection

As mentioned in the main paper, the division of the feature set into pivots and non-pivots is based on the unlabeled data from both the source and the target domains, using the method of ZR17 (which is in turn based on (Blitzer et al., 2007)). Here we provide the details of the pivot selection criterion.

Pivot features are frequent in the unlabeled data of both the source and the target domains, appearing at least 10 times in each, and among those features are the ones with the highest mutual information with the task (sentiment) label in the source domain labeled data. For non-pivot features we consider unigrams and bigrams that appear at least 10 times in their domain.

Reinforced Co-Training

Jiawei Wu
Department of Computer Science
University of California
Santa Barbara, CA 93106 USA
jiawei_wu@cs.ucsb.edu

Lei Li
Toutiao AI Lab
Bytedance Co. Ltd
Beijing, 100080 China
lileicc@gmail.com

William Yang Wang
Department of Computer Science
University of California
Santa Barbara, CA 93106 USA
william@cs.ucsb.edu

Abstract

Co-training is a popular semi-supervised learning framework to utilize a large amount of unlabeled data in addition to a small labeled set. Co-training methods exploit predicted labels on the unlabeled data and select samples based on prediction confidence to augment the training. However, the selection of samples in existing co-training methods is based on a predetermined policy, which ignores the sampling bias between the unlabeled and the labeled subsets, and fails to explore the data space. In this paper, we propose a novel method, Reinforced Co-Training, to select high-quality unlabeled samples to better co-train on. More specifically, our approach uses Q-learning to learn a data selection policy with a small labeled dataset, and then exploits this policy to train the co-training classifiers automatically. Experimental results on clickbait detection and generic text classification tasks demonstrate that our proposed method can obtain more accurate text classification results.

1 Introduction

Large labeled datasets are often required to obtain satisfactory performance for natural language processing tasks. However, it is time-consuming to label text corpus manually. In the meanwhile, there are abundant unlabeled text corpora available on the web. Semi-supervised methods permit learning improved supervised models by jointly train on a small labeled dataset and a large unlabeled dataset (Zhu, 2006; Chapelle et al., 2009).

Co-training is one of the widely used semi-supervised methods, where two complementary classifiers utilize large amounts of unlabeled examples to bootstrap the performance of each other iteratively (Blum and Mitchell, 1998; Nigam and Ghani, 2000). Co-training can be readily applied to NLP tasks since data in these tasks naturally have two or more views, such as multi-lingual data (Wan, 2009) and document data (headline and content) (Ghani, 2000; Denis et al., 2003). In the co-training framework, each classifier is trained on one of the two views (aka a subset of features) of both labeled and unlabeled data, under the assumption that either view is sufficient to classify. In each iteration, the co-training algorithm selects high confidence samples scored by each of the classifiers to form an auto-labeled dataset, and the other classifier is then updated with both labeled data and additional auto-labeled set. However, as shown in Figure 1, most of existing co-training methods have some disadvantages. Firstly, the sample selection step ignores distributional bias between the labeled and unlabeled sets. It is common in practice to use unlabeled datasets collected differently from the labeled set, resulting in a significant difference in their sample distribution. After iterative co-training, the sampling

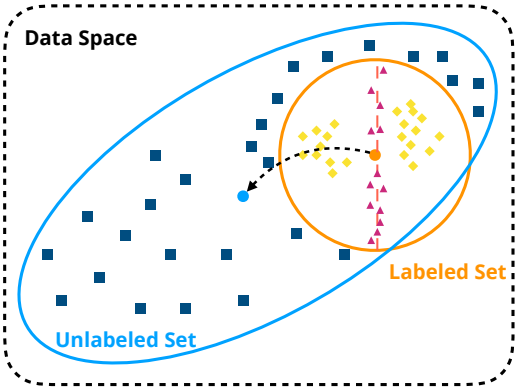


Figure 1: Illustration of sample-selection issues in co-training methods. (1) Randomly sampled unlabeled examples (□) will result in high sampling bias, which will cause bias shift towards the unlabeled dataset (←). (2) High-confidence examples (◇) will contribute little during the model training, especially for discriminating the boundary examples (△), resulting in myopic trained models.

Proceedings of NAACL-HLT 2018, pages 1252–1262
New Orleans, Louisiana, June 1 - 6, 2018.

bias may shift towards the unlabeled set, which results in poor performance of the trained model at the testing time. To remedy such bias, an ideal algorithm should select those samples according to the target (potentially unknown) testing distribution. Secondly, the existing sample selection and training can be myopic. Conventional co-training methods select unlabeled examples with high confidence predicted by trained models. This strategy often causes only those unlabeled examples that match well to the current model being picked during iteration and the model might fail to generalize to complete sample space (Zhang and Rudnicky, 2006). It relates to the well-known exploration-exploitation trade-off in machine learning tasks. An ideal co-training algorithm should explore the space thoroughly to achieve globally better performance. These intuitions inspire our work on learning a data selection policy for the unlabeled dataset in co-training.

The iterate data selection steps in co-training can be viewed as a sequential decision-making problem. To resolve both issues discussed above, we propose **Reinforced Co-Training**, a reinforcement learning (RL)-based framework for co-training. Concretely, we introduce a joint formulation of a Q-learning agent and two co-training classifiers. In contrast to previous predetermined data sampling methods of co-training, we design a Q-agent to automatically learn a data selection policy to select high-quality unlabeled examples. To better guide the policy learning of the Q-agent, we design a state representation to delivery the status of classifiers and utilize the validation set to compute the performance-driven rewards. Empirically, we indicate that our method outperforms previous related methods on clickbait detection and generic text classification problems. In summary, our main contributions are three-fold:

- We are first to propose a joint formulation of RL and co-training methods;
- Our learning algorithm can learn a good data selection policy to select high-quality unlabeled examples for better co-training;
- We show that our method can apply to large-scale document data and outperform baselines in semi-supervised text classification.

In Section 2, we outline related work in semi-supervised learning and co-training. We then describe our proposed method in Section 3. We show experimental results in Section 4. Finally, we conclude in Section 5.

2 Related Work

Semi-supervised learning algorithms have been widely used in NLP (Liang, 2005). As for text classification, Dai and Le (2015) introduce a sequence autoencoder to pre-train the parameters for the later supervised learning process. Johnson and Zhang (2015, 2016) propose a method to learn embeddings of small text regions from unlabeled data for integration into a supervised convolutional neural network (CNN) or long short-term memory network (LSTM). Miyato et al. (2016) further apply perturbations to the word embeddings and pre-train the supervised models through adversarial training. However, these methods mainly focus on learning the local word-level information and pre-trained parameters from unlabeled data, which fails to capture the overall text-level information and potential label information.

Co-training can capture the text-level information of unlabeled data and generate pseudo labels during the training, which is especially useful on unlabeled data with two distinct views (Blum and Mitchell, 1998). However, the confidence-based data selection strategies (Goldman and Zhou, 2000; Zhou and Li, 2005; Zhang and Zhou, 2011) often focus on some special regions of the input space and fail to generate an accurate estimation of data space. Zhang and Rudnicky (2006) proposes a performance-driven data selection strategy based on pseudo-accuracy and energy regularization. Meanwhile, Chawla and Karakoulas (2005) argues that the random data sampling method often causes sampling bias shift of the trained model towards the unlabeled set.

Comparing to previous related methods, our Reinforced Co-Training model can learn a performance-driven data selection policy to select high-quality unlabeled data. Furthermore, the performance estimation is more accurate due to the validation dataset and the data selection strategy is automatically learned instead of human designed. Lastly, the selected high-quality unlabeled data can not only help explore the data space but also reduce the sampling bias shift.

Our work is also related to recent studies in "learning to learn" (Maclaurin et al., 2015; Zoph and Le, 2016; Chen et al., 2017; Wichrowska et al., 2017; Yeung et al., 2017). Learning to learn

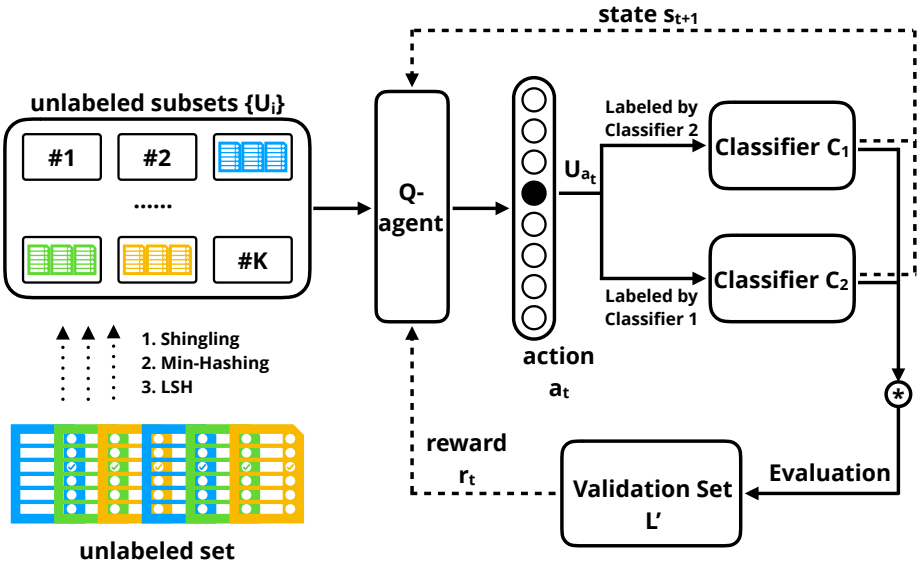


Figure 2: The Reinforced Co-Training framework.

is one of the meta-learning methods (Schmidhuber, 1987; Bengio et al., 1991), where one model is trained to learn how to optimize the parameters of another certain algorithm. While previous studies focus more on neural network optimization (Chen et al., 2017; Wichrowska et al., 2017) and few-shot learning (Vinyals et al., 2016; Ravi and Larochelle, 2016; Finn et al., 2017), we are first to explore how to learn a high-quality data selection policy in semi-supervised methods, in our case, the co-training algorithm.

3 Method

In this section, we describe our RL-based framework for co-training in detail. The conventional co-training methods follow the framework:

1. Initialize two classifiers by training on the labeled set;

2. Iteratively select a subset of unlabeled data based on a predetermined policy;

3. Iteratively update two classifiers with the selected subset of unlabeled data in addition to the labeled one.

Step 2 is the core of different co-training variants. The original co-training algorithm is equipped with a policy of selecting high-confidence samples by two classifiers. Our main idea is to improve the policy by reinforcement learning.

We formulate the data selection process as a sequential decision-making problem and the decision (action) a_t at each iteration (time step) t is to select a portion of unlabeled examples. This problem can be solved with an RL-agent by learning a policy. We first describe how we organize the large unlabeled dataset to improve the computational efficiency. Then we briefly introduce the classifier models used in co-training. After that, we describe the Q-agent, the RL-agent used in our framework and the environment in RL. The two co-training classifiers are integrated into the environment and the Q-agent can learn a good data selection policy by interacting with the environment. Finally, we describe how to train the Q-agent in our unified framework.

3.1 Partition Unlabeled Data

Considering that the number of unlabeled samples is enormous, it is not efficient for the RL-agent to select only one example at each time step t. Thus, first we want to partition documents from the unlabeled dataset into different subsets based on their similarity. At each time step t, the RL-agent applies a policy to select one subset instead of one sample and then update the two co-training classifiers, which can significantly improve the computational efficiency.

Suppose each example in the unlabeled dataset as document D, where D is the concatenation of the headline and paragraph. V is the vocabulary of

these documents. These documents are partitioned into different subsets based on Jaccard similarity, which is defined as:

$$sim(D_1, D_2) = \frac{|D_1 \cap D_2|}{|D_1 \cup D_2|}, \quad (1)$$

where $D_1, D_2 \in \mathbb{R}^{|V|}$ are the one-hot vectors of each document example.

Based on Jaccard similarity, the unlabeled examples can be split into different subsets using the following three steps, which have been widely used in large-scale web search (Rajaraman and Ullman, 2010): 1) Shingling, 2) Min-Hashing, and 3) Locality-Sensitive Hashing (LSH).

After partition, the unlabeled set U can be converted into K different subset $\{U_1, U_2, ..., U_K\}$. Meanwhile, for each subset U_i, the first added document example S_i is recorded as the representative example of the subset U_i. Choosing representative samples will help evaluate the classifiers on different subsets and obtain the state representations, which will be discussed in 3.3.1.

3.2 Classifier Models

As mentioned before, much linguistic data naturally has two or more views, such as multi-lingual data (Wan, 2009) and document data (headline + paragraph) (Ghani, 2000; Denis et al., 2003). Based on the two views of data, we can construct two classifiers respectively. At the beginning of a training episode, the two classifiers are first seeded with a small set of labeled (seeding) training data L. At each time step t, the RL-agent makes a selection action a_t, and then the unlabeled subset U_{a_t} is selected to train the two co-training classifiers. Following the standard co-training process (Blum and Mitchell, 1998), at each time step t, the classifier C_1 annotate the unlabeled subset U_{a_t} and the pseudo-labeled U_{a_t} and the small labeled set L are then used to update the classifier C_2, vice versa. In this way, we can boost the performance of C_1 and C_2 simultaneously.

3.3 Q-Learning Agent

Q-learning is a widely used method to find an optimal action-selection policy (Watkins and Dayan, 1992). The core of our model is a Q-learning agent, which is trained to learn a good policy to select high-quality unlabeled subsets for co-training. At each time step t, the agent observes the current state s_t, and selects an action a_t from a discrete set of actions $A = \{1, 2, ..., K\}$. Based on the action a_t, the two co-training classifiers C_1 and C_2 then can be updated with the unlabeled subset U_{a_t} as described in Section 3.2. After that, the agent receives a performance-driven reward r_t and the next state observation s_{t+1}. The goal of our Q-agent at each time step t is to choose the action that can maximize the future discount reward

$$R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}, \quad (2)$$

where a training episode terminates at time T and γ is the discount factor.

3.3.1 State Representation

The state representation, in our framework, is designed to deliver the status of two co-training classifiers to the Q-agent. Zhang and Rudnicky (2006) have proved that training with high-confidence examples will consequently be a process that reinforces what the current model already encodes instead of learning an accurate distribution of data space. Thus, one insight in formulating the state representation is to add some unlabeled examples with uncertainty and diversity during the training iteration. However, too much uncertainty will make two classifiers unstable, while too much diversity will cause the sampling bias shift towards the unlabeled dataset (Yeung et al., 2017). In order to automatically capture this insight and select high-quality subsects during the iteration, the Q-agent needs to fully understand the distribution of the unlabeled data.

Based on the above intuition, we formulate the agents state using the two classifiers' probability distribution on the representative example S_i of each unlabeled subset U_i. Suppose a N-class classification problem, at each time step t, we evaluate the probability distribution of two classifiers on S_i separately. The state representation then can be defined as:

$$s_t = \{P_1^1 || P_1^2, P_2^1 || P_2^2, ..., P_K^1 || P_K^2\}_t, \quad (3)$$

where P_i^1 and P_i^2 are the probability distribution of C_1 and C_2 on S_i separately, and $||$ denotes the concatenation operation. $P_i^1, P_i^2 \in \mathbb{R}^N$ and $P_i^1 || P_i^2 \in \mathbb{R}^{2N}$. Note that the state representation is re-computed at each time step t.

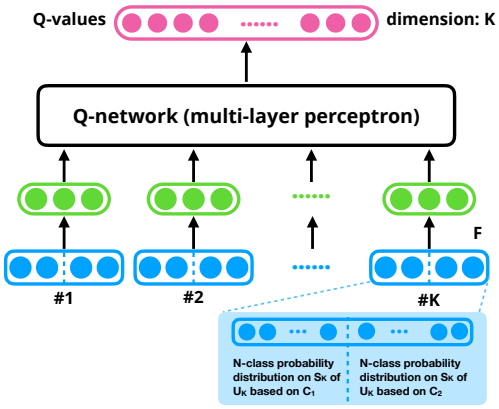


Figure 3: The structure of Q-network. It chooses a unlabeled subset from $\{U_1, U_2, ..., U_K\}$ at each time step. The state representation is computed according to the two classifiers' N-class probability distribution on the representative example S_i of each subset U_i.

3.3.2 Q-Network

The agent takes an action at at time step t using a policy

$$a_t = \max_a Q(s_t, a), \tag{4}$$

where s_t is the state representation mentioned above. The Q-value $Q(s_t, a)$ is determined by a neural network as illustrated in Figure 3. Concretely,

$$z_a = \phi(\{F(P_1^1||P_1^2), ..., F(P_K^1||P_K^2)\}; \theta), \tag{5}$$

where the function F maps state representation $P_i^1||P_i^2 \in \mathbb{R}^{2N}$ into a common embedding space of y dimensions, and $\phi(\cdot)$ is a multi-layer perception.

We then use

$$Q(s, a) = \text{softmax}(z_a) \tag{6}$$

to obtain the next action.

3.3.3 Reward Function

The agent is trained to select the high-quality unlabeled subsets to improve the performance of the two classifier C_1 and C_2. We capture this intuition by a performance-driven reward function. At time step t, the reward of each classifier is defined as the change in the classifiers accuracy after updating the unlabeled subset U_t:

$$r_t^1 = \text{Acc}_t^1(L') - \text{Acc}_{t-1}^1(L'), \tag{7}$$

where $\text{Acc}_t^1(L')$ is the model accuracy of C_1 at time step t computed on the labeled validation set L'. Then the r_t^2 is defined following the similar formulation. The final reward r_t is defined as:

$$r_t = \begin{cases} r_t^1 \times r_t^2 & \text{if } r_t^1 > 0 \text{ and } r_t^2 > 0, \\ 0 & \text{otherwise.} \end{cases}$$

Note that this reward is only available during training process.

3.4 Training and Testing

The agent is trained with the Q-learning (Watkins and Dayan, 1992), a standard reinforcement learning algorithm that can be used to learn policies for an agent interacting with an environment. In our Reinforced Co-Training framework, the environment is the classifier C_1 and C_2.

The Q-network parameters θ are learned by optimizing:

$$L_i(\theta_i) = \mathbb{E}_{s,a}[(V(\theta_{i-1}) - Q(s, a; \theta_i))^2], \tag{8}$$

where i is an iteration of optimization and

$$V(\theta_{i-1}) = \mathbb{E}_{s'}[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1})|s, a]. \tag{9}$$

.

We optimize it using stochastic gradient descent. The detail of the training process is shown in Algorithm 1.

At test time, the agent and the two co-training classifiers are again run simultaneously, but without access to the labeled validation dataset. The agent selects the unlabeled subset using the learned greedy policy:

$$a_t = max_a Q(s_t, a). \tag{10}$$

After obtaining two classifiers from co-training, based on the weighted voting, the final ensemble classifier C is defined as:

$$C = \beta C_1 + (1 - \beta) C_2. \tag{11}$$

β is the weighted parameter, which can be learned by maximizing the classification accuracy on the validation set.

4 Experiments

We evaluate our proposed Reinforced Co-training method in two settings: (1) **Clickbait detection**, where obtaining the labeled data is very time-consuming and labor-intensive in this real-world problem; (2) **Generic text classification**, where we randomly set some of the labeled data as unlabeled and train our model in a controlled setting.

Algorithm 1: The algorithm of our Reinforced Co-Training method.

Given a set L of labeled seeding training data;
Given a set L' of labeled validation data;
Given K subsets $\{U_1, U_2, ..., U_K\}$ of unlabeled data;
for *episode* $\leftarrow$ *1* **to** *M* **do**
  Train C_1 & C_2 with L
  for *time step* $t \leftarrow$ *1* **to** *T* **do**
    Choose the action $a_t = \max_a Q(s_t, a)$
    Use C_1 to label the subset U_{a_t}
    Update C_2 with pseudo-labeled U_{a_t}, L
    Use C_2 to label the subset U_{a_t}
    Update C_1 with pseudo-labeled U_{a_t}, L
    Compute the reward r_t based on L'
    Compute the state representation s_{t+1}
    Update θ using $g \propto \nabla_\theta \mathbb{E}_{s,a}[(V(\theta_{i-1}) - Q(s, a; \theta_i))^2]$

4.1 Baselines

We compare our model with multiple baselines:

- **Standard Co-Training**: Co-Training with randomly choosing unlabeled examples (Blum and Mitchell, 1998).
- **Performance-driven Co-Training**: The unlabeled examples are selected based on pseudo-accuracy and energy regularization (Zhang and Rudnicky, 2006).
- **CoTrade Co-Training**: The confidence of either classifiers prediction on unlabeled examples is estimated based on specific data editing techniques, and then high-confidence examples are used to update the classifiers (Zhang and Zhou, 2011).
- **Semi-supervised Sequence Learning (Sequence-SSL)**: The model uses an LSTM sequence autoencoder to pre-train the parameters for the later supervised learning process.(Dai and Le, 2015).
- **Semi-supervised CNN with Region Embedding (Region-SSL)**: The model learns embeddings of small text regions from unlabeled data for integration into a supervised CNN (Johnson and Zhang, 2015).
- **Adversarial Semi-supervised Learning (Adversarial-SSL)**: The model apply perturbations to word embeddings into an LSTM and pre-train the supervised models through adversarial training (Miyato et al., 2016).

Dataset	#Tweets	#Clickbait	#Non-Clickbait
Training	2,495	762	1,697
Validation	9,768	2,380	7,388
Test	9,770	2,381	7,389
Unlabeled	80,012	N/A	N/A

Table 1: Statistics of Clickbait Dataset.

4.2 Clickbait Detection

Clickbait is a pejorative term for web content whose headlines typically aim to make readers curious, but the documents usually have less relevance with the corresponding headlines (Chakraborty et al., 2016; Potthast et al., 2017; Wei and Wan, 2017). Clickbait not only wastes the readers' time but also damages the publishers' reputation, which makes detecting clickbait become an important real-world problem.

However, most of the attempts focus on news headlines, while the relevance between headlines and context is usually ignored (Chen et al., 2015; Biyani et al., 2016; Chakraborty et al., 2016). Meanwhile, the labeled data is quite limited in this problem, but the unlabeled data is easily obtained from the web (Potthast et al., 2017). Considering these two challenges, we utilize our Reinforced Co-training framework to tackle this problem and evaluate our method.

4.2.1 Datasets

We evaluate our model on a large-size clickbait dataset, Clickbait Challenge 2017 (Potthast et al., 2017). The data is collected from twitter posts including tweet headlines and paragraphs, and the training and test sets are judged on a four-point scale $[0, 0.3, 0.66, 1]$ by at least five annotators. Each sample is categorized into one class based on its average scores. The clickbait detection then can be defined as a two-class classification problem, including CLICKBAIT and NON-CLICKBAIT. There also exists an unlabeled set containing large amounts of collected samples without annotation. We then split the original test set into the validation set and final test set by 50%/50%. The statistics of this dataset are listed in Table 1.

4.2.2 Setup

For each document example in the clickbait dataset, naturally, we have two views, the headline and the paragraph. Thus, we construct the two classifiers in co-training based on these two views.

Headline Classifier The previous state-of-the-art model (Zhou, 2017) for clickbait detection uses

a self-attentive bi-directional gated recurrent unit RNN (biGRU) to model the headlines of the document and train a classifier. Following the same setting, we choose self-attentive biGRU as the headline classifier in co-training.

Paragraph Classifier The paragraphs usually have much longer sequences than the headlines. Thus, we utilize the CNN-non-static structure in Kim (2014) as the paragraph classifier to capture the paragraph information.

Note that the other three co-training baselines also use the same classifier settings.

In our Reinforce Co-Training model, we set the number of unlabeled subsets k as 80. Considering the clickbait detection as a 2-class classification problem ($N = 2$), the Q-network maps 4-d input $P_i^1||P_i^2$ in the state representation to a 3-d common embedding space ($y = 3$), with a further hidden layer of 128 units on top. The dimension k of the softmax layer is also 80.

As for the other semi-supervised baselines, Sequence-SSL, Region-SSL and Adversarial-SSL, we concatenate the headline and the paragraph as the document and train these models directly on the document data. To better analyze the experimental results, we also implement another baseline denoted as CNN (Document), which uses the CNN structure (Kim, 2014) to model the document with supervised learning. The CNN (Document) model is trained on the (seeding) training set and the validation set.

Following the previous researches (Chakraborty et al., 2016; Potthast et al., 2017), we use Precision, Recall and F1 Score to evaluate different models.

4.2.3 Results

The results of clickbait detection are shown in Table 2. From the results, we observe that: (1) Our Reinforced Co-Training model can outperform all the baselines, which indicates the capability of our methods in utilizing the unlabeled data. (2) The standard co-training is unstable due to the random data selection strategy, and the performance-driven and high-confidence data selection strategies both can improve the performance of co-training. Meanwhile, the significant improvement compared with previous co-training methods shows that the Q-agent in our model can learn a good policy to select high-quality subsets. (3) The three pre-trained based semi-supervised learning methods also show good results. We think these pre-trained based methods learn local embeddings during the unsupervised training, which may help them to recognize some important patterns in clickbait detection. (4) The self-attentive biGRU trained only on headlines of the labeled set actually show surprisingly good performance on clickbait detection, which demonstrates that most clickbait documents have obvious patterns in the headline field. The reason why CNN (Document) fails to capture these patterns may be that the concatenation of headlines and paragraphs dilutes these features. But for those cases without obvious patterns in the headline, our results demonstrate that the paragraph information is still a good supplement to detection.

Methods	**Prec.**	**Recall**	**F1 Score**
Self-attentive biGRU	0.683	0.649	0.665
CNN (Document)	0.537	0.474	0.503
Standard Co-Training	0.418	0.433	0.425
Performance Co-Training	0.581	0.629	0.604
CoTrade Co-Training	0.609	0.637	0.623
Sequence-SSL	0.595	0.589	0.592
Region-SSL	0.674	0.652	0.663
Adversarial-SSL	0.698	**0.691**	0.694
Reinforced Co-Training	**0.709**	0.684	**0.696**

Table 2: The experimental results on clickbait dataset. Prec.: precision.

	Best	**Worst**	**Average**	**STDDEV**
F1 Score	0.708	0.685	0.692	0.0068

Table 3: The robustness analysis on clickbait dataset.

4.2.4 Algorithm Robustness

Previous studies (Morimoto and Doya, 2001; Henderson et al., 2017) show that reinforcement learning-based methods usually lack robustness and are sensitive to the seeding sets and pre-trained steps. Thus, we design an experiment to detect whether our learned data section policy is sensitive to the (seeding) training set. First, based on our original data partition, we train our reinforcement learning framework to learn a Q-agent. During the test time, instead of using the same seeding set when doing comparative experiments, we randomly sample other 10 seeding sets from the labeled dataset and learn 10 classifiers based without re-training the Q-agent (data selection policy). Note that the validation set is not available during the co-training period of the test time. Finally, we evaluate these 10 classifiers using the same metric. The results are shown in Table 3.

Dataset	AG's News	DBpedia
#Classes	4	14
#Training	12,000	56,000
#Validation	12,000	56,000
#Test	7,600	70,000
#Unlabeled	96,000	448,000

Table 4: Statistics of the Text Classification Datasets.

The results demonstrate that our learning algorithm is robust to different (seeding) training sets, which indicates that the Q-agent in our model can learn a good and robust data selection policy to select high-quality unlabeled subsets to help the co-training process.

4.3 Generic Text Classification

Generic text classification is a classic problem for natural language processing, where one needs to categorized documents into pre-defined classes (Kim, 2014; Zhang et al., 2015; Johnson and Zhang, 2015, 2016; Xiao and Cho, 2016; Miyato et al., 2016). We evaluate our model on generic text classification problem to study our method in a controlled setting.

4.3.1 Datasets

Following the settings in Zhang et al. (2015), we use large-scale datasets to train and test our model. To maintain the two-view setting of the co-training method, we choose the following two datasets. The original annotated training set is then split into three sets, 10% labeled training set, 10% labeled validation set and 80% unlabeled set. The original proportion of different classes remains the same after the partition. The statistics of these two datasets are listed in Table 4.

AG's news corpus. The AGs corpus of news articles is obtained from the web and each sample has the title and description fields.

DBpedia ontology dataset. This dataset is constructed by picking 14 non-overlapping classes from DBpedia 2014. Each sample contains the title and abstract of a Wikipedia article.

4.3.2 Setup

For each document example in the above two datasets, naturally we have two views, the headline and the paragraph. Similar to clickbait detection, we also construct the two classifiers in co-training based on these two views. Following the (Kim, 2014), we set both the headline classifier and the paragraph classifier as the CNN-non-static model. Owing to that fact that the original datasets are fully labeled, we implement two other baselines: (1) CNN (Training+Validation), which is supervised trained on the partitioned training and validation sets; (2) CNN (All) which is supervised trained on the original (100%) dataset.

Methods	**AG's News**	**DBpedia**
CNN (Training+Validation)	28.32%	9.53%
CNN (All)	8.69%	0.91%
Standard Co-Training	26.52%	7.66%
Performance Co-Training	21.73%	5.84%
CoTrade Co-Training	19.06%	5.12%
Sequence-SSL	19.54%	4.64%
Region-SSL	18.27%	3.76%
Adversarial-SSL	8.45%*	0.89%*
Reinforced Co-Training	**16.64%**	**2.45%**

Table 5: The experimental results on generic text classification datasets. * Adversarial-SSL is trained on full labeled data after pre-training.

For AG's News dataset, we set the number of unlabeled subsets k as 96. The number of classes $N = 4$, and thus the Q-network maps 8-d input $P_i^1||P_i^2$ in the state representation to a 5-d common embedding space ($y = 5$), with a further hidden layer of 128 units on top. The dimension k of the softmax layer is also 96. As for DBpedia dataset, $k = 224$, $N = 14$, and $y = 10$,.

Following the previous researches (Kim, 2014), we use test error rate (%) to evaluate different models.

4.3.3 Results

The results of generic text classification are shown in Table 5. From the results, we can observe that: (1) Our Reinforced Co-Training model outperforms all the real semi-supervised baselines on two generic text classification datasets, which indicates that our method is consistent in different tasks. (2) The CNN (All) and Adversarial-SSL trained on all the original labeled data perform best, which indicates there is still an obvious gap between semi-supervised methods and full-supervised methods.

4.3.4 Algorithm Robustness

Similar to Section 4.2.4, we evaluate whether our learned data section policy is sensitive to the different partitions and (seeding) training sets. First, based on our original data partition (10%/10%/80%), we train our reinforcement learning framework. During the test time, we randomly sample other 10 data partitions instead of the one used in comparative experiments, and learn 10 ensemble classifiers based on the learned

Datasets	Best	Worst	Average	STDDEV
AG's News	14.78	17.96	16.62	1.36
DBPedia	2.18	4.06	2.75	0.94

Table 6: The robustness analysis on generic text classification. Metric: test error rate (%).

Q-agent. Note that after sample different data partitions, we will also reprocess the unlabeled sets as described in Section 3.1. We then evaluate these 10 classifiers using the same metric. The results are shown in Table 6.

The results demonstrate that our learning algorithm is robust to different (seeding) training sets and partitions of the unlabeled set, which again indicates that the Q-agent in our model is able to learn a good and robust data selection policy to select high-quality unlabeled subsets to help the co-training process.

4.4 Discussion about Stability

Previous studies (Zhang et al., 2014; Reimers and Gurevych, 2017) show that neural networks can be unstable even with the same training parameters on the same training data. As for our cases, when the two classifiers are initialized with different labeled seeding sets, they can be very unstable. However, after enough iterations with the properly selected unlabeled data, the performance would be stable generally.

Usually, the more substantial labeled training datasets will lead to more stable models. However, the problem is that the AGs News and DBpedia have 4 and 14 classes separately, while the Clickbait dataset only has 2 classes. That means the numbers of each class in AGs News, DBPedia and Clickbait actually are the same order of magnitude. Meanwhile, in our co-training setting, the prediction error is easy to accumulate because the two classifiers bootstrap the performance of each other. The classification could be harder with the increase of classes. Based on these reasons, the stability does not show a very strong correlation with the size of datasets in our experiments of Section 4.2.4 and 4.3.4.

5 Conclusion and Future Work

In this paper, we propose a novel method, Reinforced Co-Training, for training classifiers by utilizing both the labeled and unlabeled data. The Q-agent in our model can learn a good data selection policy to select high-quality unlabeled data for co-training. We evaluate our models on two tasks, clickbait detection and generic text classification. Experimental results show that our model can outperform other semi-supervised baselines, especially those conventional co-training methods. We also test the Q-agent and prove that the learned data selection policy is robust to different seeding sets and data partitions.

For future studies, we will investigate the data selection policies of other semi-supervised methods and try to learn these policies automatically. We also plan to extend our method to multi-source classification cases and utilize the multi-agent communication environment to boost the classification performance.

Acknowledgments

The authors would like to thank the anonymous reviewers for their thoughtful comments. The work was supported by an unrestricted gift from Bytedance (Toutiao).

References

Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier. 1991. Learning a synaptic learning rule. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*.

Prakhar Biyani, Kostas Tsioutsiouliklis, and John Blackmer. 2016. "8 amazing secrets for getting more click": Detecting clickbaits in news streams using article informality. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*. pages 94–100.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT)*. pages 92–100.

Abhijnan Chakraborty, Bhargavi Paranjape, Sourya Kakarla, and Niloy Ganguly. 2016. Stop clickbait: Detecting and preventing clickbaits in online news media. In *Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. pages 9–16.

Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. 2009. Semi-supervised learning. *IEEE Transactions on Neural Networks* 20(3):542–542.

Nitesh V. Chawla and Grigoris Karakoulas. 2005. Learning from labeled and unlabeled data: An empirical study across techniques and domains. *Journal of Artificial Intelligence Research* 23(1):331–366.

Yimin Chen, Niall J. Conroy, and Victoria L. Rubin. 2015. Misleading online content: Recognizing clickbait as "false new". In *Proceedings of the 2015 ACM on Workshop on Multimodal Deception Detection*. pages 15–19.

Yutian Chen, Matthew W Hoffman, Sergio Gómez Colmenarejo, Misha Denil, Timothy P Lillicrap, Matt Botvinick, and Nando Freitas. 2017. Learning to learn without gradient descent by gradient descent. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*. pages 748–756.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Proceedings of the 28th Advances in Neural Information Processing Systems (NIPS)*. pages 3079–3087.

Francois Denis, Anne Laurent, Rmi Gilleron, and Marc Tommasi. 2003. Text classification and co-training from positive and unlabeled examples. In *Proceedings of the ICML 2003 Workshop: The Continuum from Labeled to Unlabeled Data*. pages 80–87.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*. pages 1126–1135.

Rayid Ghani. 2000. Using error-correcting codes for text classification. In *Proceedings of the 17th International Conference on Machine Learning (ICML)*. pages 303–310.

Sally Goldman and Yan Zhou. 2000. Enhancing supervised learning with unlabeled data. In *Proceedings of the 17th International Conference on Machine Learning (ICML)*. pages 327–334.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. 2017. Deep reinforcement learning that matters. *arXiv preprint arXiv:1709.06560* .

Rie Johnson and Tong Zhang. 2015. Semi-supervised convolutional neural networks for text categorization via region embedding. In *Proceedings of the 28th Advances in Neural Information Processing Systems (NIPS)*. pages 919–927.

Rie Johnson and Tong Zhang. 2016. Supervised and semi-supervised text categorization using lstm for region embeddings. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*. pages 526–534.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 1746–1751.

Percy Liang. 2005. *Semi-Supervised Learning for Natural Language*. Ph.D. thesis, Massachusetts Institute of Technology.

Dougal Maclaurin, David Duvenaud, and Ryan Adams. 2015. Gradient-based hyperparameter optimization through reversible learning. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*. pages 2113–2122.

Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2016. Adversarial training methods for semi-supervised text classification. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*.

Jun Morimoto and Kenji Doya. 2001. Robust reinforcement learning. In *Proceedings of the 14th International Conference on Neural Information Processing Systems (NIPS)*. pages 1061–1067.

Kamal Nigam and Rayid Ghani. 2000. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the 9th International Conference on Information and Knowledge Management (CIKM)*. pages 86–93.

M Potthast, T Gollub, M Hagen, and B Stein. 2017. The clickbait challenge 2017: Towards a regression model for clickbait strength.

A Rajaraman and JD Ullman. 2010. Finding similar items. *Mining of Massive Datasets* 77:73–80.

Sachin Ravi and Hugo Larochelle. 2016. Optimization as a model for few-shot learning. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 338–348.

Jürgen Schmidhuber. 1987. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. Ph.D. thesis, Technische Universität München.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Proceedings of the 29th Advances in Neural Information Processing Systems (NIPS)*. pages 3630–3638.

Xiaojun Wan. 2009. Co-training for cross-lingual sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL/IJCNLP)*. pages 235–243.

Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine Learning* 8(3-4):279–292.

Wei Wei and Xiaojun Wan. 2017. Learning to identify ambiguous and misleading news headlines. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*. pages 4172–4178.

Olga Wichrowska, Niru Maheswaranathan, Matthew W Hoffman, Sergio Gómez Colmenarejo, Misha Denil, Nando Freitas, and Jascha Sohl-Dickstein. 2017. Learned optimizers that scale and generalize. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*. pages 3751–3760.

Yijun Xiao and Kyunghyun Cho. 2016. Efficient character-level document classification by combining convolution and recurrent layers. *arXiv preprint arXiv:1602.00367* .

Serena Yeung, Vignesh Ramanathan, Olga Russakovsky, Liyue Shen, Greg Mori, and Li Fei-Fei. 2017. Learning to learn from noisy web videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. pages 5154–5162.

Huaguang Zhang, Zhanshan Wang, and Derong Liu. 2014. A comprehensive review of stability analysis of continuous-time recurrent neural networks. *IEEE Transactions on Neural Networks and Learning Systems* 25(7):1229–1262.

Min-Ling Zhang and Zhi-Hua Zhou. 2011. Cotrade: Confident co-training with data editing. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 41(6):1612–1626.

Rong Zhang and Alexander I Rudnicky. 2006. A new data selection principle for semi-supervised incremental learning. In *Proceedings of the 18th International Conference on Pattern Recognition (ICPR)*. pages 780–783.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS)*. pages 649–657.

Yiwei Zhou. 2017. Clickbait detection in tweets using self-attentive network. In *Proceddings of the Clickbait Challenge*.

Zhi-Hua Zhou and Ming Li. 2005. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering* 17(11):1529–1541.

Xiaojin Zhu. 2006. Semi-supervised learning literature survey. *Technical Report 1530, Computer Science, University of Wisconsin-Madison* 2(3).

Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*.

Tensor Product Generation Networks for Deep NLP Modeling

Qiuyuan Huang[1], **Paul Smolensky**[1], **Xiaodong He**[2], **Li Deng**[1]*, **Dapeng Wu**[3]
[1]Microsoft Research, WA, USA; [2]JD AI Research; [3]University of Florida, FL, USA
{qihua,psmo}@microsoft.com; xiaodong.he@jd.com; l.deng@ieee.org
dpwu@ufl.edu

Abstract

We present a new approach to the design of deep networks for natural language processing (NLP), based on the general technique of Tensor Product Representations (TPRs) for encoding and processing symbol structures in distributed neural networks. A network architecture — the ***Tensor Product Generation Network*** (***TPGN***) — is proposed which is capable in principle of carrying out TPR computation, but which uses unconstrained deep learning to design its internal representations. Instantiated in a model for image-caption generation, TPGN outperforms LSTM baselines when evaluated on the COCO dataset. The TPR-capable structure enables interpretation of internal representations and operations, which prove to contain considerable grammatical content. Our caption-generation model can be interpreted as generating sequences of grammatical categories and retrieving words by their categories from a plan encoded as a distributed representation.

1 Introduction

In this paper we introduce a new architecture for natural language processing (NLP). On what type of principles can a computational architecture be founded? It would seem a sound principle to require that the hypothesis space for learning which an architecture provides include network hypotheses that are independently known to be suitable for performing the target task. Our proposed architecture makes available to deep learning network configurations that perform natural language generation by use of *Tensor Product Representations* (TPRs) (Smolensky and Legendre, 2006). Whether learning will create TPRs is unknown in advance, but what we can say with certainty is that the hypothesis space being searched during learning includes TPRs as one appropriate solution to the problem.

TPRs are a general method for generating vector-space embeddings of complex symbol structures. Prior work has proved that TPRs enable powerful symbol processing to be carried out using neural network computation (Smolensky, 2012). This includes generating parse trees that conform to a grammar (Cho et al., 2017), although incorporating such capabilities into deep learning networks such as those developed here remains for future work. The architecture presented here relies on simpler use of TPRs to generate sentences; grammars are not explicitly encoded here.

We test the proposed architecture by applying it to image-caption generation (on the MS-COCO dataset, (COCO, 2017)). The results improve upon a baseline deploying a state-of-the-art LSTM architecture (Vinyals et al., 2015), and the TPR foundations of the architecture provide greater interpretability.

Section 2 of the paper reviews TPR. Section 3 presents the proposed architecture, the *Tensor Product Generation Network* (TPGN). Section 4 describes the particular model we study for image captioning, and Section 5 presents the experimental results. Importantly, what the model has learned is interpreted in Section 5.3. Section 6 discusses the relation of the new model to previous work and Section 7 concludes.

2 Review of tensor product representation

The central idea of TPRs (Smolensky, 1990) can be appreciated by contrasting the TPR for a word string with a bag-of-words (BoW) vector-space embedding. In a BoW embedding, the vector that encodes `Jay saw Kay` is the same as the one that encodes `Kay saw Jay`: $\mathbf{J} + \mathbf{K} + \mathbf{s}$ where

*LD is currently at Citadel.

Proceedings of NAACL-HLT 2018, pages 1263–1273
New Orleans, Louisiana, June 1 - 6, 2018.

$\mathbf{J}, \mathbf{K}, \mathbf{s}$ are respectively the vector embeddings of the words Jay, Kay, saw.

A TPR embedding that avoids this confusion starts by analyzing Jay saw Kay as the set {Jay/SUBJ, Kay/OBJ, saw/VERB}. (Other analyses are possible: see Section 3.) Next we choose an embedding in a vector space V_F for Jay, Kay, saw as in the BoW case: $\mathbf{J}, \mathbf{K}, \mathbf{s}$. Then comes the step unique to TPRs: we choose an embedding in a vector space V_R for the ***roles*** SUBJ, OBJ, VERB: $\mathbf{r}_{\text{SUBJ}}, \mathbf{r}_{\text{OBJ}}, \mathbf{r}_{\text{VERB}}$. Crucially, $\mathbf{r}_{\text{SUBJ}} \neq \mathbf{r}_{\text{OBJ}}$. Finally, the TPR for Jay saw Kay is the following vector in $V_F \otimes V_R$:

$$\mathbf{v}_{\texttt{Jay saw Kay}} = \mathbf{J} \otimes \mathbf{r}_{\text{SUBJ}} + \mathbf{K} \otimes \mathbf{r}_{\text{OBJ}} + \mathbf{s} \otimes \mathbf{r}_{\text{VERB}} \quad (1)$$

Each word is tagged with the role it fills in the sentence; Jay and Kay fill different roles.

This TPR avoids the BoW confusion: $\mathbf{v}_{\texttt{Jay saw Kay}} \neq \mathbf{v}_{\texttt{Kay saw Jay}}$ because $\mathbf{J} \otimes \mathbf{r}_{\text{SUBJ}} + \mathbf{K} \otimes \mathbf{r}_{\text{OBJ}} \neq \mathbf{J} \otimes \mathbf{r}_{\text{OBJ}} + \mathbf{K} \otimes \mathbf{r}_{\text{SUBJ}}$. In the terminology of TPRs, in Jay saw Kay, Jay is the *filler* of the role SUBJ, and $\mathbf{J} \otimes \mathbf{r}_{\text{SUBJ}}$ is the vector embedding of the *filler/role binding* Jay/SUBJ. In the vector space embedding, the binding operation is the tensor — or generalized outer — product $\otimes$; i.e., $\mathbf{J} \otimes \mathbf{r}_{\text{SUBJ}}$ is a tensor with 2 indices defined by: $[\mathbf{J} \otimes \mathbf{r}_{\text{SUBJ}}]_{\varphi\rho} \equiv [\mathbf{J}]_{\varphi}[\mathbf{r}_{\text{SUBJ}}]_{\rho}$.

The tensor product can be used recursively, which is essential for the TPR embedding of recursive structures such as trees and for the computation of recursive functions over TPRs. However, in the present context, recursion will not be required, in which case the tensor product can be regarded as simply the matrix outer product (which cannot be used recursively); we can regard $\mathbf{J} \otimes \mathbf{r}_{\text{SUBJ}}$ as the matrix product $\mathbf{J}\mathbf{r}_{\text{SUBJ}}^{\top}$. Then Equation 1 becomes

$$\mathbf{v}_{\texttt{Jay saw Kay}} = \mathbf{J}\mathbf{r}_{\text{SUBJ}}^{\top} + \mathbf{K}\mathbf{r}_{\text{OBJ}}^{\top} + \mathbf{s}\mathbf{r}_{\text{VERB}}^{\top} \quad (2)$$

Note that the set of matrices (or the set of tensors with any fixed number of indices) is a vector space; thus Jay saw Kay $\mapsto \mathbf{v}_{\texttt{Jay saw Kay}}$ is a vector-space embedding of the symbol structures constituting sentences. Whether we regard $\mathbf{v}_{\texttt{Jay saw Kay}}$ as a 2-index tensor or as a matrix, we can call it simply a 'vector' since it is an element of a vector space: in the context of TPRs, 'vector' is used in a general sense and should not be taken to imply a single-indexed array.

Crucial to the computational power of TPRs and to the architecture we propose here is the notion of *unbinding*. Just as an *outer* product — the tensor product — can be used to *bind* the vector embedding a filler Jay to the vector embedding a role SUBJ, $\mathbf{J} \otimes \mathbf{r}_{\text{SUBJ}}$ or $\mathbf{J}\mathbf{r}_{\text{SUBJ}}^{\top}$, so an *inner* product can be used to take the vector embedding a structure and *unbind* a role contained within that structure, yielding the symbol that fills the role.

In the simplest case of orthonormal role vectors $\mathbf{r}_i$, to unbind role SUBJ in Jay saw Kay we can compute the matrix-vector product: $\mathbf{v}_{\texttt{Jay saw Kay}}\mathbf{r}_{\text{SUBJ}} = \mathbf{J}$ (because $\mathbf{r}_i^{\top}\mathbf{r}_j = \delta_{ij}$ when the role vectors are orthonormal). A similar situation obtains when the role vectors are not orthonormal, provided they are not linearly dependent: for each role such as SUBJ there is an *unbinding vector* $\mathbf{u}_{\text{SUBJ}}$ such that $\mathbf{r}_i^{\top}\mathbf{u}_j = \delta_{ij}$ so we get: $\mathbf{v}_{\texttt{Jay saw Kay}}\mathbf{u}_{\text{SUBJ}} = \mathbf{J}$. A role vector such as $\mathbf{r}_{\text{SUBJ}}$ and its unbinding vector $\mathbf{u}_{\text{SUBJ}}$ are said to be *duals* of each other. (If R is the matrix in which each column is a role vector $\mathbf{r}_j$, then R is invertible when the role vectors are linearly independent; then the unbinding vectors $\mathbf{u}_i$ are the rows of R^{-1}. When the $\mathbf{r}_j$ are orthonormal, $\mathbf{u}_i = \mathbf{r}_i$. Replacing the matrix inverse with the pseudo-inverse allows approximate unbinding if the role vectors are linearly dependent.)

We can now see how TPRs can be used to generate a sentence one word at a time. We start with the TPR for the sentence, e.g., $\mathbf{v}_{\texttt{Jay saw Kay}}$. From this vector we unbind the role of the first word, which is SUBJ: the embedding of the first word is thus $\mathbf{v}_{\texttt{Jay saw Kay}}\mathbf{u}_{\text{SUBJ}} = \mathbf{J}$, the embedding of Jay. Next we take the TPR for the sentence and unbind the role of the second word, which is VERB: the embedding of the second word is then $\mathbf{v}_{\texttt{Jay saw Kay}}\mathbf{u}_{\text{VERB}} = \mathbf{s}$, the embedding of saw. And so on.

To accomplish this, we need two representations to generate the t^{th} word: (i) the TPR of the sentence, $\mathbf{S}$ (or of the string of not-yet-produced words, $\mathbf{S}_t$) and (ii) the unbinding vector for the t^{th} word, $\mathbf{u}_t$. The architecture we propose will therefore be a recurrent network containing two subnetworks: (i) a subnet $\mathcal{S}$ hosting the representation $\mathbf{S}_t$, and a (ii) a subnet $\mathcal{U}$ hosting the unbinding vector $\mathbf{u}_t$. This is shown in Fig. 1.

3 A TPR-capable generation architecture

As Fig. 1 shows, the proposed Tensor Product Generation Network architecture (the dashed box labeled $\mathcal{N}$) is designed to support the technique

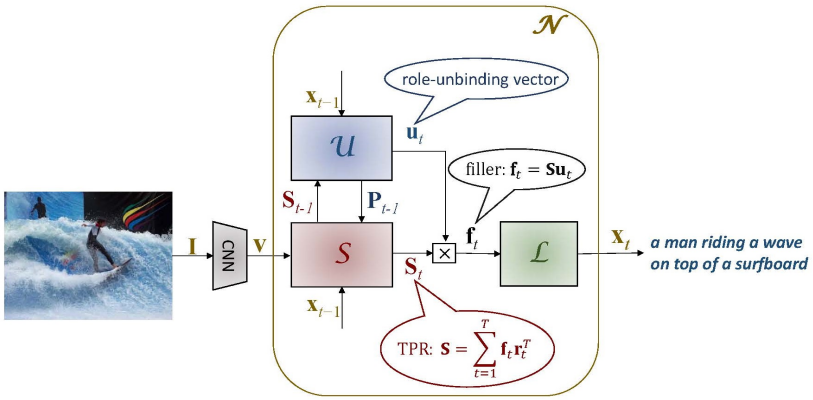


Figure 1: Architecture of TPGN, a TPR-capable generation network. "⊠" denotes the matrix-vector product.

for generation just described: the architecture is *TPR-capable*. There is a *sentence-encoding subnetwork* $\mathcal{S}$ which *could* host a TPR of the sentence to be generated, and an *unbinding subnetwork* $\mathcal{U}$ which *could* output a sequence of unbinding vectors $\mathbf{u}_t$; at time t, the embedding $\mathbf{f}_t$ of the word produced, $\mathtt{x}_t$, could then be extracted from $\mathbf{S}_t$ via the matrix-vector product (shown in the figure by "⊠"): $\mathbf{f}_t = \mathbf{S}_t\mathbf{u}_t$. The lexical-decoding subnetwork $\mathcal{L}$ converts the embedding vector $\mathbf{f}_t$ to the 1-hot vector $\mathbf{x}_t$ corresponding to the word $\mathtt{x}_t$.

Unlike some other work (Palangi et al., 2017), TPGN is not constrained to literally learn TPRs. The representations that will actually be housed in $\mathcal{S}$ and $\mathcal{U}$ are determined by end-to-end deep learning on a task: the bubbles in Fig. 1 show what *would* be the meanings of $\mathbf{S}_t$, $\mathbf{u}_t$ and $\mathbf{f}_t$ if an actual TPR scheme were instantiated in the architecture. The learned representations $\mathbf{S}_t$ will not be proven to literally be TPRs, but by analyzing the unbinding vectors $\mathbf{u}_t$ the network learns, we will gain insight into the process by which the learned matrices $\mathbf{S}_t$ give rise to the generated sentence.

The task studied here is image captioning; Fig. 1 shows that the input to this TPGN model is an image, preprocessed by a CNN which produces the initial representation in $\mathcal{S}$, $\mathbf{S}_0$. This vector $\mathbf{S}_0$ drives the entire caption-generation process: it contains all the image-specific information for producing the caption. (We will call a caption a "sentence" even though it may in fact be just a noun phrase.)

The two subnets $\mathcal{S}$ and $\mathcal{U}$ are mutually-connected LSTMs (Hochreiter and Schmidhuber, 1997): see Fig. 2. The internal hidden state of $\mathcal{U}$, $\mathbf{p}_t$, is sent as input to $\mathcal{S}$; $\mathcal{U}$ also produces output, the unbinding vector $\mathbf{u}_t$. The internal hidden state of $\mathcal{S}$, $\mathbf{S}_t$, is sent as input to $\mathcal{U}$, and also produced as output. As stated above, these two outputs are multiplied together to produce the embedding vector $\mathbf{f}_t = \mathbf{S}_t\mathbf{u}_t$ of the output word $\mathtt{x}_t$. Furthermore, the 1-hot encoding $\mathbf{x}_t$ of $\mathtt{x}_t$ is fed back at the next time step to serve as input to both $\mathcal{S}$ and $\mathcal{U}$.

What type of roles might the unbinding vectors be unbinding? A TPR for a caption could in principle be built upon *positional roles*, *syntactic/semantic roles*, or some combination of the two. In the caption `a man standing in a room with a suitcase`, the initial `a` and `man` might respectively occupy the positional roles of POS(ITION)$_1$ and POS$_2$; `standing` might occupy the syntactic role of VERB; `in` the role of SPATIAL-P(REPOSITION); while `a room with a suitcase` might fill a 5-role schema DET(ERMINER)$_1$ N(OUN)$_1$ P DET$_2$ N$_2$. In fact we will provide evidence in Sec. 5.3.2 that our network learns just this kind of hybrid role decomposition; further evidence for these particular roles is presented elsewhere.

What form of information does the sentence-encoding subnetwork $\mathcal{S}$ need to encode in $\mathbf{S}$? Continuing with the example of the previous paragraph, $\mathbf{S}$ needs to be some approximation to the TPR summing several filler/role binding matrices. In one of these bindings, a filler vector $\mathbf{f}_{\mathtt{a}}$ — which the lexical subnetwork $\mathcal{L}$ will map to the article `a` — is bound (via the outer product) to a role vector $\mathbf{r}_{\text{POS}_1}$ which is the dual of the first unbinding vector produced by the unbinding subnetwork $\mathcal{U}$: $\mathbf{u}_{\text{POS}_1}$. In the first iteration of generation the model computes $\mathbf{S}_1\mathbf{u}_{\text{POS}_1} = \mathbf{f}_{\mathtt{a}}$, which $\mathcal{L}$ then maps to `a`. Analogously, another binding approximately contained in $\mathbf{S}_2$ is $\mathbf{f}_{\mathtt{man}}\mathbf{r}_{\text{POS}_2}^{\top}$. There are corresponding approximate bindings for the remaining words

of the caption; these employ syntactic/semantic roles. One example is $\mathbf{f}_{\texttt{standing}}\mathbf{r}_V^\top$. At iteration 3, $\mathcal{U}$ decides the next word should be a verb, so it generates the unbinding vector $\mathbf{u}_V$ which when multiplied by the current output of $\mathcal{S}$, the matrix $\mathbf{S}_3$, yields a filler vector $\mathbf{f}_{\texttt{standing}}$ which $\mathcal{L}$ maps to the output `standing`. $\mathcal{S}$ decided the caption should deploy `standing` as a verb and included in $\mathbf{S}$ an approximation to the binding $\mathbf{f}_{\texttt{standing}}\mathbf{r}_V^\top$. It similarly decided the caption should deploy `in` as a spatial preposition, approximately including in $\mathbf{S}$ the binding $\mathbf{f}_{\texttt{in}}\mathbf{r}_{\text{SPATIAL-P}}^\top$; and so on for the other words in their respective roles in the caption.

4 System Description

As stated above, the unbinding subnetwork $\mathcal{U}$ and the sentence-encoding subnetwork $\mathcal{S}$ of Fig. 1 are each implemented as (1-layer, 1-directional) LSTMs (see Fig. 2); the lexical subnetwork $\mathcal{L}$ is implemented as a linear transformation followed by a softmax operation.

In the equations below, the LSTM variables internal to the $\mathcal{S}$ subnet are indexed by 1 (e.g., the forget-, input-, and output-gates are respectively $\hat{\mathbf{f}}_1, \hat{\mathbf{i}}_1, \hat{\mathbf{o}}_1$) while those of the unbinding subnet $\mathcal{U}$ are indexed by 2.

Thus the state updating equations for $\mathcal{S}$ are, for $t = 1, \cdots, T$ = caption length:

$$\hat{\mathbf{f}}_{1,t} = \sigma_g(\mathbf{W}_{1,f}\mathbf{p}_{t-1} - \mathbf{D}_{1,f}\mathbf{W}_e\mathbf{x}_{t-1} + \mathbf{U}_{1,f}\hat{\mathbf{S}}_{t-1}) \quad (3)$$

$$\hat{\mathbf{i}}_{1,t} = \sigma_g(\mathbf{W}_{1,i}\mathbf{p}_{t-1} - \mathbf{D}_{1,i}\mathbf{W}_e\mathbf{x}_{t-1} + \mathbf{U}_{1,i}\hat{\mathbf{S}}_{t-1}) \quad (4)$$

$$\hat{\mathbf{o}}_{1,t} = \sigma_g(\mathbf{W}_{1,o}\mathbf{p}_{t-1} - \mathbf{D}_{1,o}\mathbf{W}_e\mathbf{x}_{t-1} + \mathbf{U}_{1,o}\hat{\mathbf{S}}_{t-1}) \quad (5)$$

$$\mathbf{g}_{1,t} = \sigma_h(\mathbf{W}_{1,c}\mathbf{p}_{t-1} - \mathbf{D}_{1,c}\mathbf{W}_e\mathbf{x}_{t-1} + \mathbf{U}_{1,c}\hat{\mathbf{S}}_{t-1}) \quad (6)$$

$$\mathbf{c}_{1,t} = \hat{\mathbf{f}}_{1,t} \odot \mathbf{c}_{1,t-1} + \hat{\mathbf{i}}_{1,t} \odot \mathbf{g}_{1,t} \quad (7)$$

$$\hat{\mathbf{S}}_t = \hat{\mathbf{o}}_{1,t} \odot \sigma_h(\mathbf{c}_{1,t}) \quad (8)$$

Here $\hat{\mathbf{f}}_{1,t}, \hat{\mathbf{i}}_{1,t}, \hat{\mathbf{o}}_{1,t}, \mathbf{g}_{1,t}, \mathbf{c}_{1,t}, \hat{\mathbf{S}}_t \in \mathbb{R}^{d\times d}$, $\mathbf{p}_t \in \mathbb{R}^d$; $\sigma_g(\cdot)$ is the (element-wise) logistic sigmoid function; $\sigma_h(\cdot)$ is the hyperbolic tangent function; the operator $\odot$ denotes the Hadamard (element-wise) product; $\mathbf{W}_{1,f}, \mathbf{W}_{1,i}, \mathbf{W}_{1,o}, \mathbf{W}_{1,c} \in \mathbb{R}^{(d\times d)\times d}$, $\mathbf{D}_{1,f}$, $\mathbf{D}_{1,i}$, $\mathbf{D}_{1,o}$, $\mathbf{D}_{1,c} \in \mathbb{R}^{(d\times d)\times d}$, $\mathbf{U}_{1,f}$, $\mathbf{U}_{1,i}$, $\mathbf{U}_{1,o}$, $\mathbf{U}_{1,c} \in \mathbb{R}^{(d\times d)\times(d\times d)}$. For clarity, biases — included throughout the model — are omitted from all equations in this paper. The initial state $\hat{\mathbf{S}}_0$ is initialized by:

$$\hat{\mathbf{S}}_0 = \mathbf{C}_s(\mathbf{v} - \bar{\mathbf{v}}) \quad (9)$$

where $\mathbf{v} \in \mathbb{R}^{2048}$ is the vector of visual features extracted from the current image by ResNet (Gan et al., 2017) and $\bar{\mathbf{v}}$ is the mean of all such vectors; $\mathbf{C}_s \in \mathbb{R}^{(d\times d)\times 2048}$. On the output side, $\mathbf{x}_t \in \mathbb{R}^V$ is a 1-hot vector with dimension equal to the size of the caption vocabulary, V, and $\mathbf{W}_e \in \mathbb{R}^{d\times V}$ is a word embedding matrix, the i-th column of which is the embedding vector of the i-th word in the vocabulary; it is obtained by the Stanford GLoVe algorithm with zero mean (Pennington et al., 2017). $\mathbf{x}_0$ is initialized as the one-hot vector corresponding to a "start-of-sentence" symbol.

For $\mathcal{U}$ in Fig. 1, the state updating equations are:

$$\hat{\mathbf{f}}_{2,t} = \sigma_g(\hat{\mathbf{S}}_{t-1}\mathbf{w}_{2,f} - \mathbf{D}_{2,f}\mathbf{W}_e\mathbf{x}_{t-1} + \mathbf{U}_{2,f}\mathbf{p}_{t-1}) \quad (10)$$

$$\hat{\mathbf{i}}_{2,t} = \sigma_g(\hat{\mathbf{S}}_{t-1}\mathbf{w}_{2,i} - \mathbf{D}_{2,i}\mathbf{W}_e\mathbf{x}_{t-1} + \mathbf{U}_{2,i}\mathbf{p}_{t-1}) \quad (11)$$

$$\hat{\mathbf{o}}_{2,t} = \sigma_g(\hat{\mathbf{S}}_{t-1}\mathbf{w}_{2,o} - \mathbf{D}_{2,o}\mathbf{W}_e\mathbf{x}_{t-1} + \mathbf{U}_{2,o}\mathbf{p}_{t-1}) \quad (12)$$

$$\mathbf{g}_{2,t} = \sigma_h(\hat{\mathbf{S}}_{t-1}\mathbf{w}_{2,c} - \mathbf{D}_{2,c}\mathbf{W}_e\mathbf{x}_{t-1} + \mathbf{U}_{2,c}\mathbf{p}_{t-1}) \quad (13)$$

$$\mathbf{c}_{2,t} = \hat{\mathbf{f}}_{2,t} \odot \mathbf{c}_{2,t-1} + \hat{\mathbf{i}}_{2,t} \odot \mathbf{g}_{2,t} \quad (14)$$

$$\mathbf{p}_t = \hat{\mathbf{o}}_{2,t} \odot \sigma_h(\mathbf{c}_{2,t}) \quad (15)$$

Here $\mathbf{w}_{2,f}, \mathbf{w}_{2,i}, \mathbf{w}_{2,o}, \mathbf{w}_{2,c} \in \mathbb{R}^d$, $\mathbf{D}_{2,f}$, $\mathbf{D}_{2,i}$, $\mathbf{D}_{2,o}$, $\mathbf{D}_{2,c} \in \mathbb{R}^{d\times d}$, and $\mathbf{U}_{2,f}$, $\mathbf{U}_{2,i}$, $\mathbf{U}_{2,o}$, $\mathbf{U}_{2,c}$ $\in \mathbb{R}^{d\times d}$. The initial state $\mathbf{p}_0$ is the zero vector.

The dimensionality of the crucial vectors shown in Fig. 1, $\mathbf{u}_t$ and $\mathbf{f}_t$, is increased from $d\times 1$ to $d^2\times 1$ as follows. A block-diagonal $d^2 \times d^2$ matrix $\mathbf{S}_t$ is created by placing d copies of the $d \times d$ matrix $\hat{\mathbf{S}}_t$ as blocks along the principal diagonal. This matrix is the output of the sentence-encoding subnetwork $\mathcal{S}$. Now the 'filler vector' $\mathbf{f}_t \in \mathbb{R}^{d^2}$ — 'unbound' from the sentence representation $\mathbf{S}_t$ with the 'unbinding vector' $\mathbf{u}_t$ — is obtained by Eq. (16).

$$\mathbf{f}_t = \mathbf{S}_t\mathbf{u}_t \quad (16)$$

Here $\mathbf{u}_t \in \mathbb{R}^{d^2}$, the output of the unbinding subnetwork $\mathcal{U}$, is computed as in Eq. (17), where $\mathbf{W}_u \in \mathbb{R}^{d^2\times d}$ is $\mathcal{U}$'s output weight matrix.

$$\mathbf{u}_t = \sigma_h(\mathbf{W}_u\mathbf{p}_t) \quad (17)$$

Finally, the lexical subnetwork $\mathcal{L}$ produces a decoded word $\mathbf{x}_t \in \mathbb{R}^V$ by

$$\mathbf{x}_t = \sigma_s(\mathbf{W}_x\mathbf{f}_t) \quad (18)$$

where $\sigma_s(\cdot)$ is the softmax function and $\mathbf{W}_x \in \mathbb{R}^{V\times d^2}$ is the overall output weight matrix. Since $\mathbf{W}_x$ plays the role of a word de-embedding matrix, we can set

$$\mathbf{W}_x = (\mathbf{W}_e)^\top \quad (19)$$

where $\mathbf{W}_e$ is the word-embedding matrix. Since $\mathbf{W}_e$ is pre-defined, we directly set $\mathbf{W}_x$ by Eq. (19) without training $\mathcal{L}$ through Eq. (18). Note that $\mathcal{S}$ and $\mathcal{U}$ are learned jointly through end-to-end training as shown in Algorithm 1.

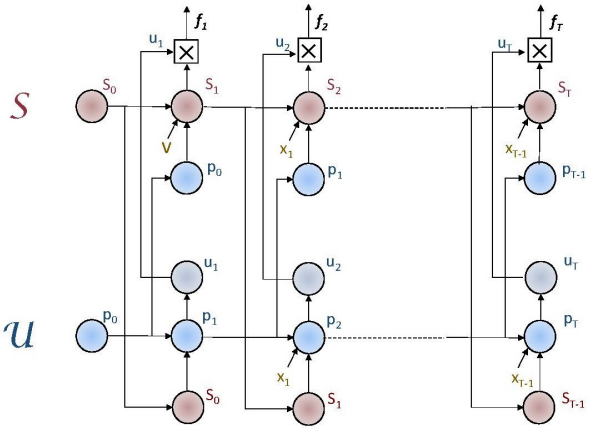


Figure 2: The sentence-encoding subnet $\mathcal{S}$ and the unbinding subnet $\mathcal{U}$ are inter-connected LSTMs; $\mathbf{v}$ encodes the visual input while the $\mathbf{x}_t$ encode the words of the output caption.

Algorithm 1 End-to-end training of $\mathcal{S}$ and $\mathcal{U}$

Input: Image feature vector $\mathbf{v}^{(i)}$ and corresponding caption $\mathbf{X}^{(i)} = [\mathbf{x}_1^{(i)}, \cdots, \mathbf{x}_T^{(i)}]$ $(i = 1, \cdots, N)$, where N is the total number of samples.
Output: $\mathbf{W}_{1,f}, \mathbf{W}_{1,i}, \mathbf{W}_{1,o}, \mathbf{W}_{1,c}, \mathbf{C}_s, \mathbf{D}_{1,f}, \mathbf{D}_{1,i}, \mathbf{D}_{1,o}, \mathbf{D}_{1,c}, \mathbf{U}_{1,f}, \mathbf{U}_{1,i}, \mathbf{U}_{1,o}, \mathbf{U}_{1,c}, \mathbf{w}_{2,f}, \mathbf{w}_{2,i}, \mathbf{w}_{2,o}, \mathbf{w}_{2,c}, \mathbf{D}_{2,f}, \mathbf{D}_{2,i}, \mathbf{D}_{2,o}, \mathbf{D}_{2,c}, \mathbf{U}_{2,f}, \mathbf{U}_{2,i}, \mathbf{U}_{2,o}, \mathbf{U}_{2,c}, \mathbf{W}_u, \mathbf{W}_x$.

Initialize $\mathbf{S}_0$ by (9);
Initialize $\mathbf{x}_0$ as the one-hot vector corresponding to the start-of-sentence symbol;
Initialize $\mathbf{p}_0$ as the zero vector;
Randomly initialize weights $\mathbf{W}_{1,f}, \mathbf{W}_{1,i}, \mathbf{W}_{1,o}, \mathbf{W}_{1,c}, \mathbf{C}_s, \mathbf{D}_{1,f}, \mathbf{D}_{1,i}, \mathbf{D}_{1,o}, \mathbf{D}_{1,c}, \mathbf{U}_{1,f}, \mathbf{U}_{1,i}, \mathbf{U}_{1,o}, \mathbf{U}_{1,c}, \mathbf{w}_{2,f}, \mathbf{w}_{2,i}, \mathbf{w}_{2,o}, \mathbf{w}_{2,c}, \mathbf{D}_{2,f}, \mathbf{D}_{2,i}, \mathbf{D}_{2,o}, \mathbf{D}_{2,c}, \mathbf{U}_{2,f}, \mathbf{U}_{2,i}, \mathbf{U}_{2,o}, \mathbf{U}_{2,c}, \mathbf{W}_u, \mathbf{W}_x$;
for n from 1 to N **do**
  for t from 1 to T **do**
    Calculate (3) – (8) to obtain $\mathbf{S}_t$;
    Calculate (10) – (15) to obtain $\mathbf{p}_t$;
    Calculate (17) to obtain $\mathbf{u}_t$;
    Calculate (16) to obtain $\mathbf{f}_t$;
    Calculate (18) to obtain $\mathbf{x}_t$;
    Update weights $\mathbf{W}_{1,f}, \mathbf{W}_{1,i}, \mathbf{W}_{1,o}, \mathbf{W}_{1,c}, \mathbf{C}_s, \mathbf{D}_{1,f}, \mathbf{D}_{1,i}, \mathbf{D}_{1,o}, \mathbf{D}_{1,c}, \mathbf{U}_{1,f}, \mathbf{U}_{1,i}, \mathbf{U}_{1,o}, \mathbf{U}_{1,c}, \mathbf{w}_{2,f}, \mathbf{w}_{2,i}, \mathbf{w}_{2,o}, \mathbf{w}_{2,c}, \mathbf{D}_{2,f}, \mathbf{D}_{2,i}, \mathbf{D}_{2,o}, \mathbf{D}_{2,c}, \mathbf{U}_{2,f}, \mathbf{U}_{2,i}, \mathbf{U}_{2,o}, \mathbf{U}_{2,c}, \mathbf{W}_u$ by the back-propagation algorithm;
  end for
end for

5 Experimental results

5.1 Dataset

To evaluate the performance of our proposed model, we use the COCO dataset (COCO, 2017). The COCO dataset contains 123,287 images, each of which is annotated with at least 5 captions. We use the same pre-defined splits as in (Karpathy and Fei-Fei, 2015; Gan et al., 2017): 113,287 images for training, 5,000 images for validation, and 5,000 images for testing. We use the same vocabulary as that employed in (Gan et al., 2017), which consists of 8,791 words.

5.2 Evaluation

For the CNN of Fig. 1, we used ResNet-152 (He et al., 2016), pretrained on the ImageNet dataset. The feature vector $\mathbf{v}$ has 2048 dimensions. Word embedding vectors in $\mathbf{W}_e$ are downloaded from the web (Pennington et al., 2017). The model is implemented in TensorFlow (Abadi et al., 2015) with the default settings for random initialization and optimization by backpropagation.

In our experiments, we choose $d = 25$ (where d is the dimension of vector $\mathbf{p}_t$). The dimension of $\mathbf{S}_t$ is 625×625 (while $\hat{\mathbf{S}}_t$ is 25×25); the vocabulary size $V = 8,791$; the dimension of $\mathbf{u}_t$ and $\mathbf{f}_t$ is $d^2 = 625$.

The main evaluation results on the MS COCO dataset are reported in Table 5.2. The widely-used BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), and CIDEr (Vedantam et al., 2015) metrics are reported in our quantitative evaluation of the performance of the proposed model. In evaluation, our baseline is the widely used CNN-LSTM captioning method originally proposed in (Vinyals et al., 2015). For comparison, we include results in that paper in the first line of Table 5.2. We also re-implemented the model using the latest ResNet features and report the results in the second line of Table 5.2. Our re-implementation of the CNN-LSTM method matches the performance reported in (Gan et al., 2017), showing that the baseline is a state-of-the-art implementation. For TPGN, we use parameter settings in a similar range to those in (Gan et al., 2017). TPGN has comparable, although slightly

Methods	METEOR	BLEU-1	BLEU-2	BLEU-3	BLEU-4	CIDEr
NIC (Vinyals et al., 2015)	0.237	0.666	0.461	0.329	0.246	0.855
CNN-LSTM	0.238	0.698	0.525	0.390	0.292	0.889
TPGN	**0.243**	**0.709**	**0.539**	**0.406**	**0.305**	**0.909**

Table 1: Performance of the proposed TPGN model on the COCO dataset.

more, parameters than the CNN-LSTM. The training time of TPGN is roughly 50% more than the CNN-LSTM model. The weights in TPGN are updated at every mini-batch; in the experiments, we use a batch size of 64 images. As shown in Table 5.2, compared to the CNN-LSTM baseline, the proposed TPGN appreciably outperforms the benchmark schemes in all metrics across the board. The improvement in BLEU-n is greater for greater n; TPGN particularly improves generation of longer subsequences. The results attest to the effectiveness of the TPGN architecture.

It is worth mentioning that this paper is aimed at developing a Tensor Product Representation (TPR) inspired network to replace the core layers in an LSTM; therefore, it is directly comparable to an LSTM baseline. So in the experiments, we focus on comparison to a strong CNN-LSTM baseline. We acknowledge that more recent papers (Xu et al., 2017; Rennie et al., 2017; Yao et al., 2017; Lu et al., 2017; Gan et al., 2017) reported better performance on the task of image captioning. Performance improvements in these more recent models are mainly due to using better image features such as those obtained by Region-based Convolutional Neural Networks (R-CNN), or using reinforcement learning (RL) to directly optimize metrics such as CIDEr, or using more complex attention mechanisms (Gan et al., 2017) to provide a better context vector for caption generation, or using an ensemble of multiple LSTMs, among others. However, the LSTM is still playing a core role in these works and we believe improvement over the core LSTM, in both performance and interpretability, is still very valuable; that is why we compare the proposed TPGN with a state-of-the-art native LSTM (the second line of Table 5.2).

5.3 Interpretation of learned unbinding vectors

To get a sense of how the sentence encodings $\mathbf{S}_t$ learned by TPGN approximate TPRs, we now investigate the meaning of the role-unbinding vector $\mathbf{u}_t$ the model uses to unbind from $\mathbf{S}_t$ — via Eq. (16) — the filler vector $\mathbf{f}_t$ that produces — via Eq. (18) — the one-hot vector $\mathbf{x}_t$ of the t^{th} generated caption word. The meaning of an unbinding vector is the meaning of the role it unbinds. Interpreting the unbinding vectors reveals the meaning of the roles in a TPR that $\mathbf{S}$ approximates.

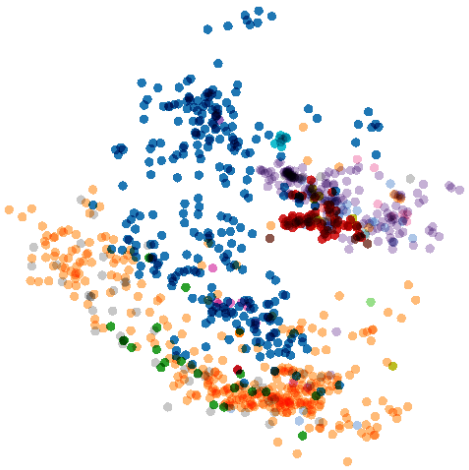


Figure 3: Unbinding vectors of 1000 words; different POS tags of words are represented by different colors.

5.3.1 Visualization of $\mathbf{u}_t$

We run the TPGN model with 5,000 test images as input, and obtain the unbinding vector $\mathbf{u}_t$ used to generate each word $\mathbf{x}_t$ in the caption of a test image. We plot 1,000 unbinding vectors $\mathbf{u}_t$, which correspond to the first 1,000 words in the resulting captions of these 5,000 test images. There are 17 parts of speech (POS) in these 1,000 words. The POS tags are obtained by the Stanford Parser (Manning, 2017).

We use the Embedding Projector in TensorBoard (Google, 2017) to plot 1,000 unbinding vectors $\mathbf{u}_t$ with a custom linear projection in TensorBoard to reduce 625 dimensions of $\mathbf{u}_t$ to 2 dimensions shown in Fig. 3 through Fig. 7.

Fig. 3 shows the unbinding vectors of 1000 words; different POS tags of words are represented by different colors. In fact, we can partition the 625-dim space of $\mathbf{u}_t$ into 17 regions, each of which

contains 76.3% words of the same type of POS on average; i.e., each region is dominated by words of one POS type. This clearly indicates that *each unbinding vector contains important grammatical information about the word it generates*. As examples, Fig. 4 to Fig. 7 show the distribution of the unbinding vectors of nouns, verbs, adjectives, and prepositions, respectively. Furthermore, we show that the subject and the object of a sentence can be distinguished based on $\mathbf{u}_t$ in (Huang et al., 2018).

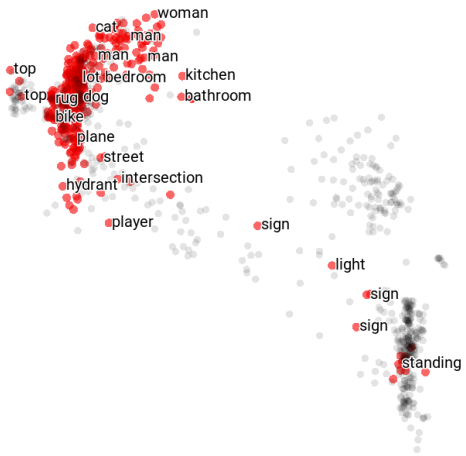


Figure 4: Unbinding vectors of 360 nouns in red and 640 words of other types of POS in grey.

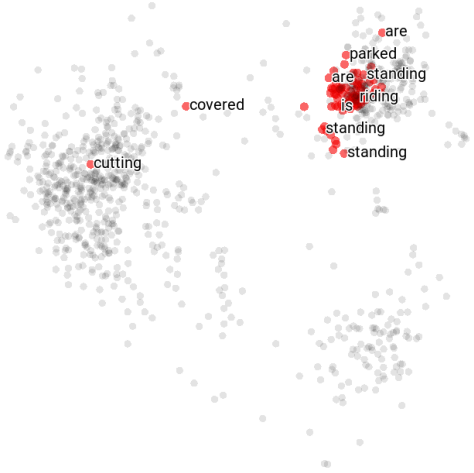


Figure 5: Unbinding vectors of 81 verbs in red and 919 words of other types of POS in grey.

5.3.2 Clustering of $\mathbf{u}_t$

Since the previous section indicates that there is a clustering structure for $\mathbf{u}_t$, in this section we partition $\mathbf{u}_t$ into N_u clusters and examine the grammar roles played by $\mathbf{u}_t$.

First, we run the trained TPGN model on the 113,287 training images, obtaining the role-

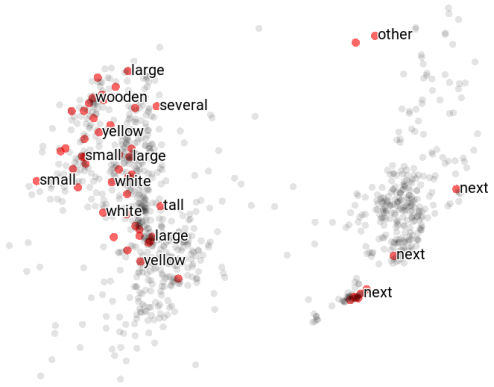


Figure 6: Unbinding vectors of 55 adjectives in red and 945 words of other types of POS in grey.

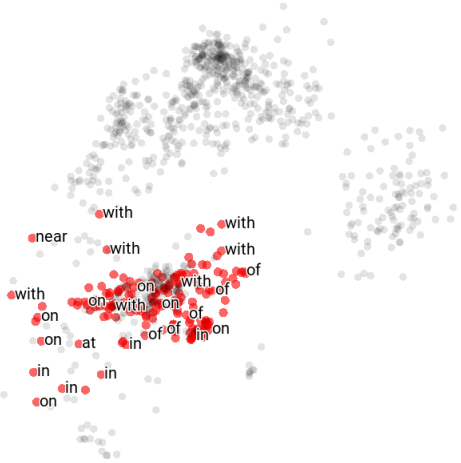


Figure 7: Unbinding vectors of 169 prepositions in red and 831 words of other types of POS in grey.

unbinding vector $\mathbf{u}_t$ used to generate each word $\mathbf{x}_t$ in the caption sentence. There are approximately 1.2 million $\mathbf{u}_t$ vectors over all the training images. We apply the K-means clustering algorithm to these vectors to obtain N_u clusters and the centroid $\boldsymbol{\mu}_i$ of each cluster i ($i = 0, \cdots, N_u - 1$).

Then, we run the TPGN model with 5,000 test images as input, and obtain the role vector $\mathbf{u}_t$ of each word $\mathbf{x}_t$ in the caption sentence of a test image. Using the nearest neighbor rule, we obtain the index i of the cluster that each $\mathbf{u}_t$ is assigned to.

The partitioning of the unbinding vectors $\mathbf{u}_t$ into $N_u = 2$ clusters exposes the most fundamental distinction made by the roles. We find that the vectors assigned to Cluster 1 generate words which are nouns, pronouns, indefinite and definite articles, and adjectives, while the vectors assigned to Cluster 0 generate verbs, prepositions, conjunctions, and adverbs. Thus Cluster 1 contains the noun-related words, Cluster 0 the verb-like words

Category	N_w	N_r	P_c
Nouns	16683	16115	0.969
Pronouns	462	442	0.957
Indefinite articles	7248	7107	0.981
Definite articles	797	762	0.956
Adjectives	2543	2237	0.880
Verbs	3558	3409	0.958
Prepositions & conjunctions	8184	7859	0.960
Adverbs	13	8	0.615

Table 2: Conformity to N/V generalization ($N_u = 2$).

ID	Interpretation (proportion)
2	Position 1 (1.00)
3	Position 2 (1.00)
1	Noun (0.54), Determiner (0.43)
5	Determiner (0.50), Noun (0.19), Preposition (0.15)
7	Noun (0.88), Adjective (0.09)
9	Determiner (0.90), Noun (0.10)
0	Preposition (0.64), . (0.16), V (0.14)
4	Preposition: spatial (0.72) non-spatial (0.19)
6	Preposition (0.59), . (0.14)
8	Verb (0.37), Preposition (0.36), . (0.20)

Table 3: Interpretation of unbinding clusters ($N_u = 10$)

(verbs, prepositions and conjunctions are all potentially followed by noun-phrase complements, for example). Cross-cutting this distinction is another dimension, however: the initial word in a caption (always a determiner) is sometimes generated with a Cluster 1 unbinding vector, sometimes with a Cluster 0 vector. Outside the caption-initial position, exceptions to the nominal/verbal $\sim$ Cluster 1/0 generalization are rare, as attested by the high rates of conformity to the generalization shown in Table 5.3.1.

Table 5.3.1 shows the likelihood of correctness of this 'N/V' generalization for the words in 5,000 sentences captioned for the 5,000 test images; N_w is the number of words in the category, N_r is the number of words conforming to the generalization, and $P_c = N_r/N_w$ is the proportion conforming. We use the Natural Language Toolkit (NLTK, 2017) to identify the part of speech of each word in the captions.

A similar analysis with $N_u = 10$ clusters reveals the results shown in Table 5.3.1; these results concern the first 100 captions, which were inspected manually to identify interpretable patterns. (More comprehensive results will be discussed elsewhere.)

The clusters can be interpreted as falling into 3 groups (see Table 5.3.1). Clusters 2 and 3 are clearly positional roles: every initial word is generated by a role-unbinding vector from Cluster 2, and such vectors are not used elsewhere in the string. The same holds for Cluster 3 and the second caption word.

For caption words after the second word, position is replaced by syntactic/semantic properties for interpretation purposes. The vector clusters aside from 2 and 3 generate words with a dominant grammatical category: for example, unbinding vectors assigned to the cluster 4 generate words that are 91% likely to be prepositions, and 72% likely to be spatial prepositions. Cluster 7 generates 88% nouns and 9% adjectives, with the remaining 3% scattered across other categories. As Table 5.3.1 shows, clusters 1, 5, 7, 9 are primarily nominal, and 0, 4, 6, and 8 primarily verbal. (Only cluster 5 spans the N/V divide.)

6 Related work

This work follows a great deal of recent caption-generation literature in exploiting end-to-end deep learning with a CNN image-analysis front end producing a distributed representation that is then used to drive a natural-language generation process, typically using RNNs (Mao et al., 2015; Vinyals et al., 2015; Devlin et al., 2015; Chen and Zitnick, 2015; Donahue et al., 2015; Karpathy and Fei-Fei, 2015; Kiros et al., 2014a,b; Xu et al., 2017; Rennie et al., 2017; Yao et al., 2017; Lu et al., 2017). Our grammatical interpretation of the structural roles of words in sentences makes contact with other work that incorporates deep learning into grammatically-structured networks (Tai et al., 2015; Kumar et al., 2016; Kong et al., 2017; Andreas et al., 2015; Yogatama et al., 2016; Maillard et al., 2017; Socher et al., 2010; Pollack, 1990). Here, the network is not itself structured to match the grammatical structure of sentences being processed; the structure is fixed, but is designed to support the learning of distributed representations that incorporate structure internal to the representations themselves — filler/role structure.

TPRs are also used in NLP in (Palangi et al., 2017) but there the representation of each individual input word is constrained to be a literal TPR filler/role binding. (The idea of using the outer product to construct internal representations was also explored in (Fukui et al., 2016).) Here, by contrast, the learned representations are not themselves constrained, but the global structure of the network is designed to display the somewhat abstract property of being TPR-capable: the archi-

tecture uses the TPR unbinding operation of the matrix-vector product to extract individual words for sequential output.

7 Conclusion

Tensor Product Representation (TPR) (Smolensky, 1990) is a general technique for constructing vector embeddings of complex symbol structures in such a way that powerful symbolic functions can be computed using hand-designed neural network computation. Integrating TPR with deep learning is a largely open problem for which the work presented here proposes a general approach: design deep architectures that are TPR-capable — TPR computation is within the scope of the capabilities of the architecture in principle. For natural language generation, we proposed such an architecture, the Tensor Product Generation Network (TPGN): it embodies the TPR operation of unbinding which is used to extract particular symbols (e.g., words) from complex structures (e.g., sentences). The architecture can be interpreted as containing a part that encodes a sentence and a part that selects one structural role at a time to extract from the sentence. We applied the approach to image-caption generation, developing a TPGN model that was evaluated on the COCO dataset, on which it outperformed LSTM baselines on a range of standard metrics. Unlike standard LSTMs, however, the TPGN model admits a level of interpretability: we can see which roles are being unbound by the unbinding vectors generated internally within the model. We find such roles contain considerable grammatical information, enabling POS tag prediction for the words they generate and displaying clustering by POS.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. https://www.tensorflow.org/.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2015. Deep compositional question answering with neural module networks. arxiv preprint. *arXiv preprint arXiv:1511.02799* 2.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*. Association for Computational Linguistics, pages 65–72.

Xinlei Chen and Lawrence Zitnick. 2015. Mind's eye: A recurrent visual representation for image caption generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 2422–2431.

Pyeong Whan Cho, Matthew Goldrick, and Paul Smolensky. 2017. Incremental parsing in a continuous dynamical system: Sentence processing in Gradient Symbolic Computation. *Linguistics Vanguard* 3. DOI:10.1515/lingvan-2016-0105.

COCO. 2017. Coco dataset for image captioning. http://mscoco.org/dataset/#download.

Jacob Devlin, Hao Cheng, Hao Fang, Saurabh Gupta, Li Deng, Xiaodong He, Geoffrey Zweig, and Margaret Mitchell. 2015. Language models for image captioning: The quirks and what works. *arXiv preprint arXiv:1505.01809* .

Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 2625–2634.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding. *arXiv preprint arXiv:1606.01847* .

Zhe Gan, Chuang Gan, Xiaodong He, Yunchen Pu, Kenneth Tran, Jianfeng Gao, Lawrence Carin, and Li Deng. 2017. Semantic compositional networks for visual captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Google. 2017. Embedding projector in tensorboard. https://www.tensorflow.org/programmers_guide/embedding.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference*

on Computer Vision and Pattern Recognition. pages 770–778.

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

Qiuyuan Huang, Li Deng, Dapeng Wu, Chang Liu, and Xiaodong He. 2018. Attentive tensor product learning for language generation and grammar parsing. *arXiv preprint arXiv:1802.07089* .

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 3128–3137.

Ryan Kiros, Ruslan Salakhutdinov, and Rich Zemel. 2014a. Multimodal neural language models. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. pages 595–603.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. 2014b. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539* .

Lingpeng Kong, Chris Alberti, Daniel Andor, Ivan Bogatyy, and David Weiss. 2017. Dragnn: A transition-based framework for dynamically connected neural networks. *arXiv preprint arXiv:1703.04474* .

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*. pages 1378–1387.

Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. 2017. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. volume 6.

Jean Maillard, Stephen Clark, and Dani Yogatama. 2017. Jointly learning sentence embeddings and syntax with unsupervised tree-lstms. *arXiv preprint arXiv:1705.09189* .

Christopher Manning. 2017. Stanford parser. https://nlp.stanford.edu/software/lex-parser.shtml.

Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. 2015. Deep captioning with multimodal recurrent neural networks (m-rnn). In *Proceedings of International Conference on Learning Representations*.

NLTK. 2017. Natural language toolkit (nltk). http://www.nltk.org.

Hamid Palangi, Paul Smolensky, Xiaodong He, and Li Deng. 2017. Deep learning of grammatically-interpretable representations through question-answering. *arXiv preprint arXiv:1705.08432* .

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 311–318.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2017. Stanford glove: Global vectors for word representation. https://nlp.stanford.edu/projects/glove/.

Jordan B Pollack. 1990. Recursive distributed representations. *Artificial Intelligence* 46(1):77–105.

Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Paul Smolensky. 1990. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial intelligence* 46(1-2):159–216.

Paul Smolensky. 2012. Symbolic functions from neural computation. *Philosophical Transactions of the Royal Society — A: Mathematical, Physical and Engineering Sciences* 370:3543 – 3569.

Paul Smolensky and Géraldine Legendre. 2006. *The harmonic mind: From neural computation to optimality-theoretic grammar. Volume 1: Cognitive architecture*. MIT Press.

Richard Socher, Christopher D Manning, and Andrew Y Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*. pages 1–9.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075* .

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 4566–4575.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 3156–3164.

Kaisheng Xu, Hanli Wang, and Pengjie Tang. 2017. Image captioning with deep lstm based on sequential residual. In *Proceedings of IEEE International Conference on Multimedia and Expo (ICME)*. pages 361–366.

Ting Yao, Yingwei Pan, Yehao Li, Zhaofan Qiu, and Tao Mei. 2017. Boosting image captioning with attributes. In *Proceedings of International Conference on Computer Vision*.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2016. Learning to compose words into sentences with reinforcement learning. *arXiv preprint arXiv:1611.09100* .

The Context-dependent Additive Recurrent Neural Net

Quan Hung Tran [1,2] **Tuan Manh Lai** [2] **Gholamreza Haffari** [1]
Ingrid Zukerman [1] **Trung Bui** [2] **Hung Bui** [3]
[1] Monash University, Clayton, Australia
[2] Adobe Research, San Jose, CA
[3] DeepMind, Mountain View, CA

Abstract

Contextual sequence mapping is one of the fundamental problems in Natural Language Processing. Instead of relying solely on the information presented in a text, the learning agents have access to a strong external signal given to assist the learning process. In this paper, we propose a novel family of Recurrent Neural Network unit: the *Context-dependent Additive Recurrent Neural Network* (*CARNN*) that is designed specifically to leverage this external signal. The experimental results on public datasets in the dialog problem (Babi dialog Task 6 and Frame), contextual language model (Switchboard and Penn Discourse Tree Bank) and question answering (TrecQA) show that our novel CARNN-based architectures outperform previous methods.

1 Introduction

Sequence mapping is one of the most prominent class of problems in Natural Language Processing (NLP). This is due to the fact that written language is sequential in nature. In English, a word is a sequence of characters, a sentence is a sequence of words, a paragraph is a sequence of sentences, and so on. However, understanding a piece of text may require far more than just extracting the information from that piece itself. If the piece of text is a paragraph of a document, the reader may have to consider it together with other paragraphs in the document and the topic of the document. To understand an utterance in a conversation, the utterance has to be put into the context of the conversation, which includes the goals of the participants and the dialog history. Hence the notion of context is an intrinsic component of language understanding.

Inspired by recent works in dialog systems (Seo et al., 2017; Liu and Perez, 2017), we formalize the contextual sequence mapping problem as a sequence mapping problem with a strong controlling contextual element that regulates the flow of information. The system has two sources of signals: (i) the main text input, for example, the history utterance sequence in dialog systems or the sequence of words in language modelling; and (ii) *the context signal*, e.g., the previous utterance in a dialog system, the discourse information in contextual language modelling or the question in question answering.

Our contribution in this work is two-fold. First, we propose a new family of recurrent unit, the *Context-dependent Additive Recurrent Neural Network* (*CARNN*), specifically constructed for contextual sequence mapping. Second, we design novel neural network architectures based on CARNN for dialog systems and contextual language modelling, and enhance the state of the art architecture (IWAN (Shen et al., 2017)) on question answering. Our novel building block, the CARNN, draws inspiration from the Recurrent Additive Network (Lee et al., 2017), which showed that most of the non-linearity in the successful Long Short Term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997) is not necessary. In the same spirit, our CARNN unit minimizes the use of non-linearity in the model to facilitate the ease of gradient flow. We also seek to keep the number of parameters to a minimum to improve trainability.

We experiment with our models on a broad range of problems: dialog systems, contextual language modelling and question answering. Our systems outperform previous methods on several public datasets, which include the Babi Task 6 (Bordes and Weston, 2017) and the Frame dataset (Asri et al., 2017) for dialog, the Switchboard (Jurafsky et al., 1997) and Penn Discourse Tree Bank (Miltsakaki et al., 2004) for contextual language modelling, and the TrecQA

Proceedings of NAACL-HLT 2018, pages 1274–1283
New Orleans, Louisiana, June 1 - 6, 2018.

dataset (Wang et al., 2007) for question answering. We propose a different architecture for each task, but all models share the basic building block, the CARNN.

2 Background and Notation

Notation. As our paper describes several architectures with vastly different setups and input types, we introduce the following notation to maintain consistency and improve readability. First, the m-th input to the recurrent unit will be denoted $\mathbf{e}_m$. In language modelling, $\mathbf{e}_m$ is the embedding of the m-th word; while in dialog, it is the embedding of the m-th utterance (which is a combination of the embedding of the words inside the utterance, $\mathbf{x}_1^m \dots \mathbf{x}_{M_m}^m$). All the gates are denoted by $\mathbf{g}$, all the hidden vectors (outputs of the RNN) are denoted by $\mathbf{h}$. $\mathbf{W}s$ and $\mathbf{b}s$ are the RNN's parameters, σ denotes the sigmoid activation function, and $\odot$ denotes the element-wise product.

LSTM. The Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) is arguably one of the most popular building blocks for RNN. The main components of the LSTM are three gates: an input gate $\mathbf{g}_m^i$ to regulate the information flow from the input to the memory cell $\mathbf{c}_m$, a forget gate $\mathbf{g}_m^f$ to regulate the information flow from the previous time step's memory cell $\mathbf{c}_{m-1}$, and an output gate $\mathbf{g}_m^o$ that regulates how the model produces the outputs (hidden state $\mathbf{h}_m$) from the memory cell $\mathbf{c}_t$. The computations of LSTM are as follows:

$$\begin{aligned}
\tilde{\mathbf{c}}_m &= tanh(\mathbf{W}_h^c \mathbf{h}_{m-1} + \mathbf{W}_x^c \mathbf{e}_m + \mathbf{b}_c) \\
\mathbf{g}_m^i &= \sigma(\mathbf{W}_h^i \mathbf{h}_{m-1} + \mathbf{W}_x^i \mathbf{e}_m + \mathbf{b}_i) \\
\mathbf{g}_m^f &= \sigma(\mathbf{W}_h^f \mathbf{h}_{m-1} + \mathbf{W}_x^f \mathbf{e}_m + \mathbf{b}_f) \\
\mathbf{g}_m^o &= \sigma(\mathbf{W}_h^o \mathbf{h}_{m-1} + \mathbf{W}_x^o \mathbf{e}_m + \mathbf{b}_o) \\
\mathbf{c}_m &= \mathbf{g}_m^i \odot \tilde{\mathbf{c}}_m + \mathbf{g}_m^f \odot \mathbf{c}_{m-1} \\
\mathbf{h}_m &= \mathbf{g}_m^o \odot tanh(\mathbf{c}_m)
\end{aligned} \tag{1}$$

RAN. The Recurrent Additive Neural Network (RAN) (Lee et al., 2017) is an improvement over the traditional LSTM. However, there are three major differences between the two. First, RAN simplifies the output computations by removing the output gate. Second, RAN simplifies the memory cell computations by removing the direct dependency between the candidate update memory cell $\tilde{\mathbf{c}}_m$ and the previous hidden vector $\mathbf{h}_{m-1}$. Finally, RAN removes the non-linearity from the transition dynamic of RNN by removing the $tanh$ non-linearity from the $\tilde{\mathbf{c}}_m$. The equations for RAN are as follows:

$$\begin{aligned}
\tilde{\mathbf{c}}_m &= \mathbf{W}_x^c \mathbf{e}_m \\
\mathbf{g}_m^i &= \sigma(\mathbf{W}_h^i \mathbf{h}_{m-1} + \mathbf{W}_x^i \mathbf{e}_m + \mathbf{b}_i) \\
\mathbf{g}_m^f &= \sigma(\mathbf{W}_h^f \mathbf{h}_{m-1} + \mathbf{W}_x^f \mathbf{e}_m + \mathbf{b}_f) \\
\mathbf{c}_m &= \mathbf{g}_m^i \odot \tilde{\mathbf{c}}_m + \mathbf{g}_m^f \odot \mathbf{c}_{m-1} \\
\mathbf{h}_m &= s(\mathbf{c}_m)
\end{aligned} \tag{2}$$

where s can be an identity function (identity RAN) or the $tanh$ activation function ($tanh$ RAN).

As shown in (Lee et al., 2017), RAN's memory cells $\mathbf{c}_m$ can be decomposed into a weighted sum of the inputs. Their experimental results show that RAN performs as well as LSTM for language modelling, while having significantly fewer parameters.

3 The Context-dependent Additive Recurrent Neural Net (CARNN)

In this section, we describe our novel recurrent units for the context-dependent sequence mapping problem.

Our RNN units use a different gate arrangement than that used by RAN. However, if we consider a broader definition of identity RAN, i.e., an RNN where hidden unit outputs can be decomposed into a weighted sum of inputs, where the weights are functions of the gates, then our first CARNN unit (nCARNN) can be viewed as an extension of identity RAN with additional controlling context.

The next two CARNN units (iCARNN and sCARNN) further simplify the nCARNN unit to improve trainability.

3.1 Non-independent gate CARNN (nCARNN)

The main components of our recurrent units are the two gates (an update gate $\mathbf{g}^u$ and a reset gate $\mathbf{g}^f$), which jointly regulate the information from the input. The input vector, after being pushed through an affine transformation, is added into the previous hidden vector $\mathbf{h}_{m-1}$. The computations of the unit are as follows:

$$\begin{aligned}
\mathbf{g}_m^u &= \sigma(\mathbf{W}_u^c \mathbf{c} + \mathbf{W}_u^h \mathbf{h}_{m-1} + \mathbf{W}_u^e \mathbf{e}_m + \mathbf{b}_u) \\
\mathbf{g}_m^f &= \sigma(\mathbf{W}_f^c \mathbf{c} + \mathbf{W}_f^h \mathbf{h}_{m-1} + \mathbf{W}_f^e \mathbf{e}_m + \mathbf{b}_f) \\
\bar{\mathbf{e}}_m &= \mathbf{W}_{\bar{e}} \mathbf{e}_m + \mathbf{b}_{\bar{e}} \\
\mathbf{h}_m &= \mathbf{g}_m^u \odot (\mathbf{g}_m^f \odot \bar{\mathbf{e}}_m) + (1 - \mathbf{g}_m^u) \odot \mathbf{h}_{m-1}
\end{aligned} \tag{3}$$

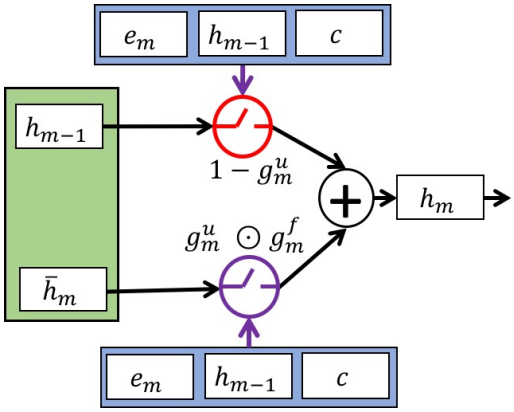


Figure 1: Context Dependent Additive Recurrent Neural Network. Note that only nCARNN has the previous hidden state $\mathbf{h}_{m-1}$ in its gate computation, iCARNN and sCARNN do not.

where $\mathbf{c}$ is the representation of the global context.

Apart from the non-linearity in the gates, our model is a linear function of the inputs. Hence, the final hidden layer of our RNN, denoted as $\mathbf{h}_M$, is a weighted sum of the inputs and a bias term $\mathbf{B}_i$ (Equation 4), where the weights are functions of the gates and $\mathbf{W}_{\bar{e}}$ is a dimension reduction matrix.

$$\begin{aligned}\mathbf{h}_M &= \mathbf{g}_M^u \odot \mathbf{g}_M^f \odot \bar{\mathbf{e}}_M + (1 - \mathbf{g}_M^u) \odot \mathbf{h}_{M-1} \\ &= \sum_{i=1}^{M} (\mathbf{g}_i^u \odot \mathbf{g}_i^f \odot \prod_{j=i+1}^{M} (1 - \mathbf{g}_j^u)) \odot \bar{\mathbf{e}}_i \\ &= \sum_{i=1}^{M} [(\mathbf{g}_i^u \odot \mathbf{g}_i^f \odot \prod_{j=i+1}^{M} (1 - \mathbf{g}_j^u)) \odot \mathbf{W}_{\bar{e}} \mathbf{e}_i + \mathbf{B}_i]\end{aligned} \tag{4}$$

From the decomposition in Equation 4, it seems that the outputs of an RNN with the nCARNN unit can be efficiently computed in parallel. That is, we can compute the weight for each input in parallel, and take their weighted sum to produce any desired hidden vector output. However, there is one obstacle: since the gates are functions of the previous hidden states, they still need to be computed sequentially. But if we assume that the external controlling context $\mathbf{c}$ is strong enough to regulate the flow of information, we can remove the previous hidden state (local context $\mathbf{h}_{m-1}$) from the gate computations, and make the RNN computations parallel. The next two variants of CARNN implement this idea by removing the local context from gate computations.

3.2 Independent gate CARNN (iCARNN)

The Gated Recurrent Unit (GRU) (Chung et al., 2014) and LSTM networks use a local context (the previous hidden state $\mathbf{h}_{m-1}$) and the current input to regulate the flow of information. In contrast, our model, relies on the global controlling context $\mathbf{c}$ at every step, and thus, might not need the local context $\mathbf{h}_{m-1}$ at all. Removing the local context can reduce the computational complexity of the model, but it may result in a loss of local sequential information. To test the effectiveness of this trade-off, we propose another variant of our unit, the *independent gate CARNN* (*iCARNN*), where the gate computations are simplified, and the gates are functions of the controlling context and the inputs. This formulation of CARNN is formally defined as follows.

$$\begin{aligned}\mathbf{g}_m^u &= \sigma(\mathbf{W}_u^c \mathbf{c} + \mathbf{W}_u^e \mathbf{e}_m + \mathbf{b}_u) \\ \mathbf{g}_m^f &= \sigma(\mathbf{W}_f^c \mathbf{c} + \mathbf{W}_f^e \mathbf{e}_m + \mathbf{b}_f) \\ \bar{\mathbf{e}}_m &= \mathbf{W}_{\bar{e}} \mathbf{e}_m + \mathbf{b}_{\bar{e}} \\ \mathbf{h}_m &= \mathbf{g}_m^u \odot (\mathbf{g}_m^f \odot \bar{\mathbf{e}}_m) + (1 - \mathbf{g}_m^u) \odot \mathbf{h}_{m-1}\end{aligned} \tag{5}$$

Compared to the traditional RNN, iCARNN's gates computations do not take into account the sequence context, i.e., the previous hidden vector computations, and the gates at all time steps can be computed in parallel. However, iCARNN, unlike memory network models (Sukhbaatar et al., 2015; Liu and Perez, 2017), still retains the sequential nature of RNN. This is because even though the gates at different time steps do not depend on each other, the hidden vector output at the m-th time step $\mathbf{h}_m$ depends on the previous gate ($\mathbf{g}_{m-1}^u$), and hence on the previous input.

3.3 Simplified candidate CARNN (sCARNN)

The standard GRU and the LSTM employ a linear transformation on the input representation before it is incorporated into the hidden representation. We have followed this convention with the previous variants of our unit. Although this transformation improves dimensional flexibility of the input/output vector, and adds representational power to the model with additional parameters, it also increases computational complexity. Fixing the output dimension to be the same as the input dimension makes it possible to reduce the computational complexity of the model. This leads us to propose another variant of the CARNN where the candidate update $\bar{\mathbf{e}}_m$ is the original embedding of the

current input (Equation 6). We call this variation the *simplified candidate CARNN* (*sCARNN*). The combination of lower gate computational complexity and the parallel-ability allow the paralleled sCARNN version to be 30% faster (30% lower training time for each epoch) than nCARNN in the question answering and dialog experiments, and 15% faster in the language model experiment. The sCARNN is formally defined as follows.

$$\begin{aligned}\mathbf{g}_m^u &= \sigma(\mathbf{W}_u^c\mathbf{c} + \mathbf{W}_u^e\mathbf{e}_m + \mathbf{b}_u)\\ \mathbf{g}_m^f &= \sigma(\mathbf{W}_f^c\mathbf{c} + \mathbf{W}_f^e\mathbf{e}_m + \mathbf{b}_f)\\ \mathbf{h}_m &= \mathbf{g}_m^u \odot (\mathbf{g}_m^f \odot \mathbf{e}_m) + (1 - \mathbf{g}_m^u) \odot \mathbf{h}_{m-1}\end{aligned} \tag{6}$$

sCARNN can still be decomposed into a weighted sum of the sequence of input elements, and retains the parallel computation capability of the iCARNN.

$$\begin{aligned}\mathbf{h}_M &= \mathbf{g}_M^u \odot \mathbf{g}_M^f \odot \mathbf{e}_m + (1 - \mathbf{g}_M^u) \odot \mathbf{h}_{M-1}\\ &= \sum_{i=1}^{M}(\mathbf{g}_i^u \odot \mathbf{g}_i^f \odot \prod_{j=i+1}^{M}(1 - \mathbf{g}_j^u)) \odot \mathbf{e}_i\end{aligned} \tag{7}$$

4 CARNN-based models for NLP problems

In this section, we explain the details of our CARNN-based architectures for end-to-end dialog, language modelling and question answering. In each of these applications, one of the main design concerns is the choice of contextual information. As we will demonstrate in this section, the controlling context $\mathbf{c}$ can be derived from various sources: a sequence of words (dialog and question answering), a class variable (language modelling). Virtually any sources of strong information that can be encoded into vectors can be used as controlling context.

4.1 End-to-end dialog

To produce a response, we first encode the whole dialog history into a real vector representation $\mathbf{h}_{his}$. To this effect, we perform two steps: first, we encode each utterance (sequence of words) into a real vector, and next, we encode this sequence of real vector representations into $\mathbf{h}_{his}$. We employ the Position Encoder (Bordes and Weston, 2017) for the first step, and CARNNs for the second step.

Summarizing individual utterances. Let's denote the sequence of word-embeddings in the m-th utterance $\mathbf{x}_1^m, \ldots \mathbf{x}_{N_m}^m$. These word embeddings are jointly trained with the model. Following previous work in end-to-end dialog systems, we opt to use the Position Encoder (Liu and Perez, 2017; Bordes and Weston, 2017) for encoding utterances.

The Position Encoder is an improvement over the average embedding of bag of words, as it takes into account the position of the words in a sequence. This encoder has been empirically shown to perform well on the Babi dialog task (Liu and Perez, 2017; Bordes and Weston, 2017); more details about the Position Encoder can be found in (Sukhbaatar et al., 2015). Let's denote the the embeddings of a sequence of utterances $\mathbf{e}_1, \ldots \mathbf{e}_{M-1}$.

Summarizing the dialog history. The CARNN models take the embeddings of the sequence of utterances and produce the final representation $\mathbf{h}_{his}$. We further enhance the output of the CARNN by adding the residual connection to the input (He et al., 2016; Tran et al., 2017), and the attention mechanism (Bahdanau et al., 2015) over the history.

$$\begin{aligned}\mathbf{h}_1, ..\mathbf{h}_{M-1} &= CARNN(\mathbf{e}_1, ..\mathbf{e}_{M-1}, \mathbf{c})\\ \forall m \in [1..M-1] &: \tilde{\mathbf{h}}_m = \mathbf{h}_m + \mathbf{e}_m\\ \alpha_1..\alpha_{M-1} &= softmax(\tilde{\mathbf{h}}_1^T\mathbf{c}, .., \tilde{\mathbf{h}}_{M-1}^T\mathbf{c})\\ \mathbf{h}_{his} &= \sum_{m=1}^{M-1}\alpha_m\tilde{\mathbf{h}_m}\end{aligned} \tag{8}$$

where α are the attention weights, $\mathbf{h}_m$ is the m-th output of the base CARNN, $\mathbf{e}_m$ is the embedding of the m-th input utterance, and $\mathbf{c} = \mathbf{e}_M$ is the context embedding.

Our model chooses the response from a set of pre-determined system answers (a task setup following Bordes and Weston (2017); Liu and Perez (2017); Seo et al. (2017)). However, in the dialog case, the answers themselves are sequences of words, and treating them as distinct classes may not be the best approach. In fact, previous work in memory networks (Liu and Perez, 2017; Bordes and Weston, 2017) employs a feature function Φ to extract features from the candidate responses. In our work, we do not use any feature extraction, and simply use the Position Encoder to encode the

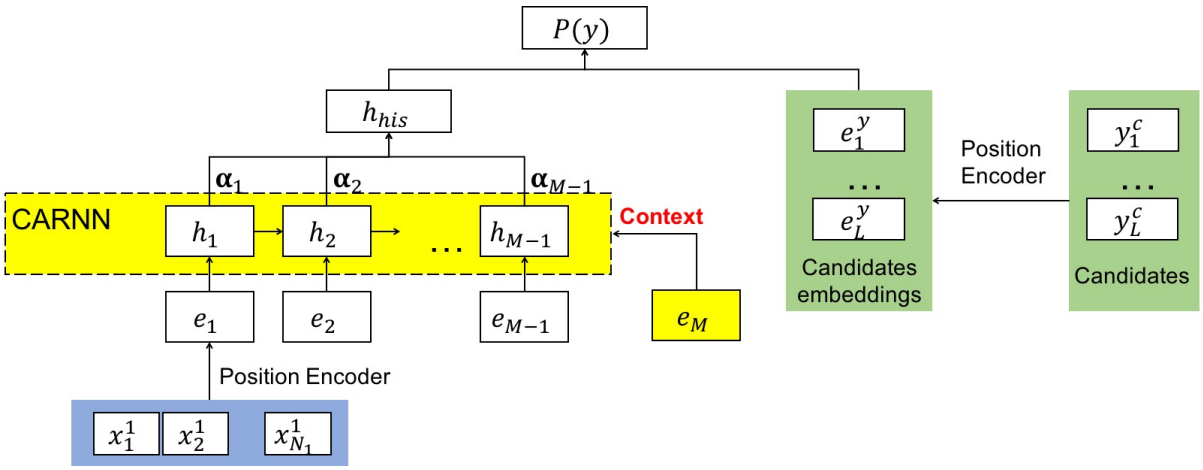


Figure 2: CARNN for dialog.

responses as shown in Figure 2, which depicts our architecture of CARNN for dialog.

$$\forall l \in [1..L] : \mathbf{e}_l = Position\_Encoder(y_l^c) \quad (9)$$

We then put a distribution over the candidate responses conditioned on the summarized dialog history $\mathbf{h}_{his}$ (Equation 10).

$$\mathbb{P}(y) = softmax(\mathbf{h}_{his}^T \mathbf{e}_1^y, ..., \mathbf{h}_{his}^T \mathbf{e}_L^y) \quad (10)$$

4.2 Contextual language model

Typically, language models operate at the sentence level, i.e., the sentences are treated independently. Several researchers have explored inter-sentence and inter-document level contextual information for language modelling (Ji et al., 2016a,b; Tran et al., 2016; Lau et al., 2017).

Following Ji et al. (2016a,b), we investigate two types of contextual information: (i) the previous sentence context; and (ii) a latent variable capturing the connection information between sentences, such as discourse relation in the Penn Discourse Tree Bank dataset or Dialog Acts in the Switchboard dataset.

Previous sentence context. The previous sentence (time-step $t-1$) contextual information is encoded by a simplified version of the nCARNN, where the global context is absent. The final hidden vector of this sequence is then fed into the current recurrent computation (time-step t) as the context for that sequence. Equation 11 shows this procedure.

$$\begin{aligned} \mathbf{c}^{t-1} &\longleftarrow nCARNN(\mathbf{e}_1^{t-1}, ..\mathbf{e}_{M^{t-1}}^{t-1}) \\ \mathbf{h}_1^t, ..\mathbf{h}_{M^{t-1}}^t &= CARNN(\mathbf{e}_1^t, ..\mathbf{e}_{M^t}^t, \mathbf{c}^{t-1}) \\ w_{m+1}^t &\sim softmax(\mathbf{W}^{(l)} \mathbf{h}_m^t + \mathbf{b}^{(l)}) \end{aligned} \quad (11)$$

Latent variable context. Ji et al. (2016b) proposed to embed the predicted latent variables using an embedding matrix, and use this real vector as the contextual information. In our work, we design a multi-task learning scenario where the previous sentence context encoder has additional supervised information obtained from the annotated latent variable (L^{t-1}). This additional information from the latent variable is only used to train the previous sentence encoder, and enhance the context $\mathbf{c}^{t-1}$ (Equation 12). During test time, the language model uses the same computation steps as the previous sentence context version.

$$\begin{aligned} \mathbb{P}(L^{t-1}) &= softmax(\mathbf{W}^{(c)} \mathbf{c}^{t-1} + \mathbf{b}^{(c)}) \\ \mathbb{L}_l^{t-1} &\sim \mathbb{P}(L^{t-1}) \end{aligned} \quad (12)$$

During training, the total loss function ($\mathbb{L}_{l,w}^t$) is the linear combination of the average log-loss from the current sentence's words ($\mathbb{L}_w^t$) and the log-loss from the previous latent variable ($\mathbb{L}_l^{t-1}$).

$$\mathbb{L}_{l,w}^t = \alpha \mathbb{L}_w^t + (1-\alpha)\mathbb{L}_l^{t-1} \quad (13)$$

where α is a linear mixing parameter. In our experiments, tuning α does not yield significant improvements, hence we set $\alpha = 0.5$.

4.3 Question answering

Answer selection is an important component of a typical question answering system. This task can be briefly described as follows: Given a question q and a candidate set of sentences $c_1, c_2, \ldots c_n$, the goal is to identify positive sentences that contain the answer. Many researchers have investigated employing neural networks for this task (Rao

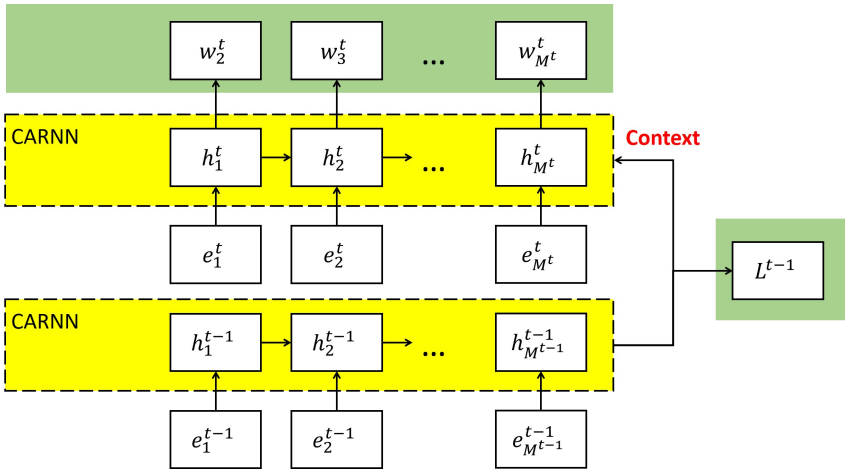


Figure 3: CARNN for context-dependent language model.

et al., 2016; Wang et al., 2017; Bian et al., 2017; Shen et al., 2017; Tay et al., 2017; He et al., 2015). Below is an example from the answer selection TrecQA corpus:

> **Question**: Who established the Nobel prize awards?
> **Positive answer**: The Nobel Prize was established in the will of Alfred Nobel, a Swede who invented dynamite and died in 1896.
> **Negative answer**: The awards aren't given in specific categories.

The IWAN model proposed in (Shen et al., 2017) achieves state-of-the-art performance on the Clean version TrecQA dataset (Wang et al., 2007) for answer selection. In general, given two sentences, the model aims to calculate a score to measure their similarity. For each sentence, the model first uses a bidirectional LSTM to obtain a context-aware representation for each position in the sentence. The representations will later be utilized by the model to compute similarity score of the two sentences according to the degree of their alignment (Shen et al., 2017).

The original IWAN model employed LSTM to encode the sentence pair into sequences of real vector representations. However, these sequences are independent, and do not take into account the information from the other sentence. In order to overcome this limitation, we enhance the IWAN model with a "cross context CARNN-based sentence encoder" that replaces the bidirectional LSTM. When the cross context CARNN sentence encoder processes a sentence, it takes the encoding of the other sentence, encoded by a Position Encoder, as the controlling context (Figure 4).

5 Experiments

5.1 End-to-end dialog

Datasets. For the dialog experiments, we focus on two popular datasets for dialog: the Babi dataset (Bordes and Weston, 2017) and the Malluba Frame dataset (Asri et al., 2017).[1]

In our main set of experiments for dialog, we use the original Babi task 6 dataset, and test on the end-to-end dialog setting (the same setting used by Seo et al. (2017); Bordes and Weston (2017); Liu and Perez (2017)). That is, the systems have to produce complete responses and learn the dialog behaviour solely from the ground truth responses without help from manual features, rules or templates. Apart from this main set of experiments, we apply our end-to-end systems as dialog managers and test on a slightly different setting in the next two sets of experiments.

In the second set of experiments, we use our end-to-end systems as "dialog managers". The only difference compared to the end-to-end dialog setting is that the systems produce templatized responses instead of complete responses. Our motivation for this dialog manager setting is that in our preliminary experiments with the Babi dataset, we found out that many of the classification errors are due to very closely related responses, all of which fit the corresponding context. We argue that if we treat the systems as dialog managers, then we can delexicalize and group similar responses. Thus following Williams et al. (2017), we construct a templatized set of responses. For example, all the

[1] Among the Babi tasks, we focus mainly on task 6, which is based on real human-machine interactions. The other five Babi datasets comprise synthetically generated data.

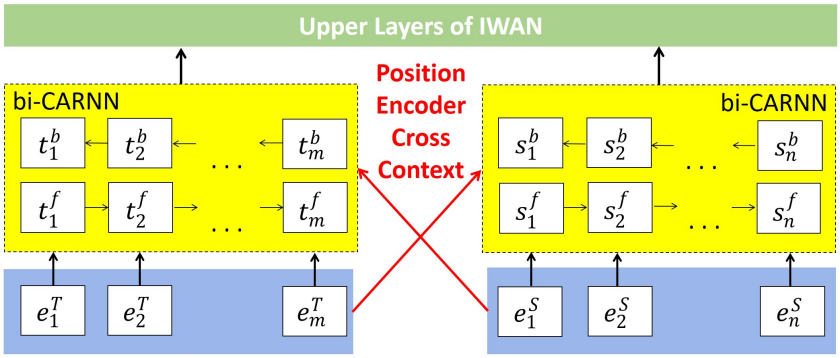


Figure 4: CARNN for Question Answering.

responses similar to *"**india_house** is in the **west** part of town"* will be grouped into *"**_name_** is in the **_loc_** part of town"*. The set of responses is reduced to 75 templatized responses. We call this new dataset "Babi reduced".[2]

The third set of experiments is conducted on the Frame dataset. The general theme in this dataset is similar to that of the Babi task 6, but the responses in the Frame dataset are generally in free form, rather than being sourced from a limited set. Thus, we define a dialog task on the Frame data set similar to the Babi reduced dialog task by simplifying and grouping the responses.[3] The final set of responses consists of 129 response classes. For the experiments on the Frame dataset, we randomly choose 80% of the conversations as the training set, and 10% each for testing and development.

Baselines. In the dialog experiments, we focus on the existing published results with end-to-end settings, namely the Memory Network (MN) (Bordes and Weston, 2017), the Gated Memory Network (GMN) (Liu and Perez, 2017) and the Query Reduction Network (QRN) (Seo et al., 2017).[4] For the Frame and Babi reduced datasets, we use the publicly available implementation of the QRN,[5] and our implementation of the GMN with hyper parameters similar to those reported by Liu and Perez (2017); Seo et al. (2017). Note that the original results presented by Seo et al. (2017), take into account partial matches (matching only a portion of the ground truth response), and hence cannot be directly translated into the standard response accuracy reported by other researchers (we have confirmed this with Seo *et al.*). For a direct comparison with the QRN, we use the evaluation settings employed in other papers (Liu and Perez, 2017; Sukhbaatar et al., 2015).

Model	Babi	Babi reduced	Frame
nCARNN	51.3%*	55.8%*	27.4%*
iCARNN	52.0%*	55.2%*	28.5%*
sCARNN	50.9%*	55.9%*	25.7%*
CARNN voting	**53.2%***	**56.9%***	**29.1%***
QRN (2017)	46.8%	54.7%	24.0%
GMN (2017)	47.4%	54.1%	23.6%
MN (2017)	41.1%	–	–

Table 1: Dialog accuracy on Babi and Frame among end-to-end systems. * indicates statistical significance with $p < 0.1$ compared to QRN.

Results and discussion. Table 1 shows the results of the end-to-end models for the dialog task. All the CARNN-based systems are implemented in Tensorflow (Abadi et al., 2015) with a hidden vector size of 1024. As seen in Table 1, our models achieve the best results, and within the variants of our models, the iCARNN either performs the best, or very close to the best on all datasets. Majority voting provides a significant boost to the performance of the CARNN models. Upon comparison with the baseline systems, CARNN models tend to perform better on instances which require the system to remember specific information through a long dialog history. In Figure 5, the user already mentioned that he/she wants to find a "cheap" restaurant, but the GMN and QRN seem to "forget" this information. We speculate that due

[2]We do not have access to Williams et al. (2017)'s template set, thus the results in Babi reduced are not comparable to those obtained by Williams et al. (2017).

[3]We use only one of the annotated *"Dialog acts"* and its first slot key as a template for the response.

[4]Williams et al. (2017) and Liu and Lane (2017) reported very strong performances (55.6% and 52.8% respectively) for the Babi dataset. However, these systems do not learn dialog behaviour solely from Babi's ground truth responses, and thus do not have end-to-end dialog setups. As stated in their papers, Williams *et al.*use hand-coded rules and task-specific templates, while Liu *et al.*employ the external users' goal annotations that are outside the Babi dataset.

[5]https://github.com/uwnlp/qrn

U: im looking for a cheap restaurant
S: ... What type of food do you want??
...5 dialog turns...
S: Could you please repeat that?
U: vietnamese food

CARNN action: api_call vietnamese R_location cheap
QRN action: api_call vietnamese R_location R_price
GMN action: api_call vietnamese R_location R_price

Figure 5: Sample dialog from our system compared to the baselines. Only CARNN's predicted action takes into account the original cheap restaurant request and matches the ground truth action (in the systems' api calls, "R_price" denotes "any price").

to the ease of training, CARNN models summarize the dialog history better, and allow for longer information dependency.

The CARNN units are originally designed in the dialog context. During model calibration, we also tested in the dialog experiments two other CARNN versions with both higher and lower complexity. The lower complexity CARNN version resembles sCARNN without the forget gate, and the higher complexity CARNN version resembles the LSTM unit with all three gates (forget, update and output), with the gates being modified from the original LSTM gates to be functions of the external contextual information. Both of these versions do not perform as well as the three main CARNN versions (48.7% and 48.6% for the high- and low-complexity versions respectively in the Babi task).

5.2 Contextual language model

Datasets. We employ two datasets for the experiments with the contextual language model: the Switchboard Dialog Act corpus and the Penn Discourse Tree Bank corpus. There are 1155 telephone conversations in the Switchboard corpus, where each conversation has an average of 176 utterances. There were originally 226 Dialog Act (DA) labels in the corpus, but they are usually clustered into 42 labels. The Penn Tree Bank corpus provides discourse relation annotation between the spans of text. We used the preprocessed data by Ji et al. (2016b), where the explicit discourse relations are mapped into a dummy relation. Our data splits are the same as those described in the baselines (Ji et al., 2016a,b).

Baselines. We compare our system with the Recurrent Neural Net (RNNLM) with LSTM unit (Ji et al., 2016a), the Document Contextual Language Model (DCLM) (Ji et al., 2016a) and the Discourse Relation Language Model (DRLM) (Ji et al., 2016b). The RNNLM's architecture is the same as that described in (Mikolov et al., 2013) with sigmoid non-linearity replaced by LSTM. The DCLM exploits the inter-sentences context by concatenating the representation of the previous sentence with the input vector (context-to-context) or the hidden vector (context-to-output). The DRLM introduces the latent variable contextual models using a generative architecture that treats Dialog Acts or discourse relations as latent variables.

Model	Penn Discourse Tree Bank	Switchboard
nCARNN (w/o latent)	96.95	30.17
iCARNN (w/o latent)	94.72	32.49
sCARNN (w/o latent)	87.39	31.50
nCARNN (with latent)	96.64	**29.72**
iCARNN (with latent)	94.16	32.16
sCARNN (with latent)	**86.68**	31.49
RNNLM (2016b)	117.8	56.0
DCLM (2016a)	112.2	45.3
DRLM (2016b)	108.3	39.6

Table 2: Perplexity on Switchboard and Penn Discourse Tree Bank.

Model	MAP	MRR
IWAN (our implementation) + nCARNN*	0.827	0.889
IWAN (our implementation) + iCARNN*	0.826	**0.907**
IWAN (our implementation) + sCARNN*	**0.829**	0.875
IWAN (our implementation)	0.794	0.879
IWAN (2017)	0.822	0.889
Compare-Aggregate (2017)	0.821	0.899
BiMPM (2017)	0.802	0.875
NCE-CNN (2016)	0.801	0.877
HyperQA (2017)	0.784	0.865

Table 3: MAP and MRR for question answering. * indicates statistical significance with $\alpha < 0.05$ in t-test compared to IWAN (our implementation).

Results and discussion. Table 2 shows the test set perplexities across the systems for the Penn Tree Bank and Switchboard datasets. Interestingly, in these experiments, the system with the least computational complexity, the sCARNN, performs best on Penn Discourse Tree Bank, and second best on Switchboard. Generally, we found out that adding the Dialog Act/Discourse supervised signal in a multi-task learning scheme provides a boost to performance, but this improvement is small.

5.3 Question answering

Datasets. The TrecQA dataset (Wang et al., 2007) is a widely-used benchmark for answer selection. There are two versions of TrecQA: original and clean. The original TrecQA consists of 1,229 training questions, 82 development questions, and 100 test questions. Recently, researchers (Rao et al., 2016; Shen et al., 2017) developed a clean version, where they removed questions in the development and test sets with no answers or only positive/negative answers. This reduced the development and test set's sizes to 65 and 68 questions respectively.

Baselines. We compare the performance of our models with that of the state-of-the-art models on the clean version of the TREC-QA dataset (Shen et al., 2017; Bian et al., 2017; Wang et al., 2017; Rao et al., 2016; Tay et al., 2017). We do not have access to the original implementation of IWAN, hence we use our implementation of the IWAN model as the basis for our models.

Results and discussion. Table 3 shows the MAP (Mean Average Precision) and MRR (Mean Reciprocal Rank) of our systems and the baselines. To the best of our knowledge, our systems outperform all previous systems on this dataset. Enhancing IWAN with cross-context CARNN statistically significantly improves performance. Among the variants, the iCARNN is the most consistent in both MAP and MRR. During our error analysis, we noted that the top answer returned by IWAN models with either LSTM or CARNNs are usually good. However, in many cases, lower ranked answers returned by the LSTM model are not as good as those produced by the CARNN models. We show an example of this in Table 4.

6 Conclusion and future work

In this paper, we propose a novel family of RNN units which are particularly useful for the contextual sequence mapping problem: the CARNNs. Together with our neural net architectures, CARNN-based systems outperform previous methods on several public datasets for dialog (Frame and Babi Task 6), question answering (TrecQA) and contextual language modelling (Switchboard and Penn Discourse Tree Bank). In the future, we plan to investigate the effectiveness of CARNN units in other sequence modelling tasks.

Question: **During what war did Nimitz serve?**

IWAN-LSTM	IWAN-iCARNN
Since the museum opened in 1983, Fredericksburg has become a haven for retired military servicemen who come to trace Nimitz's career and the events of World War II.	Since the museum opened in 1983, Fredericksburg has become a haven for retired military servicemen who come to trace Nimitz's career and the events of World War II.
Bill McCain, who graduated from West Point, chased Pancho Villa with Gen. Blackjack Pershing, served as an artillery officer during World War I and attained the rank of brigadier general.	Indeed, the ancestors of Chester W. Nimitz, the U.S. naval commander in chief of the Pacific in World War II, were among the first German pioneers to settle the area.
There was his grandfather, Admiral John "Slew" McCain, Class of 1906, a grizzled old sea dog who commanded aircraft carriers in the Pacific during World War II.	Slew McCain's peers at the Naval Academy were Chester Nimitz and William "Bull" Halsey, who would become major commanders during World War II.

Table 4: Top three answers produced by CARNN and LSTM. Blue colored answers are correct and red ones are incorrect.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Layla El Asri, Hannes Schulz, Shikhar Sharma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra, and Kaheer Suleman. 2017. Frames: A corpus for adding memory to goal-oriented dialogue systems. *arXiv preprint arXiv:1704.00057* .

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

Weijie Bian, Si Li, Zhao Yang, Guang Chen, and Zhiqing Lin. 2017. A compare-aggregate model with dynamic-clip attention for answer selection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*. pages 1987–1990.

Antoine Bordes and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. In *ICLR 2017*.

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

Hua He, Kevin Gimpel, and Jimmy J Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *EMNLP*. pages 1576–1586.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 770–778.

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

Yangfeng Ji, Trevor Cohn, Lingpeng Kong, Chris Dyer, and Jacob Eisenstein. 2016a. Document context language models. In *ICLR (Workshop track)*.

Yangfeng Ji, Gholamreza Haffari, and Jacob Eisenstein. 2016b. A latent variable recurrent neural network for discourse-driven language models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 332–342.

Daniel Jurafsky, Elizabeth Shriberg, and Debra Biasca. 1997. Switchboard SWBD-DAMSL Shallow-Discourse-Function Annotation Coders Manual, Draft 13. Technical report, University of Colorado.

Jey Han Lau, Timothy Baldwin, and Trevor Cohn. 2017. Topically driven neural language model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 355–365.

Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2017. Recurrent additive networks. *arXiv preprint arXiv:1705.07393* .

Bing Liu and Ian Lane. 2017. An end-to-end trainable neural network model with belief tracking for task-oriented dialog. In *Interspeech 2017*.

Fei Liu and Julien Perez. 2017. Gated end-to-end memory networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, Valencia, Spain, pages 1–10.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Eleni Miltsakaki, Rashmi Prasad, Aravind K Joshi, and Bonnie L Webber. 2004. The penn discourse treebank. In *LREC*.

Jinfeng Rao, Hua He, and Jimmy Lin. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, pages 1913–1916.

Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Query-regression networks for machine comprehension. In *ICLR*.

Gehui Shen, Yunlun Yang, and Zhi-Hong Deng. 2017. Inter-weighted alignment network for sentence pair modeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 1190–1200.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*. pages 2440–2448.

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2017. Enabling efficient question answer retrieval via hyperbolic neural networks. *CoRR* abs/1707.07847.

Quan Tran, Andrew MacKinlay, and Antonio Jimeno Yepes. 2017. Named entity recognition with stack residual lstm and trainable bias decoding. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. volume 1, pages 566–575.

Quan Hung Tran, Ingrid Zukerman, and Gholamreza Haffari. 2016. Inter-document contextual language model. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 762–766.

Mengqiu Wang, Noah A Smith, and Teruko Mitamura. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. In *EMNLP-CoNLL*. volume 7, pages 22–32.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814* .

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 665–677.

Combining Character and Word Information in Neural Machine Translation Using a Multi-Level Attention

Huadong Chen[†], **Shujian Huang**[†]*, **David Chiang**[‡], **Xinyu Dai**[†], **Jiajun Chen**[†]
[†]State Key Laboratory for Novel Software Technology, Nanjing University
{chenhd,huangsj,daixinyu,chenjj}@nlp.nju.edu.cn
[‡]Department of Computer Science and Engineering, University of Notre Dame
dchiang@nd.edu

Abstract

Natural language sentences, being hierarchical, can be represented at different levels of granularity, like words, subwords, or characters. But most neural machine translation systems require the sentence to be represented as a sequence at a single level of granularity. It can be difficult to determine which granularity is better for a particular translation task. In this paper, we improve the model by incorporating multiple levels of granularity. Specifically, we propose (1) an encoder with character attention which augments the (sub)word-level representation with character-level information; (2) a decoder with multiple attentions that enable the representations from different levels of granularity to control the translation cooperatively. Experiments on three translation tasks demonstrate that our proposed models outperform the standard word-based model, the subword-based model and a strong character-based model.

1 Introduction

Neural machine translation (NMT) models (Britz et al., 2017) learn to map from source language sentences to target language sentences via continuous-space intermediate representations. Since word is usually thought of as the basic unit of language communication (Jackendoff, 1992), early NMT systems built these representations starting from the word level (Sutskever et al., 2014; Bahdanau et al., 2015; Cho et al., 2014; Weng et al., 2017). Later systems tried using smaller units such as subwords to address the problem of out-of-vocabulary (OOV) words (Sennrich et al., 2016; Wu et al., 2016).

Although they obtain reasonable results, these word or sub-word methods still have some potential weaknesses. First, the learned representations of (sub)words are based purely on their contexts, but the potentially rich information inside the unit itself is seldom explored. Taking the Chinese word 被打伤 (*bei-da-shang*) as an example, the three characters in this word are a passive voice marker, "hit" and "wound", respectively. The meaning of the whole word, "to be wounded", is fairly compositional. But this compositionality is ignored if the whole word is treated as a single unit.

Secondly, obtaining the word or sub-word boundaries can be non-trivial. For languages like Chinese and Japanese, a word segmentation step is needed, which must usually be trained on labeled data. For languages like English and German, word boundaries are easy to detect, but subword boundaries need to be learned by methods like BPE. In both cases, the segmentation model is trained only in monolingual data, which may result in units that are not suitable for translation.

On the other hand, there have been multiple efforts to build models operating purely at the character level (Ling et al., 2015a; Yang et al., 2016; Lee et al., 2017). But splitting this finely can increase potential ambiguities. For example, the Chinese word 红茶 (*hong-cha*) means "black tea," but the two characters means "red" and "tea," respectively. It shows that modeling the character sequence alone may not be able to fully utilize the information at the word or sub-word level, which may also lead to an inaccurate representation. A further problem is that character sequences are longer, making them more costly to process with a recurrent neural network model (RNN).

While both word-level and character-level information can be helpful for generating better representations, current research which tries to exploit both word-level and character-level information only composed the word-level representation by character embeddings with the word boundary information (Ling et al., 2015b; Costa-jussà and

* Corresponding author.

Proceedings of NAACL-HLT 2018, pages 1284–1293
New Orleans, Louisiana, June 1 - 6, 2018.

Fonollosa, 2016) or replaces the word representation with its inside characters when encountering the out-of-vocabulary words (Luong and Manning, 2016; Wu et al., 2016). In this paper, we propose a novel encoder-decoder model that makes use of both character and word information. More specifically, we augment the standard encoder to attend to individual characters to generate better source word representations (§3.1). We also augment the decoder with a second attention that attends to the source-side characters to generate better translations (§3.2).

To demonstrate the effectiveness of the proposed model, we carry out experiments on three translation tasks: Chinese-English, English-Chinese and English-German. Our experiments show that: (1) the encoder with character attention achieves significant improvements over the standard word-based attention-based NMT system and a strong character-based NMT system; (2) incorporating source character information into the decoder by our multi-scale attention mechanism yields a further improvement, and (3) our modifications also improve a subword-based NMT model. To the best of our knowledge, this is the first work that uses the source-side character information for all the (sub)words in the sentence to enhance a (sub)word-based NMT model in both the encoder and decoder.

2 Neural Machine Translation

Most NMT systems follow the encoder-decoder framework with attention mechanism proposed by Bahdanau et al. (2015). Given a source sentence $\mathbf{x} = x_1 \cdots x_l \cdots x_L$ and a target sentence $\mathbf{y} = y_1 \cdots y_j \cdots y_J$, we aim to directly model the translation probability:

$$P(\mathbf{y} \mid \mathbf{x}; \theta) = \prod_{1}^{J} P(y_j \mid \mathbf{y}_{<j}, \mathbf{x}; \theta),$$

where θ is a set of parameters and $\mathbf{y}_{<j}$ is the sequence of previously generated target words. Here, we briefly describe the underlying framework of the encoder-decoder NMT system.

2.1 Encoder

Following Bahdanau et al. (2015), we use a bidirectional RNN with gated recurrent units (GRUs) (Cho et al., 2014) to encode the source sentence:

$$\begin{aligned} \overrightarrow{h_l} &= \mathrm{GRU}(\overrightarrow{h_{l-1}}, s_l; \overrightarrow{\theta}) \\ \overleftarrow{h_l} &= \mathrm{GRU}(\overleftarrow{h_{l-1}}, s_l; \overleftarrow{\theta}) \end{aligned} \tag{1}$$

where s_l is the l-th source word's embedding, GRU is a gated recurrent unit, $\overrightarrow{\theta}$ and $\overleftarrow{\theta}$ are the parameters of forward and backward GRU, respectively; see Cho et al. (2014) for a definition.

The *annotation* of each source word x_l is obtained by concatenating the forward and backward hidden states:

$$\overleftrightarrow{h_l} = \begin{bmatrix} \overrightarrow{h_l} \\ \overleftarrow{h_l} \end{bmatrix}.$$

The whole sequence of these annotations is used by the decoder.

2.2 Decoder

The decoder is a forward RNN with GRUs predicting the translation $\mathbf{y}$ word by word. The probability of generating the j-th word y_j is:

$$P(y_j \mid \mathbf{y}_{<j}, \mathbf{x}; \theta) = \mathrm{softmax}(\begin{bmatrix} t_{j-1} \\ d_j \\ c_j \end{bmatrix})$$

where t_{j-1} is the word embedding of the $(j-1)$-th target word, d_j is the decoder's hidden state of time j, and c_j is the *context vector* at time j. The state d_j is computed as

$$d_j = \mathrm{GRU}\left(d_{j-1}, \begin{bmatrix} t_{j-1} \\ c_j \end{bmatrix}; \theta_d\right).$$

The attention mechanism computes the context vector c_i as a weighted sum of the source annotations,

$$c_j = \sum_{i=1}^{I} \alpha_{j,l} \overleftrightarrow{h_l} \tag{2}$$

where the attention weight α_{ji} is

$$\alpha_{ji} = \frac{\exp(e_{ji})}{\sum_{i'=1}^{I} \exp(e_{ji'})} \tag{3}$$

and

$$e_{jl} = v_a^{\mathrm{T}} \tanh(W_a d_{j-1} + U_a \overleftrightarrow{h_l}) \tag{4}$$

where v_a, W_a and U_a are the weight matrices of the attention model, and e_{jl} is an attention model that scores how well d_{j-1} and $\overleftrightarrow{h_l}$ match.

With this strategy, the decoder can attend to the source annotations that are most relevant at a given time.

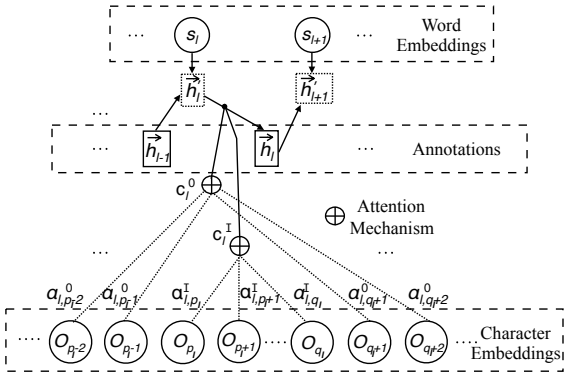


Figure 1: Forward encoder with character attention at time step l. The encoder alternates between reading word embeddings and character context vectors. ${c_l}^I$ and ${c_l}^O$ denotes the inside and outside character-level context vectors of the l-th word, respectively.

3 Character Enhanced Neural Machine Translation

In this section, we present models which make use of both character-level and word-level information in the encoder-decoder framework.

3.1 Encoder with Character Attention

The encoder maps the source sentence to a sequence of representations, which is then used by the attention mechanism. The standard encoder operates purely on (sub)words or characters. However, we want to encode both, since both levels can be linguistically significant (Xiong et al., 2017).

To incorporate multiple levels of granularity, we extend the encoder with two character-level attentions. For each source word, the characters of the whole sentence can be divided into two parts, those inside the word and those outside the word. The inside characters contain information about the internal structure of the word. The outside characters may provide information about patterns that cross word boundaries. In order to distinguish the influence of the two, we use two separate attentions, one for inside characters and one for outside characters.

Note that we compute attention directly from the character embedding sequence instead of using an additional RNN layer. This helps to avoid the vanishing gradient problem that would arise from increasing the sequence length, and also keeps the computation cost at a low level.

Figure 1 illustrates the forward encoder with character attentions. We write the character embeddings as $\mathbf{o} = o_1 \cdots o_k \cdots o_K$. Let p_l and q_l be the starting and ending character position, respectively, of word x_l. Then $o_{p_l} \cdots o_{q_l}$ are the inside characters of word x_l; $o_1 \cdots o_{p_l-1}$ and $o_{q_l+1} \cdots o_K$ are the outside characters of word x_l.

The encoder is an RNN that alternates between reading (sub)word embeddings and character-level information. At each time step, we first read the word embedding:

$$\overrightarrow{h_l}' = \mathrm{GRU}(\overrightarrow{h_{l-1}}, s_l; \overrightarrow{\theta'}) \tag{5}$$

Then we use the attention mechanisms to compute character context vectors for the inside characters:

$$c_l^I = \sum_{m=p_l}^{q_l} \alpha_{lm}^I o_m$$

$$\alpha_{lm}^I = \frac{\exp(e_{lm})}{\sum_{m'=p_l}^{q_l} \exp(e_{lm'})}$$

$$e_{lm} = v^I \cdot \tanh(W^I \overrightarrow{h_l}' + U^I o_m).$$

The outside character context vector ${c_l}^O$ is calculated in a similar way, using a different set of parameters, i.e. W^O, U^O, v^O instead of W^I, U^I, v^I.

The inside and outside character context vectors are combined by a feed-forward layer and fed into the encoder RNN, forming the character-enhanced word representation $\overrightarrow{h_l}$:

$$c_l^C = \tanh(W^I c_l^I + W^O {c_l}^O)$$

$$\overrightarrow{h_l} = \mathrm{GRU}(\overrightarrow{h_l}', c_l^C; \overrightarrow{\theta})$$

Note that this GRU does not share parameters with the GRU in (5).

The backward hidden states are calculated in a similar manner.

3.2 Decoder with Multi-Scale Attention

In order to fully exploit the character-level information, we also make extensions to the decoder, so that the character-level information can be taken into account while generating the translation.

We propose a multi-scale attention mechanism to get the relative information of current decoding step from both word-level and character-level representations. This attention mechanism is build from the high-level to the low-level representation, in order to enhance high-level representation with fine-grained internal structure and context. The multi-scale attention mechanism is built (as shown in Figure 2) from word-level to character-level.

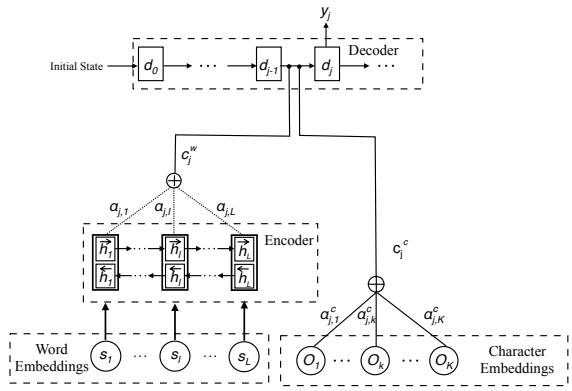


Figure 2: Illustration of the decoder with our multi-scale attention mechanism.

First, we get the word-level information. The context vector c_j^w is calculated following the standard attention model (Eq. 2–4). And the hidden state $\tilde{d}_j$ is updated.

$$\tilde{d}_j = \text{GRU}\left(d_{j-1}, \begin{bmatrix} t_{j-1} \\ c_j^w \end{bmatrix}; \tilde{\theta}_d\right), \tag{6}$$

Then we attend to the character-level representation, which provides more information about the word's internal structure. The context vector c_j^c is calculated based on the updated hidden state above,

$$\begin{aligned} c_j^c &= \sum_{k=1}^{K} \alpha_{jk}^c o_k \\ \alpha_{jk}^c &= \frac{\exp(e_{jk})}{\sum_{k'=1}^{K} \exp(e_{jk'})} \\ e_{j,k} &= v^c \cdot \tanh(W^c \tilde{d}_j + U^c o_k). \end{aligned}$$

Finally, the word-level context vector $c_j^w$and character-level context vector c_j^c are concatenated:

$$c_j = \begin{bmatrix} c_j^w \\ c_j^c \end{bmatrix}.$$

And the final context vector c_j is used to help predict the next target word.

$$P(y_j \mid \mathbf{y}_{<j}, \mathbf{x}; \theta) = \text{softmax}(\begin{bmatrix} t_{j-1} \\ d_j \\ c_j \end{bmatrix})$$

where d_j is

$$d_j = \text{GRU}(\tilde{d}_j, c_j^c; \theta_d),$$

With this mechanism, both the (sub)word-level and character-level representations could be used to predict the next translation, which helps to ensure a more robust and reasonable choice. It may also help to alleviate the under-translation problem because the character information could be a complement to the word.

4 Experiments

We conduct experiments on three translation tasks: Chinese-English (Zh-En), English-Chinese (En-Zh) and English-German (En-De). We write Zh↔En to refer to the Zh-En and En-Zh tasks together.

4.1 Datasets

For Zh↔En, the parallel training data consists of 1.6M sentence pairs extracted from LDC corpora, with 46.6M Chinese words and 52.5M English words, respectively.[1] We use the NIST MT02 evaluation data as development data, and MT03, MT04, MT05, and MT06 as test data. The Chinese side of the corpora is word segmented using ICTCLAS.[2] The English side of the corpora is lowercased and tokenized.

For En-De, we conduct our experiments on the WMT17 corpus. We use the pre-processed parallel training data for the shared news translation task provided by the task organizers.[3] The dataset consitst of 5.6M sentence pairs. We use `newstest2016` as the development set and evaluate the models on `newstest2017`.

4.2 Baselines

We compare our proposed models with several types of NMT systems:

- **NMT:** the standard attentional NMT model with words as its input (Bahdanau et al., 2015).
- **RNN-Char:** the standard attentional NMT model with characters as its input.
- **CNN-Char:** a character-based NMT model, which implements the convolutional neural network (CNN) based encoder (Costa-jussà and Fonollosa, 2016).
- **Hybrid:** the mixed word/character model proposed by Wu et al. (2016).

[1] LDC2002E18, LDC2003E14, the Hansards portion of LDC2004T08, and LDC2005T06.

[2] `http://ictclas.nlpir.org`

[3] `http://data.statmt.org/wmt17/translation-task/preprocessed/de-en/`

System	MT02	MT03	MT04	MT05	MT06	Mean	Δ
NMT	33.76	31.88	33.15	30.55	27.47	30.76	
Word-att	34.28	32.26	33.82	31.02	27.93	31.26	+0.50
Char-att	34.85	33.71	34.91	32.08	28.66	32.34	+1.58

Table 1: Performance of the encoder with character attention and the encoder with word attention. Char-att and Word-att denotes the encoder with character attention and the encoder with word attention, respectively.

- **BPE:** a subword level NMT model, which processes the source side sentence by Byte Pair Encoding (BPE) (Sennrich et al., 2016).

We used the dl4mt implementation of the attentional model,[4] reimplementing the above models.

4.3 Details

Training For Zh↔En, we filter out the sentence pairs whose source or target side contain more than 50 words. We use a shortlist of the 30,000 most frequent words in each language to train our models, covering approximately 98.2% and 99.5% of the Chinese and English tokens, respectively. The word embedding dimension is 512. The hidden layer sizes of both forward and backward sequential encoder are 1024. For fair comparison, we also set the character embedding size to 512, except for the CNN-Char system. For CNN-Char, we follow the standard setting of the original paper (Costa-jussà and Fonollosa, 2016).

For En-De, we build the baseline system using joint BPE segmentation (Sennrich et al., 2017). The number of joint BPE operations is 90,000. We use the total BPE vocabulary for each side.

We use Adadelta (Zeiler, 2012) for optimization with a mini-batch size of 32 for Zh↔En and 50 for En-De.

Decoding and evaluation We use beam search with length-normalization to approximately find the most likely translation. We set beam width to 5 for Zh↔En and 12 for En-De. The translations are evaluated by BLEU (Papineni et al., 2002). We use the `multi-bleu` script for Zh↔En,[5] and the `multi-bleu-detok` script for En-De.[6]

[4] https://github.com/nyu-dl/dl4mt-tutorial

[5] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl

[6] https://github.com/EdinburghNLP/nematus/blob/master/data/multi-bleu-detok.perl

4.4 Results: Encoder with character attention

This set of experiments evaluates the effectiveness of our proposed character enhanced encoder. In Table 1, we first compare the encoder with character attention (Char-att) with the baseline word-based model. The result shows that our extension of the encoder can obtain significantly better performance (+1.58 BLEU).

Then, in order to investigate whether the improvement comes from the extra parameters in the character layer, we compare our model to a word embedding enhanced encoder. When the word embedding enhanced encoder encodes a word, it attends to the word's embedding and other word embedding in the sentence instead of attending to the word's inside and outside character embeddings. The results show that the word embedding enhanced encoder (Word-att) only gets a 0.5 BLEU improvement than the baseline, while our model is significantly better (+1.58 BLEU). This shows that the benefit comes from the augmented character-level information which help the word-based encoder to learn a better source-side representation.

Finally, we compare our character enhanced model with several types of systems including a strong character-based model proposed by Costa-jussà and Fonollosa (2016) and a mixed word/character model proposed by Wu et al. (2016). In Table 2, rows 2 and 2′ confirm the finding of Yang et al. (2016) that the traditional RNN model performs less well when the input is a sequence of characters. Rows 4 and 4′ indicate that Wu et al. (2016)'s scheme to combine of words and characters is effective for machine translation. Our model (row 5) outperforms other models on the Zh-En task, but only outperforms the word-based model on En-Zh. The results may suggest that the CNN and RNN methods is also strong in building the source representation.

Task	#	System	MT02	MT03	MT04	MT05	MT06	Mean	Δ
Zh-En	1	NMT	33.76	31.88	33.15	30.55	27.47	30.76	
	2	RNN-Char	32.22	31.05	31.41	28.85	25.99	29.32	−1.44
	3	CNN-Char	33.69	32.06	33.10	30.40	27.67	30.81	+0.05
	4	Hybrid	34.33	33.10	33.41	30.96	28.00	31.37	+0.60
	5	Char-att	34.85	33.71	34.91	32.08	28.66	32.34	+1.58
	6	Multi-att	34.61	33.26	34.42	31.06	28.24	31.75	+0.98
	7	5+6	35.42	33.9	35.23	32.62	29.36	32.68	+2.02
En-Zh	1′	NMT	31.58	22.20	23.47	22.50	21.47	22.41	
	2′	RNN-Char	28.78	21.03	21.70	19.81	20.98	20.88	−1.53
	3′	CNN-Char	31.36	23.60	24.71	22.75	23.05	23.53	+1.12
	4′	Hybrid	31.31	24.45	24.65	23.10	23.62	23.96	+1.55
	5′	Char-att	30.93	23.63	24.42	21.92	23.6	23.39	+0.98
	6′	Multi-att	30.17	22.09	24.09	22.29	23.8	23.07	+0.66
	7′	5′+6′	32.91	25.02	25.69	24.03	25.20	24.99	+2.58

Table 2: Performance of different systems on the Chinese-English and English-Chinese translation tasks. Our encoder with character attention (Char-att) improves over all other models on Zh-En and over the word-based baseline on En-Zh. Adding our decoder with multi-scale attention (Multi-att) outperforms all other models.

Task	#	System	MT02	MT03	MT04	MT05	MT06	Mean	Δ
Zh-En	8	BPE	34.66	33.65	34.69	30.80	27.66	31.70	+0.94
	9	Char-att	35.20	34.93	35.39	31.62	28.56	32.63	+1.86
	10	9+Multi-att	36.68	35.39	35.93	32.08	29.74	33.29	+2.52
En-Zh	8′	BPE	30.17	22.09	24.09	22.29	23.80	23.07	+0.66
	9′	Char-att	30.95	23.07	25.19	22.74	24.27	23.82	+1.41
	10′	9′+Multi-att	32.36	24.91	25.79	23.42	24.88	24.75	+2.34

Table 3: Comparison of our models on top of the BPE-based NMT model and the original BPE-based model on the Chinese-English and English-Chinese translation tasks. Our models improve over the BPE baselines.

4.5 Results: Multi-scale attention

Rows 6 and 6′ in Table 2 verify that our multi-scale attention mechanism can obtain better results than baseline systems. Rows 7 and 7′ in Table 2 show that our proposed multi-scale attention mechanism further improves the performance of our encoder with character attention, yielding a significant improvement over the standard word-based model on both Zh-En (+2.02 vs. row 1) task and En-Zh translation task (+2.58 vs. row 1′).

Compared to the CNN-Char model, our model still gets +1.97 and +1.46 BLEU improvement on Zh-En and En-Zh, respectively. Compared to the mixed word/character model proposed by (Wu et al., 2016), we find that our best model gives a better result, demonstrating the benefits of exploiting the character level information during decoding.

System	Dev	Test	Δ
BPE	28.41	23.05	
Char-att	29.80	23.87	+0.82
+Multi-att	30.52	24.48	+1.43

Table 4: Case-sensitive BLEU on the English-German translation tasks. Our systems improve over a baseline BPE system.

4.6 Results: Subword-based models

Currently, subword-level NMT models are widely used for achieving open-vocabulary translation. Sennrich et al. (2016) introduced a subword-level NMT model using subword-level segmentation based on the byte pair encoding (BPE) algorithm. In this section, we investigate the effectiveness of our character enhanced model on top of the BPE model. Table 3 shows the results on the Zh-En task

(a) OOV words

Source	互联网 业务 仍 将 是 中国 **通信业** 增长 速度 最 快 的 业务 [...]
Reference	internet will remain the business with the fastest growth in china 's *telecommunication industry* [...]
NMT	internet business will remain the fastest growing business in china . [...]
Hybrid	the internet will remain the fastest of china 's *communications* growth speed [...]
Ours	internet business will continue to be the fastest growing business of china 's *telecommunications industry* [...]

(b) Frequent words

Source	[...] 不 是 目前 在 巴勒斯坦 **被占领土** 所 发生 的 事件 的 根源 [...]
Reference	[...] not the source of what happened on the palestinian *territory occupied* by israel [...]
NMT	[...] such actions were not the source of the incidents of the current palestinian *occupation* [...]
CNN-Char	[...] not the only source of events that took place in the palestinian *occupation* [...]
Ours	[...] not the root of the current incidents that took place in the palestinian *occupied territories* [...]
Source	[...] 将 **东西方 冷战** 的 象征 柏林 墙 的 三 块 墙体 赠送 到 了 联合国
Reference	[...] presented the united nations with three pieces of the berlin wall , a symbol of *the cold war between the east and the west* .
NMT	[...] sent the three pieces of UNK UNK to the un to the un
CNN-Char	[...] sent three pieces of UNK to the united nations, which was *the cold war in eastern china* .
Ours	[...] presented the un on the 4th of the three wall UNK of *the eastern and western cold war* .

Table 5: Sample translations. For each example, we show the source, the reference and the translation from our best model. "Ours" means our model with both Char-att and Multi-att.

and En-Zh translation task. Rows 8 and 8′ confirm that BPE slightly improves the performance of the word-based model. But both our character enhanced encoder and the multi-scale attention yield better results. Our best model leads to improvements of up to 1.58 BLEU and 1.68 BLEU on the Zh-En task and En-Zh translation task, respectively.

We also conduct experiments on the En-De translation task (as shown in Table 4). The result is consistent with Zh-En task and En-Zh translation tasks. Our best model obtains 1.43 BLEU improvement over the BPE model.

System	**with OOV**	**Δ**	**no OOV**	**Δ**
NMT	28.47		38.21	
Hybrid	29.83	+1.36	37.79	−0.43
Ours	30.80	+2.33	39.39	+1.18

Table 6: Translation performance on source sentences with and without OOV words. "Ours" means our model with both Char-att and Multi-att.

4.7 Analysis

We have argued that the character information is important not only for OOV words but also frequent words. To test this claim, we divided the MT03 test set into two parts according to whether

the sentence contains OOV words, and evaluated several systems on the two parts. Table 6 lists the results. Although the hybrid model achieves a better result on the sentences which contain OOV words, it actually gives a worse result on the sentences without OOV words. By contrast, our model yields the best results on both parts of the data. This shows that frequent words also benefit from fine-grained character-level information.

Table 5 shows three translation examples. Table 5(a) shows the translation of an OOV word 通信业 (*tong-xin-ye*, telecommunication industry). The baseline NMT system can't translate the whole word because it is not in the word vocabulary. The hybrid model translates the word to "communication," which is a valid translation of the first two characters 通信. This mistranslation also appears to affect other parts of the sentence adversely. Our model translates the OOV word correctly.

Table 5(b) shows two translation samples involving frequent words. For the compound word 被占领土 (*beizhanlingtu*, occupied territory), the baseline NMT system only partly translates the word as "occupation" and ignores the main part 领土 (*lingtu*, territory). The CNN-Char model, which builds up the word-level representation from characters, also cannot capture 领土 (*lingtu*). However, our model correctly translates the word as "occupied territories." (The phrase "by Israel" in the reference was inserted by the translator.) The word 东西方 (*dongxifang*, east and west) and 冷战 (*lengzhan*, cold war) are deleted by the baseline model, and even the CNN-Char model translates 东西方 (*dongxifang*) incorrectly. By contrast, our model can make use of both words and characters to translate the word 东西方 (*dongxifang*) reasonably well as "eastern and western."

5 Related Work

Many recent studies have focused on using character-level information in neural machine translation systems. These efforts could be roughly divided into the following two categories.

The first line of research attempted to build neural machine translation models purely on characters without explicit segmentation. Lee et al. (2017) proposed to directly learn the segmentation from characters by using convolution and pooling layers. Yang et al. (2016) composed the high-level representation by the character embedding and its surrounding character-level context with a bidirectional and concatenated row convolution network. Different from their models, our model aims to use characters to enhance words representation instead of depending on characters solely; our model is also much simpler.

The other line of research attempted to combine character-level information with word-level information in neural machine translation models, which is more similar with our work. Ling et al. (2015a) employed a bidirectional LSTM to compose character embeddings to form the word-level information with the help of word boundary information. Costa-jussà and Fonollosa (2016) replaced the word-lookup table with a convolutional network followed by a highway network (Srivastava et al., 2015), which learned the word-level representation by its constituent characters. Zhao and Zhang (2016) designed a *decimator* for their encoder, which effectively uses a RNN to compute a word representation from the characters of the word. These approaches only consider word boundary information and ignore the word-level meaning information itself. In contrast, our model can make use of both character-level and word-level information.

Luong and Manning (2016) proposed a hybrid scheme that consults character-level information whenever the model encounters an OOV word. Wu et al. (2016) converted the OOV words in the word-based model into the sequence of its constituent characters. These methods only focus on dealing with OOV words by augmenting the character-level information. In our work, we augment the character information to all the words.

6 Conclusion

In this paper, we have investigated the potential of using character-level information in word-based and subword-based NMT models by proposing a novel character-aware encoder-decoder framework. First, we extended the encoder with a character attention mechanism for learning better source-side representations. Then, we incorporated information about source-side characters into the decoder with a multi-scale attention, so that the character-level information can cooperate with the word-level information to better control the translation. The experiments have demonstrated the effectiveness of our models. Our analysis showed that both OOV words and frequent

words benefit from the character-level information.

Our current work only uses the character-level information in the source-side. For future work, it will be interesting to make use of finer-grained information on the target side as well.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments. This work is supported by the National Science Foundation of China (No. 61772261 and 61672277) and the Jiangsu Provincial Research Foundation for Basic Research (No. BK20170074). Part of Huadong Chen's contribution was made while visiting University of Notre Dame. His visit was supported by the joint PhD program of China Scholarship Council.

References

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

Denny Britz, Quoc Le, and Reid Pryzant. 2017. Effective domain mixing for neural machine translation.

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

Marta R. Costa-jussà and José A. R. Fonollosa. 2016. Character-based neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 357–361, Berlin, Germany. Association for Computational Linguistics.

Ray S. Jackendoff. 1992. *Semantic Structures*. MIT Press.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, 5:365–378.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W. Black. 2015a. Character-based neural machine translation. *CoRR*, abs/1511.04586.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015b. Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*.

Minh-Thang Luong and D. Christopher Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1054–1063. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. ACL*, pages 311–318.

Rico Sennrich, Alexandra Birch, Anna Currey, Ulrich Germann, Barry Haddow, Kenneth Heafield, Antonio Valerio Miceli Barone, and Philip Williams. 2017. The university of edinburgh's neural mt systems for wmt17.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pages 2377–2385, Cambridge, MA, USA. MIT Press.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112.

Rongxiang Weng, Shujian Huang, Zaixiang Zheng, XIN-YU DAI, and Jiajun CHEN. 2017. Neural machine translation with word predictions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 136–145, Copenhagen, Denmark. Association for Computational Linguistics.

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

H. Xiong, Z. He, X. Hu, and H. Wu. 2017. Multi-channel Encoder for Neural Machine Translation. arXiv:1712.02109.

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2016. A character-aware encoder for neural machine translation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3063–3070, Osaka, Japan. The COLING 2016 Organizing Committee.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

Shenjian Zhao and Zhihua Zhang. 2016. An efficient character-level neural machine translation. *CoRR*, abs/1608.04738.

Dense Information Flow for Neural Machine Translation

Yanyao Shen[1], Xu Tan[2], Di He[3], Tao Qin[2], and Tie-Yan Liu[2]

[1]University of Texas at Austin
[2]Microsoft Research, Asia
[3]Key Laboratory of Machine Perception, MOE, School of EECS, Peking University
shenyanyao@utexas.edu, {xuta,taoqin,tie-yan.liu}@microsoft.com,di_he@pku.edu.cn

Abstract

Recently, neural machine translation has achieved remarkable progress by introducing well-designed deep neural networks into its encoder-decoder framework. From the optimization perspective, residual connections are adopted to improve learning performance for both encoder and decoder in most of these deep architectures, and advanced attention connections are applied as well. Inspired by the success of the DenseNet model in computer vision problems, in this paper, we propose a densely connected NMT architecture (DenseNMT) that is able to train more efficiently for NMT. The proposed DenseNMT not only allows dense connection in creating new features for both encoder and decoder, but also uses the dense attention structure to improve attention quality. Our experiments on multiple datasets show that DenseNMT structure is more competitive and efficient.

1 Introduction

Neural machine translation (NMT) is a challenging task that attracts lots of attention in recent years. Starting from the encoder-decoder framework (Cho et al., 2014), NMT starts to show promising results in many language pairs. The evolving structures of NMT models in recent years have made them achieve higher scores and become more favorable. The attention mechanism (Bahdanau et al., 2015) added on top of encoder-decoder framework is shown to be very useful to automatically find alignment structure, and single-layer RNN-based structure has evolved into deeper models with more efficient transformation functions (Gehring et al., 2017; Kaiser et al., 2017; Vaswani et al., 2017).

One major challenge of NMT is that its models are hard to train in general due to the complexity of both the deep models and languages. From the optimization perspective, deeper models are hard to efficiently back-propagate the gradients, and this phenomenon as well as its solution is better explored in the computer vision society. Residual networks (ResNet) (He et al., 2016) achieve great performance in a wide range of tasks, including image classification and image segmentation. Residual connections allow features from previous layers to be accumulated to the next layer easily, and make the optimization of the model efficiently focus on refining upper layer features.

NMT is considered as a challenging problem due to its sequence-to-sequence generation framework, and the goal of comprehension and reorganizing from one language to the other. Apart from the encoder block that works as a feature generator, the decoder network combining with the attention mechanism bring new challenges to the optimization of the models. While nowadays best-performing NMT systems use residual connections, we question whether this is the most efficient way to propagate information through deep models. In this paper, inspired by the idea of using dense connections for training computer vision tasks (Huang et al., 2016), we propose a densely connected NMT framework (DenseNMT) that efficiently propagates information from the encoder to the decoder through the attention component. Taking the CNN-based deep architecture as an example, we verify the efficiency of DenseNMT. Our contributions in this work include: (i) by comparing the loss curve, we show that DenseNMT allows the model to pass information more efficiently, and speeds up training; (ii) we show through ablation study that dense con-

Proceedings of NAACL-HLT 2018, pages 1294–1303
New Orleans, Louisiana, June 1 - 6, 2018.

nections in all three blocks altogether help improve the performance, while not increasing the number of parameters; (iii) DenseNMT allows the models to achieve similar performance with much smaller embedding size; (iv) DenseNMT on IWSLT14 German-English and Turkish-English translation tasks achieves new benchmark BLEU scores, and the result on WMT14 English-German task is more competitive than the residual connections based baseline model.

2 Related Work

ResNet and DenseNet. ResNet (He et al., 2016) proposes residual connections, which directly add representation from the previous layer to the next layer. Originally proposed for image classification tasks, the residual structure have proved its efficiency in model training across a wide range of tasks, and are widely adopted in recent advanced NMT models (Wu et al., 2016; Vaswani et al., 2017; Gehring et al., 2017). Following the idea of ResNet, DenseNet (Huang et al., 2016) further improves the structure and achieves state-of-the-art results. It allows the transformations (e.g., CNN) to be directly calculated over all previous layers. The benefit of DenseNet is to encourage upper layers to create new representations instead of refining the previous ones. On other tasks such as segmentation, dense connections also achieve high performance (Jégou et al., 2017). Very recently, (Godin et al., 2017) shows that dense connections help improve language modeling as well. Our work is the first to explore dense connections for NMT tasks.

Attention mechanisms in NMT. The attention block is proven to help improve inference quality due to existence of alignment information (Bahdanau et al., 2015). Traditional sequence-to-sequence architectures (Kalchbrenner and Blunsom, 2013; Cho et al., 2014) pass the last hidden state from the encoder to the decoder; hence source sentences of different length are encoded into a fixed-size vector (i.e., the last hidden state), and the decoder should catch all the information from the vector. Later, early attention-based NMT architectures, including (Bahdanau et al., 2015), pass all the hidden states (instead of the last state) of the last encoder layer to the decoder. The decoder then uses an attention mechanism to selectively focus on those hidden states while generating each word in the target sentence. Latest architecture (Gehring et al., 2017) uses multi-step attention, which allows each decoder layer to acquire separate attention representations, in order to maintain different levels of semantic meaning. They also enhance the performance by using embeddings of input sentences. In this work, we further allow every encoder layer to directly pass the information to the decoder side.

Encoder/decoder networks. RNNs such as long short term memory (LSTM) are widely used in NMT due to their ability of modeling long-term dependencies. Recently, other more efficient structures have been proposed in substitution for RNN-based structures, which includes convolution (Gehring et al., 2017; Kaiser et al., 2017) and self-attention (Vaswani et al., 2017). More specifically, ConvS2S (Gehring et al., 2017) uses convolution filter with a gated linear unit, Transformer (Vaswani et al., 2017) uses self-attention function before a two-layer position-wise feed-forward networks, and SliceNet (Kaiser et al., 2017) uses a combination of ReLU, depthwise separable convolution, and layer normalization. The advantage of these non-sequential transformations is the significant parallel speedup as well as more advanced performances, which is the reason we select CNN-based models for our experiments.

3 DenseNMT

In this section, we introduce our DenseNMT architecture. In general, compared with residual connected NMT models, DenseNMT allows each layer to provide its information to all subsequent layers directly. Figure 1-3 show the design of our model structure by parts.

We start with the formulation of a regular NMT model. Given a set of sentence pairs $S = \{(x^i, y^i)|i=1,\cdots,N\}$, an NMT model learns parameter θ by maximizing the log-likelihood function:

$$\sum_{i=1}^{N} \log \mathcal{P}(y^i|x^i;\theta). \quad (1)$$

For every sentence pair $(x, y) \in S$, $\mathcal{P}(y|x;\theta)$ is calculated based on the decomposition:

$$\mathcal{P}(y|x;\theta) = \prod_{j=1}^{m} \mathcal{P}(y_j|y_{<j}, x;\theta), \quad (2)$$

where m is the length of sentence y. Typically, NMT models use the encoder-attention-decoder

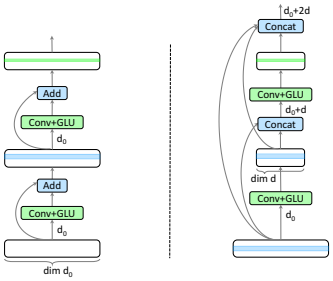


Figure 1: Comparison of dense-connected encoder and residual-connected encoder. Left: regular residual-connected encoder. Right: dense-connected encoder. Information is directly passed from blue blocks to the green block.

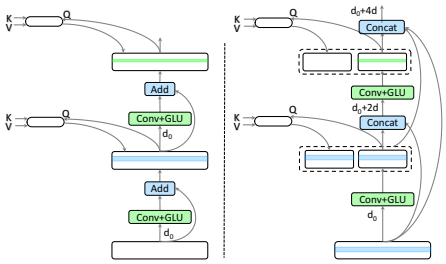


Figure 2: Comparison of dense-connected decoder and residual-connected decoder. Left: regular residual-connected decoder. Right: dense-connected decoder. Ellipsoid stands for attention block. Information is directly passed from blue blocks to the green block.

framework (Bahdanau et al., 2015), and potentially use multi-layer structure for both encoder and decoder. Given a source sentence x with length n, the encoder calculates hidden representations by layer. We denote the representation in the l-th layer as h^l, with dimension $n \times d^l$, where d^l is the dimension of features in layer l. The hidden representation at each position h^l_j is either calculated by:

$$h^l_j = \mathcal{H}^{\texttt{rec}}(h^{l-1}_j, h^l_{j-1}) \tag{3}$$

for recurrent transformation $\mathcal{H}^{\texttt{rec}}(\cdot)$ such as LSTM and GRU, or by:

$$h^l_j = \mathcal{H}^{\texttt{par}}(h^{l-1}) \tag{4}$$

for parallel transformation $\mathcal{H}^{\texttt{par}}(\cdot)$. On the other hand, the decoder layers $\{z^l\}$ follow similar structure, while getting extra representations from the encoder side. These extra representations are also called *attention*, and are especially useful for capturing alignment information.

In our experiments, we use convolution based transformation for $\mathcal{H}^{\texttt{par}}(\cdot)$ due to both its efficiency and high performance, more formally,

$$h^l_j = \texttt{GLU}([h^{l-1}_{j-r}, \cdots, h^{l-1}_{j+r}]W^l + b^l) \triangleq \mathcal{H}(h^{l-1}). \tag{5}$$

GLU is the gated linear unit proposed in (Dauphin et al., 2017) and the kernel size is $2r+1$. DenseNMT is agnostic to the transformation function, and we expect it to also work well combining with other transformations, such as LSTM, self-attention and depthwise separable convolution.

3.1 Dense encoder and decoder

Different from residual connections, later layers in the dense encoder are able to use features from all previous layers by concatenating them:

$$h^{l+1} = \mathcal{H}([h^l, h^{l-1}, \cdots, h^0]). \tag{6}$$

Here, $\mathcal{H}(\cdot)$ is defined in Eq. (5), $[\cdot]$ represents concatenation operation. Although this brings extra connections to the network, with smaller number of features per layer, the architecture encourages feature reuse, and can be more compact and expressive. As shown in Figure 1, when designing the model, the hidden size in each layer is much smaller than the hidden size of the corresponding layer in the residual-connected model.

While each encoder layer perceives information from its previous layers, each decoder layer z^{l+1} has two information sources: previous layers $z^i, i \leq l$, and attention values $a^i, i \leq l$. Therefore, in order to allow dense information flow, we redefine the generation of $(l+1)$-th layer as a nonlinear function over all its previous decoder layers and previous attentions. This can be written as:

$$z^{l+1} = \mathcal{H}([z^l, a^l, z^{l-1}, a^{l-1}, \cdots, z^1, a^1, z^0]), \tag{7}$$

where a^i is the attention value using i-th decoder layer and information from encoder side, which will be specified later. Figure 2 shows the comparison of a dense decoder with a regular residual decoder. The dimensions of both attention values and hidden layers are chosen with smaller values, yet the perceived information for each layer consists of a higher dimension vector with more representation power. The output of the decoder is a linear transformation of the concatenation of all layers by default. To compromise to the increment of dimensions, we use summary layers, which will be introduced in Section 3.3. With summary layers, the output of the decoder is only a linear transformation of the concatenation of the upper few layers.

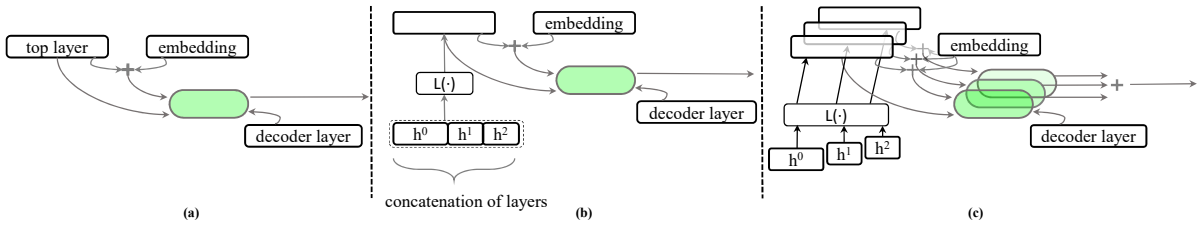


Figure 3: Illustration of DenseAtt mechanisms. For clarity, We only plot the attention block for a single decoder layer. (a): multi-step attention (Gehring et al., 2017), (b): DenseAtt-1, (c): DenseAtt-2. $\mathcal{L}(\cdot)$ is the linear projection function. The ellipsoid stands for the core attention operation as shown in Eq. (8).

3.2 Dense attention

Prior works show a trend of designing more expressive attention mechanisms (as discussed in Section 2). However, most of them only use the last encoder layer. In order to pass more abundant information from the encoder side to the decoder side, the attention block needs to be more expressive. Following the recent development of designing attention architectures, we propose DenseAtt as the dense attention block, which serves for the dense connection between the encoder and the decoder side. More specifically, two options are proposed accordingly. For each decoding step in the corresponding decoder layer, the two options both calculate attention using multiple encoder layers. The first option is more compressed, while the second option is more expressive and flexible. We name them as DenseAtt-1 and DenseAtt-2 respectively. Figure 3 shows the architecture of (a) multi-step attention (Gehring et al., 2017), (b) DenseAtt-1, and (c) DenseAtt-2 in order. In general, a popular multiplicative attention module can be written as:

$$\mathcal{F}(Q, K, V) = \texttt{Softmax}(Q \times K) \times V, \quad (8)$$

where Q, K, V represent query, key, value respectively. We will use this function $\mathcal{F}$ in the following descriptions.

DenseAtt-1 In the decoding phase, we use a layer-wise attention mechanism, such that each decoder layer absorbs different attention information to adjust its output. Instead of treating the last hidden layer as the encoder's output, we treat the concatenation of all hidden layers from encoder side as the output. The decoder layer multiplies with the encoder output to obtain the attention weights, which is then multiplied by a linear combination of the encoder output and the sentence embedding. The attention output of each layer a^l can be formally written as:

$$a^l = \mathcal{F}\left(\mathcal{L}(z^l), \mathcal{L}\left([\{h^i\}]\right), \mathcal{L}\left([\{h^i\}]\right) + \mathcal{L}(h^0)\right), \quad (9)$$

where $\mathcal{F}(\cdot,\cdot,\cdot)$ is the multiplicative attention function, $[\cdot]$ is a concatenation operation that combines all features, and $\mathcal{L}(\cdot)$ is a linear transformation function that maps each variable to a fixed dimension in order to calculate the attention value. Notice that we explicitly write the $\mathcal{L}(h^0)$ term in (9) to keep consistent with the multi-step attention mechanism, as pictorially shown in Figure 3(a).

DenseAtt-2 Notice that the transformation $\mathcal{L}([\{h^i\}])$ in DenseAtt-1 forces the encoder layers to be mixed before doing attention. Since we use multiple hidden layers from the encoder side to get an attention value, we can alternatively calculate multiple attention values before concatenating them. In another word, the decoder layer can get different attention values from different encoder layers. This can be formally expressed as:

$$a^l = \sum_{i=1}^{L} \mathcal{F}\left(\mathcal{L}(z^l), \mathcal{L}(h^i), \mathcal{L}([h^i, h^0])\right), \quad (10)$$

where the only difference from Eq. (9) is that the concatenation operation is substituted by a summation operation, and is put after the attention function $\mathcal{F}$. This method further increases the representation power in the attention block, while maintaining the same number of parameters in the model.

3.3 Summary layers

Since the number of features fed into nonlinear operation is accumulated along the path, the parameter size increases accordingly. For example, for the L-th encoder layer, the input dimension of features is $(L-1)d + d_0$, where d is the feature

dimension in previous layers, d_0 is the embedding size. In order to avoid the calculation bottleneck for later layers due to large L, we introduce the *summary layer* for deeper models. It summarizes the features for all previous layers and projects back to the embedding size, so that later layers of both the encoder and the decoder side do not need to look back further. The summary layers can be considered as contextualized word vectors in a given sentence (McCann et al., 2017). We add one summary layer after every $(\texttt{sumlen} - 1)$ layers, where `sumlen` is the hyperparameter we introduce. Accordingly, the input dimension of features is at most $(\texttt{sumlen} - 1) \cdot d + d_0$ for the last layer of the encoder. Moreover, combined with the summary layer setting, our DenseAtt mechanism allows each decoder layer to calculate the attention value focusing on the last few encoder layers, which consists of the last contextual embedding layer and several dense connected layers with low dimension. In practice, we set `sumlen` as 5 or 6.

3.4 Analysis of information flow

Figure 1 and Figure 2 show the difference of information flow compared with a residual-based encoder/decoder. For residual-based models, each layer can absorb a single high-dimensional vector from its previous layer as the only information, while for DenseNMT, each layer can utilize several low-dimensional vectors from its previous layers and a high-dimensional vector from the first layer (embedding layer) as its information. In DenseNMT, each layer directly provides information to its later layers. Therefore, the structure allows feature reuse, and encourages upper layers to focus on creating new features. Furthermore, the attention block allows the embedding vectors (as well as other hidden layers) to guide the decoder's generation more directly; therefore, during back-propagation, the gradient information can be passed directly to all encoder layers simultaneously.

4 Experimental Setup

4.1 Datasets

We use three datasets for our experiments: IWSLT14 German-English, Turkish-English, and WMT14 English-German.

We preprocess the IWSLT14 German-English dataset following byte-pair-encoding (BPE) method (Sennrich et al., 2015b)[1]. We learn 25k BPE codes using the joint corpus of source and target languages. We randomly select 7k from IWSLT14 German-English as the development set , and the test set is a concatenation of dev2010, tst2010, tst2011 and tst2012, which is widely used in prior works (Ranzato et al., 2015; Bahdanau et al., 2017; Huang et al., 2017).

For the Turkish-English translation task, we use the data provided by IWSLT14 (Cettolo et al., 2014) and the SETimes corpus (Cettolo et al., 2014) following (Sennrich et al., 2015a). After removing sentence pairs with length ratio over 9, we obtain 360k sentence pairs. Since there is little commonality between the two languages, we learn 30k size BPE codes separately for Turkish and English. In addition to this, we give another preprocessing for Turkish sentences and use word-level English corpus. For Turkish sentences, following (Gulcehre et al., 2015; Sennrich et al., 2015a), we use the morphology tool Zemberek with disambiguation by the morphological analysis (Sak et al., 2007) and removal of non-surface tokens[2]. Following (Sennrich et al., 2015a), we concatenate tst2011, tst2012, tst2013, tst2014 as our test set. We concatenate dev2010 and tst2010 as the development set.

We preprocess the WMT14 English-German[3] dataset using a BPE code size of 40k. We use the concatenation of newstest2013 and newstest2012 as the development set.

4.2 Model and architect design

As the baseline model (*BASE-4L*) for IWSLT14 German-English and Turkish-English, we use a 4-layer encoder, 4-layer decoder, residual-connected model[4], with embedding and hidden size set as 256 by default. As a comparison, we design a densely connected model with same number of layers, but the hidden size is set as 128 in order to keep the model size consistent. The models adopting DenseAtt-1, DenseAtt-2 are named as *DenseNMT-4L-1* and *DenseNMT-4L-2* respectively. In order to check the effect of dense connections on deeper models, we also construct a series of 8-layer models. We set the hidden number to be 192, such that both 4-layer models and 8-layer models have similar number of parameters.

[1] https://github.com/rsennrich/subword-nmt
[2] github.com/orhanf/zemberekMorphTR
[3] https://nlp.stanford.edu/projects/nmt/
[4] https://github.com/facebookresearch/fairseq

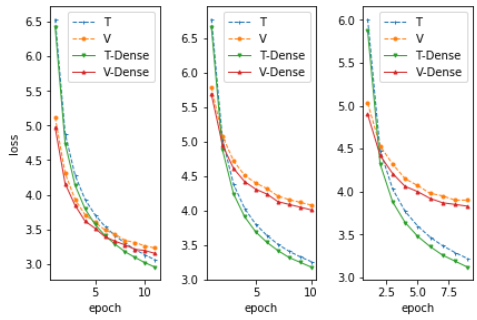


Figure 4: Training curve (T) and validation curve (V) comparison. Left: IWSLT14 German-English (De-En). Middle: Turkish-English, BPE encoding (Tr-En). Right: Turkish-English, morphology encoding (Tr-En-morph).

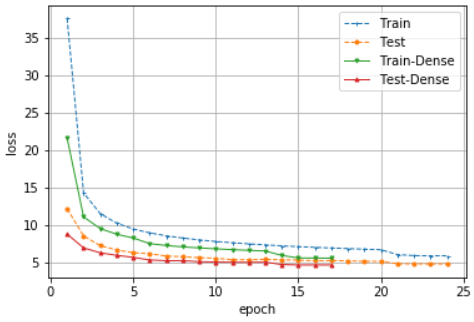


Figure 5: Training curve and test curve comparison on WMT14 English-German translation task.

For dense structured models, we set the dimension of hidden states to be 96.

Since NMT model usually allocates a large proportion of its parameters to the source/target sentence embedding and softmax matrix, we explore in our experiments to what extent decreasing the dimensions of the three parts would harm the BLEU score. We change the dimensions of the source embedding, the target embedding as well as the softmax matrix simultaneously to smaller values, and then project each word back to the original embedding dimension through a linear transformation. This significantly reduces the number of total parameters, while not influencing the upper layer structure of the model.

We also introduce three additional models we use for ablation study, all using 4-layer structure. Based on the residual connected *BASE-4L* model, (1) *DenseENC-4L* only makes encoder side dense, (2) *DenseDEC-4L* only makes decoder side dense, and (3) *DenseAtt-4L* only makes the attention dense using DenseAtt-2. There is no summary layer in the models, and both *DenseENC-4L* and *DenseDEC-4L* use hidden size 128. Again, by reducing the hidden size, we ensure that different 4-layer models have similar model sizes.

Our design for the WMT14 English-German model follows the best performance model provided in (Gehring et al., 2017). The construction of our model is straightforward: our 15-layer model *DenseNMT-En-De-15* uses dense connection with DenseAtt-2, $\texttt{sumlen} = 6$. The hidden number in each layer is $1/4$ that of the original model, while the kernel size maintains the same.

4.3 Training setting

We use Nesterov Accelerated Gradient (NAG) (Nesterov, 1983) as our optimizer, and the initial learning rate is set to 0.25. For German-English and Turkish-English experiments, the learning rate will shrink by 10 every time the validation loss increases. For the English-German dataset, in consistent with (Gehring et al., 2017), the learning rate will shrink by 10 every epoch since the first increment of validation loss. The system stops training until the learning rate is less than 10^{-4}. All models are trained end-to-end without any warmstart techniques. We set our batch size for the WMT14 English-German dataset to be 48, and additionally tune the length penalty parameter, in consistent with (Gehring et al., 2017). For other datasets, we set batch size to be 32. During inference, we use a beam size of 5.

5 Results

5.1 Training curve

We first show that DenseNMT helps information flow more efficiently by presenting the training loss curve. All hyperparameters are fixed in each plot, only the models are different. In Figure 4, the loss curves for both training and dev sets (before entering the finetuning period) are provided for De-En, Tr-En and Tr-En-morph. For clarity, we compare *DenseNMT-4L-2* with *BASE-4L*. We observe that DenseNMT models are consistently better than residual-connected models, since their loss curves are always below those of the baseline models. The effect is more obvious on the WMT14 English-German dataset. We rerun the best model provided by (Gehring et al., 2017) and compare with our model. In Figure 5, where train/test loss curve are provided, DenseNMT-En-

		De-En			Tr-En			Tr-En-morph		
	Embed size	64	128	256	64	128	256	64	128	256
	Model size (M)	8 ± 1	11 ± 1	17 ± 1	11 ± 1	17 ± 1	28 ± 1	13 ± 1	21 ± 1	36 ± 1
4L	BASE-4L	28.97	29.99	30.43	**19.80**	20.26	20.99	18.90	18.81	20.08
	DenseNMT-4L-1	**30.11**	**30.80**	31.26	19.21	20.08	21.36	18.83	20.16	21.43
	DenseNMT-4L-2	29.77	30.01	**31.40**	19.59	**20.86**	**21.48**	**19.04**	**20.19**	**21.57**
8L	BASE-8L	30.15	30.91	31.51	20.40	21.60	21.92	20.21	20.76	22.62
	DenseNMT-8L-1	**30.91**	**31.54**	32.08	21.82	**22.20**	23.20	21.20	21.73	22.60
	DenseNMT-8L-2	30.70	31.17	**32.26**	**21.93**	21.98	**23.25**	**21.73**	**22.44**	**23.45**

Table 1: BLEU score on IWSLT German-English and Turkish-English translation tasks. We compare models using different embedding sizes, and keep the model size consistent within each column.

De-15 reaches the same level of loss and starts finetuning (validation loss starts to increase) at epoch 13, which is 35% faster than the baseline.

Adding dense connections changes the architecture, and would slightly influence training speed. For the WMT14 En-De experiments, the computing time for both DenseNMT and the baseline (with similar number of parameters and same batch size) tested on single M40 GPU card are 1571 and 1710 word/s, respectively. While adding dense connections influences the per-iteration training slightly (8.1% reduction of speed), it uses many fewer epochs, and achieves a better BLEU score. In terms of training time, DenseNMT uses 29.3%(before finetuning)/22.9%(total) less time than the baseline.

5.2 DenseNMT improves accuracy with similar architectures and model sizes

	De-En	Tr-En	Tr-En-morph
BASE	30.43	20.99	20.08
DenseENC-4L	30.72	21.32	21.24
DenseDEC-4L	31.23	21.04	21.06
DenseAtt-4L	31.05	21.35	21.08
DenseNMT-4L-1	31.26	21.36	21.43
DenseNMT-4L-2	31.40	21.48	21.57

Table 2: Ablation study for encoder block, decoder block, and attention block in DenseNMT.

Table 1 shows the results for De-En, Tr-En, Tr-En-morph datasets, where the best accuracy for models with the same depth and of similar sizes are marked in boldface. In almost all genres, DenseNMT models are significantly better than the baselines. With embedding size 256, where all models achieve their best scores, DenseNMT outperforms baselines by 0.7-1.0 BLEU on De-En, 0.5-1.3 BLEU on Tr-En, 0.8-1.5 BLEU on Tr-En-morph. We observe significant gain using other embedding sizes as well.

Furthermore, in Table 2, we investigate DenseNMT models through ablation study. In order to make the comparison fair, six models listed have roughly the same number of parameters. On De-En, Tr-En and Tr-En-morph, we see improvement by making the encoder dense, making the decoder dense, and making the attention dense. Fully dense-connected model *DenseNMT-4L-1* further improves the translation accuracy. By allowing more flexibility in dense attention, *DenseNMT-4L-2* provides the highest BLEU scores for all three experiments.

From the experiments, we have seen that enlarging the information flow in the attention block benefits the models. The dense attention block provides multi-layer information transmission from the encoder to the decoder, and to the output as well. Meanwhile, as shown by the ablation study, the dense-connected encoder and decoder both give more powerful representations than the residual-connected counterparts. As a result, the integration of the three parts improve the accuracy significantly.

5.3 DenseNMT with smaller embedding size

From Table 1, we also observe that DenseNMT performs better with small embedding sizes compared to residual-connected models with regular embedding size. For example, on Tr-En model, the 8-layer *DenseNMT-8L-2* model with embedding size 64 matches the BLEU score of the 8-layer BASE model with embedding size 256, while the number of parameter of the former one is only 40% of the later one. In all genres, DenseNMT model with embedding size 128 is comparable or even better than the baseline model with embedding size 256.

While overlarge embedding sizes hurt accuracy because of overfitting issues, smaller sizes are not

	Test Set				
	tst2011	**tst2012**	**tst2013**	**tst2014**	**total**
RNN (Gulcehre et al., 2015)	18.40	18.77	19.86	18.64	/
BASE	21.66	22.45	23.76	22.59	22.62
DenseNMT-8L-2	22.52	23.81	23.91	23.68	23.45
DenseNMT-8L-2(embed 256, hid 128)	23.33	24.65	24.92	24.54	24.36

Table 3: Accuracy on Turkish-English translation task in terms of BLEU score.

preferable because of insufficient representation power. However, our dense models show that with better model design, the embedding information can be well concentrated on fewer dimensions, e.g., 64. This is extremely helpful when building models on mobile and small devices where the model size is critical. While there are other works that stress the efficiency issue by using techniques such as separable convolution (Kaiser et al., 2017), and shared embedding (Vaswani et al., 2017), our DenseNMT framework is orthogonal to those approaches. We believe that other techniques would produce more efficient models through combining with our DenseNMT framework.

	Greedy	**Beam**
MIXER (Ranzato et al., 2015)	20.73	21.83
AC (Bahdanau et al., 2017)	27.49	28.53
NPMT (Huang et al., 2017)	27.83	28.96
NPMT+LM (Huang et al., 2017)	/	29.16
DenseNMT-8L-2 (word)	29.11	30.33
DenseNMT-8L-1 (BPE)	30.50	32.08
DenseNMT-8L-2 (BPE)	30.80	32.26

Table 4: Accuracy on IWSLT14 German-English translation task in terms of BLEU score.

5.4 DenseNMT compares with state-of-the-art results

For the IWSLT14 German-English dataset, we compare with the best results reported from literatures. To be consistent with prior works, we also provide results using our model directly on the dataset without BPE preprocessing. As shown in Table 4, DenseNMT outperforms the phrase-structure based network NPMT (Huang et al., 2017) (with beam size 10) by 1.2 BLEU, using a smaller beam size, and outperforms the actor-critic method based algorithm (Bahdanau et al., 2017) by 2.8 BLEU. For reference, our model trained on the BPE preprocessed dataset achieves 32.26 BLEU, which is 1.93 BLEU higher than our word-based model. For Turkish-English task, we compare with (Gulcehre et al., 2015) which uses the same morphology preprocessing as our Tr-En-morph. As shown in Table 3, our baseline is higher than the previous result, and we further achieve new benchmark result with 24.36 BLEU average score. For WMT14 English-German, from Table 5, we can see that DenseNMT outperforms ConvS2S model by 0.36 BLEU score using 35% fewer training iterations and 20% fewer parameters. We also compare with another convolution based NMT model: SliceNet (Kaiser et al., 2017), which explores depthwise separable convolution architectures. SliceNet-Full matches our result, and SliceNet-Super outperforms by 0.58 BLEU score. However, both models have 2.2x more parameters than our model. We expect DenseNMT structure could help improve their performance as well.

	BLEU score
GNMT (Wu et al., 2016)	24.61
ConvS2S (Gehring et al., 2017)	25.16
SliceNet-Full (Kaiser et al., 2017)	25.5
SliceNet-Super (Kaiser et al., 2017)	26.1
DenseNMT-En-De-15	25.52

Table 5: Accuracy on WMT14 English-German translation task in terms of BLEU score.

6 Conclusion

In this work, we have proposed DenseNMT as a dense-connection framework for translation tasks, which uses the information from embeddings more efficiently, and passes abundant information from the encoder side to the decoder side. Our experiments have shown that DenseNMT is able to speed up the information flow and improve translation accuracy. For the future work, we will combine dense connections with other deep architectures, such as RNNs (Wu et al., 2016) and self-attention networks (Vaswani et al., 2017).

References

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. *5th International Conference on Learning Representations, ICLR, 2017* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *3rd International Conference on Learning Representations, ICLR, 2015*.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th IWSLT Evaluation Campaign, IWSLT 2014. In *Proceedings of the International Workshop on Spoken Language Translation, Hanoi, Vietnam*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar*.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *International Conference on Machine Learning*. pages 933–941.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional Sequence to Sequence Learning. *Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia.* .

Fréderic Godin, Joni Dambre, and Wesley De Neve. 2017. Improving Language Modeling using Densely Connected Recurrent Neural Networks. *arXiv preprint arXiv:1707.06130* .

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On Using Monolingual Corpora in Neural Machine Translation. *Computer Speech and Language, 2016* .

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), 2016*. pages 770–778.

Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. 2016. Densely Connected Convolutional Networks. *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), 2017* .

Po-Sen Huang, Chong Wang, Zhou Dengyong, and Deng Li. 2017. Toward Neural Phrase-based Machine Translation. *arXiv preprint arXiv:1706.05565* .

Simon Jégou, Michal Drozdzal, David Vazquez, Adriana Romero, and Yoshua Bengio. 2017. The One Hundred Layers Tiramisu: Fully Convolutional Densenets for Semantic Segmentation. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*. IEEE, pages 1175–1183.

Lukasz Kaiser, Aidan N Gomez, and Francois Chollet. 2017. Depthwise Separable Convolutions for Neural Machine Translation. *arXiv preprint arXiv:1706.03059* .

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *EMNLP*. volume 3, page 413.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in Translation: Contextualized Word Vectors. *arXiv preprint arXiv:1708.00107* .

Yurii Nesterov. 1983. A method of solving a convex programming problem with convergence rate o (1/k2). In *Soviet Mathematics Doklady*. volume 27, pages 372–376.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence Level Training with Recurrent Neural Networks. *4th International Conference on Learning Representations, ICLR, 2016* .

Haşim Sak, Tunga Güngör, and Murat Saraçlar. 2007. Morphological Disambiguation of Turkish Text with Perceptron Algorithm. *Computational Linguistics and Intelligent Text Processing* pages 107–118.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015a. Improving Neural Machine Translation Models with Monolingual Data. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Berlin, Germany, 2016* .

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015b. Neural Machine Translation of Rare Words with Subword Units. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Berlin, Germany, 2016* .

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. pages 6000–6010.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's Neural Machine

Translation System: Bridging the Gap between Human and Machine Translation. *arXiv preprint arXiv:1609.08144* .

Evaluating Discourse Phenomena in Neural Machine Translation

Rachel Bawden[1] **Rico Sennrich**[2,3] **Alexandra Birch**[2] **Barry Haddow**[2]
[1]LIMSI, CNRS, Univ. Paris-Sud, Université Paris-Saclay, F-91405 Orsay, France
[2]School of Informatics, University of Edinburgh, Scotland
[3]Institute of Computational Linguistics, University of Zurich, Switzerland
rachel.bawden@limsi.fr
{rico.sennrich, a.birch}@ed.ac.uk
bhaddow@inf.ed.ac.uk

Abstract

For machine translation to tackle discourse phenomena, models must have access to extra-sentential linguistic context. There has been recent interest in modelling context in neural machine translation (NMT), but models have been principally evaluated with standard automatic metrics, poorly adapted to evaluating discourse phenomena. In this article, we present hand-crafted, discourse test sets, designed to test the models' ability to exploit previous source and target sentences. We investigate the performance of recently proposed multi-encoder NMT models trained on subtitles for English to French. We also explore a novel way of exploiting context from the previous sentence. Despite gains using BLEU, multi-encoder models give limited improvement in the handling of discourse phenomena: 50% accuracy on our coreference test set and 53.5% for coherence/cohesion (compared to a non-contextual baseline of 50%). A simple strategy of decoding the concatenation of the previous and current sentence leads to good performance, and our novel strategy of multi-encoding and decoding of two sentences leads to the best performance (72.5% for coreference and 57% for coherence/cohesion), highlighting the importance of target-side context.

1 Introduction

Machine translation (MT) systems typically translate sentences independently of each other. However, certain textual elements cannot be correctly translated without linguistic context, which may appear outside the current sentence. The most obvious examples of context-dependent phenomena problematic for MT are coreference (Guillou, 2016), lexical cohesion (Carpuat, 2009) and lexical disambiguation (Rios Gonzales et al., 2017), an example for each of which is given in (1-3). In each case, the English element in italic is ambiguous in terms of its French translation. The correct translation choice (in bold) is determined by linguistic context (underlined), which can be outside the current sentence. This disambiguating context can be source or target-side; the correct translation of anaphoric pronouns *it* and *they* depends on the gender of the translated antecedent (1). In lexical cohesion, a translation may depend on target factors, but may also be triggered by source effects and linguistic mechanisms such as repetition or alignment (2). In lexical disambiguation, source or target information may provide the appropriate context (3).

(1) The bee is busy. // *It* is making honey.
L'abeille$_{[f]}$ est occupée. // **Elle**$_{[f]}$/#il$_{[m]}$ fait du miel.

(2) Do you fancy some soup? // *Some soup?*
Tu veux de la soupe? // **De la soupe**/#du potage?

(3) And the code? // Still some *bugs*...
Et le code ? // Encore quelques **bugs**/#insectes...

Recent work on multi-encoder neural machine translation (NMT) appears promising for the integration of linguistic context (Zoph and Knight, 2016; Libovický and Helcl, 2017; Jean et al., 2017a; Wang et al., 2017). However models have almost only been evaluated using standard automatic metrics, which are poorly adapted to evaluating discourse phenomena. Targeted evaluation, in particular of coreference in MT, has proved to be time-consuming and laborious (Guillou, 2016).

In this article, we address the evaluation of discourse phenomena for MT and propose a novel contextual model. We present two hand-crafted, discourse test sets designed to test models' capacity to exploit linguistic context for coreference and coherence/cohesion for English to French translation. Using these sets, we review contextual NMT strategies trained on subtitles in a high-resource

Proceedings of NAACL-HLT 2018, pages 1304–1313
New Orleans, Louisiana, June 1 - 6, 2018.

setting. Our new combination of strategies outperforms previous methods according to our targeted evaluation and the standard metric BLEU.

2 Evaluating contextual phenomena

Traditional automatic metrics are notoriously problematic for the evaluation of discourse in MT (Hardmeier, 2014); discursive phenomena may have an impact on relatively few word forms with respect to their importance, meaning that improvements are overlooked, and a correct translation may depend on target-side coherence rather than similarity to a reference translation.

Coreference has been a major focus of discourse translation, spurred on by shared tasks on cross-lingual pronoun prediction (Guillou et al., 2016; Loáiciga et al., 2017). Participants were provided with lemmatised versions of reference translations,[1] in which pronoun forms were to be predicted. Evaluation in this setting (with the use of reference translations) was possible with traditional metrics, because the antecedents were fixed in advance. However there are at least two disadvantages to the approach: (i) models must be trained on lemmatised data and cannot be used in a real translation setting, and (ii) many of the pronouns did not need extra-sentential context; easier gains were seen for the pronouns with intra-sentential antecedents and therefore the leaderboard was dominated by sentence-level systems.

Guillou and Hardmeier's (2016) pronoun translation test suite succeeds in overcoming some of these problems by creating an automatic evaluation method, with a back-off manual evaluation. Manual evaluation has always been an essential part of evaluating MT quality, and targeted translation allows us to isolate a model's performance on specific linguistic phenomena; recent work using in-depth, qualitative manual evaluation (Isabelle et al., 2017; Scarton and Specia, 2015) is very informative. Isabelle et al. (2017) focus on specially constructed challenging examples in order to analyse differences between systems. They cover a wide range of linguistic phenomena, but since manual evaluation is costly and time-consuming, only a few examples per phenomenon are analysed, and it is difficult to obtain quick, quantitative feedback.

An alternative method, which overcomes the problem of costly, one-off analysis, is to evaluate models' capacity to correctly rank contrastive pairs of pre-existing translations, of which one is correct and the other incorrect. This method was used by Sennrich (2017) to assess the grammaticality of character-level NMT and again by Rios Gonzales et al. (2017) in a large-scale setting for lexical disambiguation for English-German. The method allows automatic quantitative evaluation of specific phenomena at large scale, at the cost of only testing for very specific translation errors. It is also the strategy that we will use here to evaluate translation of discourse phenomena.

2.1 Our contrastive discursive test sets

We created two contrastive test sets to help compare how well different contextual MT models handle (i) anaphoric pronoun translation and (ii) coherence and cohesion.[2] For each test set, models are assessed on their ability to rank the correct translation of an ambiguous sentence higher than the incorrect translation, using the disambiguating context provided in the previous source and/or target sentence.[3] All examples in the test sets are hand-crafted but inspired by real examples from OpenSubtitles2016 (Lison and Tiedemann, 2016) to ensure that they are credible and that vocabulary and syntactic structures are varied. The method can be used to evaluate any NMT model, by making it produce a score for a given source sentence and reference translation.

Our test sets differ from previous ones in that examples necessarily need the previous context (source and/or target-side) for the translations to be correctly ranked. Unlike the shared task test sets, the ambiguous pronouns' antecedents are guaranteed not to appear within the current sentence, meaning that, for MT systems to score highly, they must use discourse-level context. Compared to other test sets suites, ours differs in that evaluation is performed completely automatically and concentrates specifically on the model's ability to use context. Each of the test sets contains

[1] This was to avoid pronoun forms being trivial to predict from the morphological inflections of other forms within the sentence, an unrealistic setting for MT output.

[2] The test sets are freely available at https://diamt.limsi.fr/eval.html.

[3] We acknowledge that in reality, the disambiguating context is not guaranteed to be in the previous sentence (cf. Guillou (2016, p. 161), for the distribution of intra- and inter-sentential anaphoric pronouns). However it is important to first judge in a controlled way whether or not models are actually capable of using extra-sentential linguistic context at all, before investigating longer distance context.

200 contrastive pairs and is designed such that a non-contexual baseline system would achieve 50% accuracy.

Source:
context: Oh, I hate **flies**. Look, there's another one!
current sent.: Don't worry, I'll kill **it** for you.

Target:

1	context:	Ô je déteste les **mouches**. Regarde, il y en a une autre !
	correct:	T'inquiète, je **la** tuerai pour toi.
	incorrect:	T'inquiète, je **le** tuerai pour toi.
2	context:	Ô je déteste les **moucherons**. Regarde, il y en a un autre !
	correct:	T'inquiète, je **le** tuerai pour toi.
	incorrect:	T'inquiète, je **la** tuerai pour toi.
3	context:	Ô je déteste les **araignées**. Regarde, il y en a une autre !
	semi-correct:	T'inquiète, je **la** tuerai pour toi.
	incorrect:	T'inquiète, je **le** tuerai pour toi.
4	context:	Ô je déteste les **papillons**. Regarde, il y en a un autre !
	semi-correct:	T'inquiète, je **le** tuerai pour toi.
	incorrect:	T'inquiète, je **la** tuerai pour toi.

Figure 1: Example block from the coreference set.

Coreference test set This set contains 50 example blocks, each containing four contrastive translation pairs (see the four examples in Fig. 1). The test set's aim is to test the integration of target-side linguistic context. Each block is defined by a source sentence containing an occurrence of the anaphoric pronoun *it* or *they* and its preceding context, containing the pronoun's nominal antecedent.[4] Four contrastive translation pairs of the previous and current source sentence are given, each with a different translation of the nominal antecedent, of which two are feminine and two are masculine per block. Each pair contains a correct translation of the current sentence, in which the pronoun's gender is coherent with the antecedent's translation, and a contrastive (incorrect) translation, in which the pronoun's gender is inversed (along with agreement linked to the pronoun choice). Two of the pairs contain what we refer to as a "semi-correct" translation of the current sentence instead of a "correct" one, for which the antecedent in the previous sentence is strangely or wrongly translated (e.g. *flies* translated as *araignées* "spiders" and *papillons* "butterflies" in Fig. 1). In the "semi-correct" translation, the pronoun, whose translation is wholly dependent on the translated antecedent, is coherent with this translation choice. These semi-correct examples assess the use of target-side context, taking into account previous translation choices.

Target pronouns are evenly distributed according to number and gender with 50 examples (25 correct and 25 semi-correct) for each of the pronoun types (m.sg, f.sg, m.pl and f.pl). Since there are only two possible translations of the current sentence per example block, an MT system can only score all examples within a block correctly if it correctly disambiguates, and a non-contextual baseline system is guaranteed to score 50%.

Source:
context: What's **crazy** about me?
current sent.: Is this **crazy**?

Target:
context: Qu'est-ce qu'il y a de **dingue** chez moi ?
correct: Est-ce que ça c'est **dingue** ?
incorrect: Est-ce que ça c'est fou ?

Source:
context: What's **crazy** about me?
current sent.: Is this **crazy**?

Target:
context: Qu'est-ce qu'il y a de **fou** chez moi ?
correct: Est-ce que ça c'est **fou** ?
incorrect: Est-ce que ça c'est dingue ?

Figure 2: Example block from the coherence/cohesion test: alignment.

Source:
context: So what do you say to £50?
current sent.: It's a little **steeper** than I was expecting.

Target:
context: Qu'est-ce que vous en pensez de 50£ ?
correct: C'est un peu plus **cher** que ce que je pensais.
incorrect: C'est un peu plus **raide** que ce que je pensais.

Source:
context: How are your feet holding up?
current sent.: It's a little **steeper** than I was expecting.

Target:
context: Comment vont tes pieds ?
correct: C'est un peu plus **raide** que ce que je pensais.
incorrect: C'est un peu plus **cher** que ce que je pensais.

Figure 3: Example block from the coherence/cohesion test: lexical disambiguation.

[4]The choice to use only nominal antecedents and only two anaphoric pronouns *it* and *they* is intentional in order to provide a controlled environment in which there are two contrasting alternatives for each example. This ensures that a non-contextual baseline necessarily gives a score of 50%, and also enables us to explore this simpler case before expanding the study to explore more difficult anaphoric phenomena.

Coherence and cohesion test set Coherence and cohesion concern the interpretation of a text in the context of discourse (i.e. beyond sentence level). De Beaugrande and Dressler (1981) define the dichotomous pair as representing two separate aspects: coherence relating to the consistency of the text to concepts and world knowledge, and cohesion relating to the surface formulation of the text, as expressed through linguistic mechanisms.

This set contains 100 example blocks, each containing two contrastive pairs (see Figs. 2 and 3). Each of the blocks is constructed such that there is a single ambiguous source sentence, with two possible translations provided. The use of one translation over the other is determined by disambiguation context found in the previous sentence. The context may be found on the source side, the target side or both. In each contrastive pair, the incorrect translation of the current sentence corresponds to the correct translation of the other pair, such that the block can only be entirely correct if the disambiguating context is correctly used.

All test set examples have in common that the current English sentence is ambiguous and that its correct translation into French relies on context in the previous sentence. In some cases, the correct translation is determined more by cohesion, for example the necessity to respect alignment or repetition (Fig. 2). This means that despite two translations of an English source word being synonyms (e.g. *dingue* and *fou*, "crazy"), they are not interchangeable in a discourse context, given that the chosen formulation (alignment) requires repetition of the word of the previous sentence. In other cases, lexical choice is determined more by cohesion, for example by a general semantic context provided by the previous sentence, in a more classic disambiguation setting as in Fig. 3, where the English *steeper* is ambiguous between French *cher* "more expensive" and *raide* "sharply sloped". However, these types are not mutually exclusive and the distinction is not always so clear.

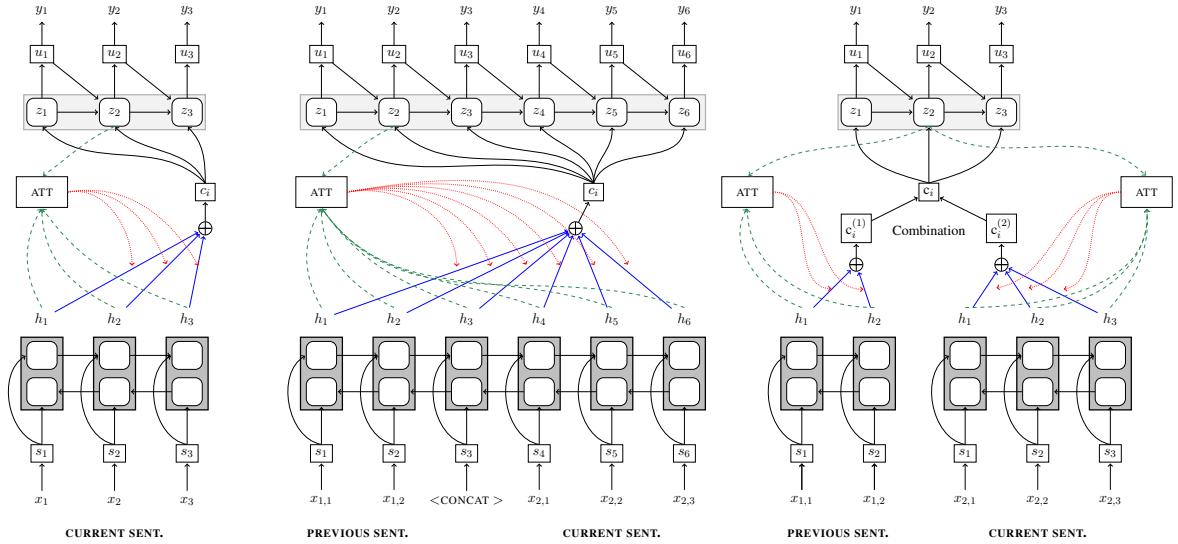
3 Contextual NMT Models

In order to correctly translate the type of phenomena mentioned in Sec. 1, translation models need to look beyond the sentence. Much of the previous work, mainly in statistical machine translation (SMT), focused on post-edition, particularly for anaphoric pronoun translation (Guillou et al., 2016; Loáiciga et al., 2017). However, coreference resolution is not yet sufficient for high quality post- or pre-edition (Bawden, 2016), and for other discourse phenomena such as lexical cohesion and lexical disambiguation, detecting the disambiguating context is far from trivial.

Recent work in NMT has explored multi-input models, which integrate the previous sentence as an auxiliary input. A simple strategy of concatenating the previous sentence to the current sentence and using a basic NMT architecture was explored by Tiedemann and Scherrer (2017), but with mixed results. A variety of multi-encoder strategies have also been tested, including using a representation of the previous sentence to initialise the main encoder and/or decoder (Wang et al., 2017) and using multiple attention mechanisms, with different strategies to combine the resulting context vectors, such as concatenation (Zoph and Knight, 2016), hierarchical attention (Libovický and Helcl, 2017) and gating (Jean et al., 2017a).

Although some of the models were evaluated in a contextual setting, for example on the cross-lingal pronoun prediction task at DiscoMT17 (Jean et al., 2017b), certain strategies only appear to give gains in a low-resource setting (Jean et al., 2017a), and, more importantly, there has yet to be an in-depth study into which strategies work best specifically for context-dependent discursive phenomena. Here we provide such a study, using the targeted test sets described in Sec. 2 to isolate and evaluate the different contextual models' capacity to exploit extra-sentential context. We test several contextual variants, using both a single encoder (Sec. 3.1) and multiple encoders (Sec. 3.2).

NMT notation All models presented are based on the widely used encoder-decoder NMT framework with attention (Bahdanau et al., 2015). At each decoder step i, the context (or summary) vector c_i of the input sequence is a weighted average of the recurrent encoder states at each input position depending on the attention weights. We refer to the recurrent state of the decoder as z_i. When multiple inputs are concerned, inputs are noted $x_j^{(k)}$, where k is the input number and j the input position. Likewise, when multiple encoders are used, $c_i^{(k)}$ refers to the k^{th} context vector where k is the encoder number. In the following section, all Ws, Us and bs are learned parameters.



(a) S2S with attention (BASELINE).

(b) Concatenate input (2-TO-2, 2-TO-1).

(c) Multi-source S2S with attention. The three combination methods tested are CONCAT, HIER and GATE.

Figure 4: The baseline model and the two contextual strategies tested (single and multi-encoder).

3.1 Single-encoder models

We train three single-source models: a baseline model and two contextual models. The baseline model translates sentences independently of each other (Fig. 4a). The two contextual models, described in (Tiedemann and Scherrer, 2017), are designed to incorporate the preceding sentence by prepending it to the current one, separated by a <CONCAT> token (Fig. 4b). The first method, which we refer to as 2-TO-2, is trained on concatenated source and target sentences, such that the previous and current sentence are translated together. The translation of the current sentence is obtained by extracting the tokens following the translated concatenation token and discarding preceding tokens.[5] The second method, 2-TO-1, follows the same principle, except that only source (and not target) sentences undergo concatenation; the model directly produces the translation of the current sentence. The comparison of these two methods allows us to assess the impact of the decoder in producing contextual translations.

3.2 Multi-encoder models

Inspired by work on multi-modal translation (Caglayan et al., 2016; Huang et al., 2016), multi-encoder translation models have recently been used to incorporate extra-sentential linguistic context in purely textual NMT (Zoph and Knight, 2016; Libovický and Helcl, 2017; Wang et al., 2017). Unlike multi-modal translation, which typically uses two complementary representations of the main input, for example a textual description and an image, linguistically contextual NMT has focused on exploiting the previous linguistic context as auxiliary input alongside the current sentence to be translated. Within this framework, we encode the previous sentence using a separate encoder (with separate parameters) to produce a context vector of the auxiliary input in a parallel fashion to the current source sentence. The two resulting context vectors $c_i^{(1)}$ and $c_i^{(2)}$ are then combined to form a single context vector c_i to be used for decoding (see Fig. 4c). We study three combination strategies here: concatenation, an attention gate and hierarchical attention. We also tested using the auxiliary context to initialise the decoder, similar to Wang et al. (2017), which was ineffective in our experiments and which we therefore do not report in this paper.

Attention concatenation The two context vectors $c_i^{(1)}$ and $c_i^{(2)}$ are concatenated and the resulting vector undergoes a linear transformation in order to return it to its original dimension to produce c_i (similar to work by Zoph and Knight (2016)).

$$c_i = W_c[c_i^{(1)}; c_i^{(2)}] + b_c \quad (1)$$

[5] Although the non-translation of the concatenation symbol is possible, in practice this was rare (<0.02%). If this occurs, the whole translation is kept.

Attention gate A gate r_i is learnt between the two vectors in order to give differing importance to the elements of each context vector, similar to the strategy of Wang et al. (2017).

$$r_i = \tanh\left(W_r c_i^{(1)} + W_s c_i^{(2)}\right) + b_r \quad (2)$$

$$c_i = r_i \odot \left(W_t c_i^{(1)}\right) + (1 - r_i) \odot \left(W_u c_i^{(2)}\right) \quad (3)$$

Hierarchical attention An additional (hierarchical) attention mechanism (Libovický and Helcl, 2017) is introduced to assign a weight to each encoder's context vector (designed for an arbitrary number of encoders).

$$e_i^{(k)} = v_b^\top \tanh\left(W_b z_{(i-1)} + U_b^{(k)} c_i^{(k)}\right) + b_e \quad (4)$$

$$\beta_i^{(k)} = \frac{\exp\left(e_i^{(k)}\right)}{\sum_{k'=1}^{K} \exp\left(e_i^{(k')}\right)} \quad (5)$$

$$c_i = \sum_{k=1}^{K} \beta_i^{(k)} U_c^{(k)} c_i^{(k)} \quad (6)$$

3.3 Novel strategy of hierarchical attention and context decoding

We also test a novel strategy of combining multiple encoders and decoding of both the previous and current sentence. We use separate, multiple encoders to encode the previous and current sentence and combine the context vectors using hierarchical attention. We train the model to produce the concatenation of the previous and current target sentences, of which the second part is kept, as in the contextual single encoder models.

4 Experiments

Each of the multi-encoder strategies is tested using the previous source and target sentences as an additional input (prefixed as S- and T- respectively) in order to test which is the most useful disambiguating context. Two additional models tested are triple-encoder models, which use both the previous source and target (prefixed as S-T-).

4.1 Data

Models are trained and tested on fan-produced parallel subtitles from OpenSubtitles2016[6] (Lison and Tiedemann, 2016). The data is first corrected using heuristics (e.g. minor corrections of OCR and encoding errors). It is then tokenised, further cleaned (keeping subtitles $\leq$80 tokens) and truecased using the Moses toolkit (Koehn et al., 2007) and finally split into subword units using BPE (Sennrich et al., 2016).[7] We run all experiments in a high-resource setting, with a training set of ≈29M parallel sentences, with vocabulary sizes of ≈55k for English and ≈60k for French.

4.2 Experimental setup

All models are sequence-to-sequence models with attention (Bahdanau et al., 2015), implemented in Nematus (Sennrich et al., 2017). Training is performed using the Adam optimiser with a learning rate of 0.0001 until convergence. We use embedding layers of dimension 512 and hidden layers of dimension 1024. For training, the maximum sentence length is 50.[8] We use batch sizes of 80, tied decoder embeddings and layer normalisation. The hyper-parameters are the same for all models and are the same as those used for the University of Edinburgh submissions to the news translation shared task at WMT16 and WMT17. Final models are ensembled using the last three checkpointed models.

Models that use the previous target sentence are trained using the previous reference translation. During translation, baseline translations are used. For the targeted evaluation, the problem does not apply since the translations that are being scored are given.

5 Results and Analysis

Overall translation quality is evaluated using the traditional automatic metric BLEU (Papineni et al., 2002) (Tab. 1) to ensure that the models do not degrade overall performance. We test the models' ability to handle discursive phenomena using the test sets described in Sec. 2 (Tab. 2). The models are described in the first half of Table 1: *#In* is the number of input sentences, the type of auxiliary input of which (previous source or target) is indicated by *Aux.*, *#Out* is the number of sentences translated, and *#Enc* is the number of encoders used to encode the input sentences. When there is a single encoder and more than one input, the input sentences are concatenated to form a single input to the encoder.

[6] http://www.opensubtitles.org

[7] 90,000 merge operations with a minimum theshold of 50.

[8] 76 when source sentences are concatenated to the previous sentence in order to keep the same percentage of training sentences as for other models.

System Description					BLEU ↑			
	Aux.	#In	#Out	#Enc.	Comedy	Crime	Fantasy	Horror
Single-encoder, non-contexual model								
BASELINE	✗	1	1	1	19.52	22.07	26.30	33.05
Single-encoder with concatenated input								
2-TO-2	src	2	2	1	**20.09**	**22.93**	26.60	33.59
2-TO-1	src	2	1	1	19.51	21.81	26.78	**34.37**
Multi-encoder models (+previous target sentence)								
T-CONCAT	trg	2	1	2	18.33	20.90	24.36	32.90
T-HIER	trg	2	1	2	17.89	20.77	25.42	31.93
T-GATE	trg	2	1	2	18.25	20.76	25.55	32.64
Multi-encoder models (+previous source sentence)								
S-CONCAT	src	2	1	2	19.35	22.41	26.50	33.67
S-HIER	src	2	1	2	**20.22**	21.90	26.81	**34.04**
S-GATE	src	2	1	2	19.89	**22.80**	**26.87**	33.81
S-T-HIER	src, trg	3	1	3	19.53	22.53	**26.87**	33.24
Multi-encoder with concatenated output								
S-HIER-TO-2	src	2	2	2	**20.85**	**22.81**	**27.17**	**34.62**
S-T-HIER-TO-2	src, trg	3	2	3	18.80	21.18	**27.68**	33.33

Table 1: Results (de-tokenised, cased BLEU) of the ensembled models on four different test sets, each containing three films from each film genre. The best, second- and third-best results are highlighted by decreasingly dark shades of green.

5.1 Overall performance

Results using the automatic metric BLEU are given in Tab. 1. The models are tested on four different genres of film: comedy, crime, fantasy and horror.[9] Scores vary dramatically depending on the genre and the best model is not always the same for each of the genres.

Contrary to intuition, using the previous target sentence as an auxiliary input (prefix T-) degrades the overall performance considerably. Testing at decoding time with the reference translations did not significantly improve this result, suggesting that it is unlikely to be a case of overfitting during training. The highest performing model is our novel S-HIER-TO-2 model with more than +1 over the baseline BLEU on almost all test sets. There is no clear second best model, since performance depends strongly on the test set used.

5.2 Targeted evaluation

Tab. 2 shows the results on the discourse test sets.

Coreference The multi-encoder models do not perform well on the coreference test set; all multi-encoder models giving at best random accuracy, as with the baseline. This set is designed to test the model's capacity to exploit previous target context. It is therefore unsurprising that multi-encoder models using just the previous source sentence perform poorly. It is possible that certain pronouns could be correctly predicted from the source antecedents, if the antecedent only has one possible translation. However, this non-robust way of translating pronouns is not tested by the test set. More surprisingly, the multi-encoder models using the previous target sentence also perform poorly on the test set. An explanation could be that the target sentence is not being encoded sufficiently well in this framework, resulting in poor learning. This hypothesis is supported by the low overall translation performance shown in Tab. 1.

Two models perform well on the test set: 2-TO-2 and our S-HIER-TO-2. The high scores, particularly on the less common feminine pronouns, which can only be achieved through using contextual linguistic information, show that these models are capable of using previous linguistic context to disambiguate pronouns. The progressively high performance of these models can be seen in Fig. 5, which illustrates the training progress of these models. The S-T-HIER-TO-2 model (which uses the previous target sentence as a third auxiliary input) performs much worse than S-HIER-TO-2, showing that the addition of the previous target sentence is detrimental to performance. Whilst the

[9] Each of the test sets contains three films from that genre, with varying sizes and difficulty. The number of sentences in each test set is as follows: comedy: 4,490, crime: 4,227, fantasy: 2,790 and horror: 2,158.

	Coreference (%)							Coherence/cohesion (%)
	ALL	M.SG.	F.SG.	M.PL.	F.PL	CORR.	SEMI	ALL
BASELINE	50.0	80.0	20.0	80.0	20.0	53.0	47.0	50.0
2-TO-2	**63.5**	**92.0**	**50.0**	84.0	**28.0**	**68.0**	**59.0**	52.0
2-TO-1	52.0	72.0	28.0	84.0	**24.0**	54.0	50.0	**53.0**
T-CONCAT	49.0	88.0	8.0	**96.0**	4.0	50.0	48.0	51.5
T-HIER	47.0	78.0	10.0	**90.0**	10.0	47.0	47.0	50.5
T-GATE	47.0	80.0	6.0	82.0	20.0	45.0	49.0	49.0
S-CONCAT	50.0	68.0	32.0	88.0	12.0	**55.0**	45.0	**53.5**
S-HIER	50.0	64.0	**36.0**	80.0	20.0	**55.0**	45.0	**53.0**
S-GATE	50.0	68.0	32.0	84.0	16.0	**55.0**	45.0	51.5
S-T-HIER	49.5	**94.0**	4.0	88.0	12.0	53.0	46.0	**53.0**
S-HIER-TO-2	**72.5**	**100.0**	**40.0**	**90.0**	**36.0**	**77.0**	**68.0**	**57.0**
S-T-HIER-TO-2	**56.5**	84.0	**36.0**	86.0	20.0	**55.0**	**58.0**	51.5

Table 2: Results on the discourse test sets (% correct). Results on the coreference set are also given for each pronoun class. CORR. and SEMI correspond respectively to the "correct" and "semi-correct" examples. The best, second- and third-best results are highlighted by decreasingly dark shades of green.

results for the "correct" examples (CORR.) are almost always higher than the "semi-correct" examples (SEMI), for which the antecedent is strangely translated, the TO-2 models also give improved results on these examples, showing that the target context is necessarily being exploited during decoding.

These results show that the translation of the previous sentence is the most important factor in the efficient use of linguistic context. Combining the S-HIER model with decoding of the previous target sentence (S-HIER-TO-2) produces some of the best results across all pronoun types, and the 2-TO-2 model performs almost always second best.

Coherence and cohesion Much less variation in scores can be seen here, suggesting that these examples are more challenging and that there is room for improvement. Unlike the coreference examples, the multi-encoder strategies exploiting the previous source sentences perform better than the baseline (up to 53.5% for S-CONCAT). Yet again, using the previous target sentence achieves near random accuracy. 2-TO-2 and 2-TO-1 achieve similarly low scores (52% and 53%), suggesting that if concatenated input is used, decoding the previous sentence does not add more information.

However, combining multi-encoding with the decoding of the previous and the current sentences (S-HIER-TO-2) greatly improves the handling of the ambiguous translations, improving the accuracy to 57%. Extending this same model to also exploit the previous target sentence (S-T-HIER-TO-2) degrades this result, giving very similar scores to T-HIER and is therefore not illustrated in FIgure 5. This provides further support for the idea that the target sentence is not encoded efficiently as an auxiliary input and adds noise to the model, whereas exploiting the target context as a bias in the recurrent decoder is more effective.

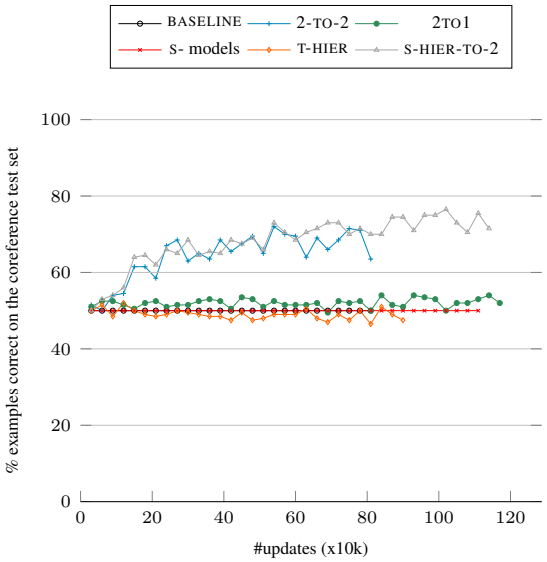


Figure 5: Progression of % correctly ranked examples (from the coreference test set) during training.

5.3 How much is the context being used?

Looking at the attention weights can sometimes offer insights into which input elements are being attended to at each step. For coreference resolution, we would expect the decoder to attend to the pronoun's antecedent. The effect is most expected when the previous target sentence is used, but it could also apply for the previous source sentence when the antecedent has only one possible translation. Unlike Tiedemann and Scherrer (2017), we do not observe increased attention between a translated pronoun and its source antecedent. Given the discourse test set results, which can only give high scores when target-side context is used, the contextual information of the type studied in this paper seems to be best exploited when channelled through the recurrent decoder node rather than when encoded through the input. This could explain why coreference is not easily seen via attention weights; the crucial information is encoded on the decoder-side rather than in the encoder.

6 Conclusion

We have presented an evaluation of discourse-level NMT models through the use of two discourse test sets targeted at coreference and lexical coherence/cohesion. We have shown that multi-encoder architectures alone have a limited capacity to exploit discourse-level context; poor results are found for coreference and more promising results for coherence/cohesion, although there is room for improvement. Our novel combination of contextual strategies greatly outperfoms existing models. This strategy uses the previous source sentence as an auxiliary input and decodes both the current and previous sentence. The observation that the decoding strategy is very effective for the handling of previous context suggests that techniques such as stream decoding, keeping a constant flow of contextual information in the recurrent node of the decoder, could be very promising for future research.

Acknowledgments

Rico Sennrich has received funding from the Swiss National Science Foundation (SNF) in the project CoNTra (grant number 105212_169888). This project has also received funding from the European Union's Horizon 2020 research and innovation programme under grant agreements 644333 (SUMMA) and 644402 (HimL).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of the 3rd International Conference on Learning Representations*. ICLR'15. ArXiv: 1409.0473.

Rachel Bawden. 2016. Cross-lingual Pronoun Prediction with Linguistically Informed Features. In *Proceedings of the 1st Conference on Machine Translation*. Berlin, Germany, WMT'16, pages 564–570.

Ozan Caglayan, Loïc Barrault, and Fethi Bougares. 2016. Multimodal Attention for Neural Machine Translation. In *arXiv:1609.03976*.

Marine Carpuat. 2009. One translation per discourse. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*. Boulder, Colorado, USA, SEW'09, pages 19–27.

Robert De Beaugrande and Wolfgang Dressler. 1981. *Introduction to Text Linguistics*. Longman, London.

Liane Guillou. 2016. *Incorporating Pronoun Function into Statistical Machine Translation*. Ph.D. thesis, School of Informatics. University of Edinburgh.

Liane Guillou and Christian Hardmeier. 2016. PROTEST: A Test Suite for Evaluating Pronouns in Machine Translation. In *Proceedings of the 10th Language Resources and Evaluation Conference*. Portorož, Slovenia, LREC'16, pages 636–643.

Liane Guillou, Christian Hardmeier, Preslav Nakov, Sara Stymne, Jörg Tiedemann, Yannick Versley, Mauro Cettolo, Bonnie Webber, and Andrei Popescu-Belis. 2016. Findings of the 2016 WMT Shared Task on Cross-lingual Pronoun Prediction. In *Proceedings of the 1st Conference on Machine Translation*. Berlin, Germany, WMT'16, pages 525–542.

Christian Hardmeier. 2014. *Discourse in Statistical Machine Translation*. Ph.D. thesis, Uppsala University, Department of Linguistics and Philology, Uppsala, Sweden.

Po-Yao Huang, Frederick Liu, Sz-Rung Shiang, Jean Oh, and Chris Dyer. 2016. Attention-based Multimodal Neural Machine Translation. In *Proceedings of the 1st Conference on Machine Translation*. Berlin, Germany, volume 2: of *WMT'16*, pages 639–645.

Pierre Isabelle, Colin Cherry, and George Foster. 2017. A Challenge Set Approach to Evaluating Machine Translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark, EMNLP'17, pages 2476–2486.

Sebastien Jean, Stanislas Lauly, Orhan Firat, and Kyunghyun Cho. 2017a. Does Neural Machine Translation Benefit from Larger Context? In *arXiv:1704.05135*. ArXiv: 1704.05135.

Sebastien Jean, Stanislas Lauly, Orhan Firat, and Kyunghyun Cho. 2017b. Neural Machine Translation for Cross-Lingual Pronoun Prediction. In *Proceedings of the 3rd Workshop on Discourse in Machine Translation*. Copenhagen, Denmark, DISCOMT'17, pages 54–57.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*. Prague, Czech Republic, ACL'07, pages 177–180.

Jindřich Libovický and Jindřich Helcl. 2017. Attention Strategies for Multi-Source Sequence-to-Sequence Learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Vancouver, Canada, ACL'17, pages 196–202.

Pierre Lison and Jörg Tiedemann. 2016. OpenSubtitles2016: Extracting Large Parallel Corpora from Movie and TV Subtitles. In *Proceedings of the 10th Language Resources and Evaluation Conference*. Portorož, Slovenia, LREC'16, pages 923–929.

Sharid Loáiciga, Sara Stymne, Preslav Nakov, Christian Hardmeier, Jörg Tiedemann, Mauro Cettolo, and Yannick Versley. 2017. Findings of the 2017 DiscoMT Shared Task on Cross-lingual Pronoun Prediction. In *Proceedings of the 3rd Workshop on Discourse in Machine Translation*. Copenhagen, Denmark, DISCOMT'17, pages 1–16.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Philadelphia, USA, ACL'02, pages 311–318.

Annette Rios Gonzales, Laura Mascarell, and Rico Sennrich. 2017. Improving Word Sense Disambiguation in Neural Machine Translation with Sense Embeddings. In *Proceedings of the 2nd Conference on Machine Translation*. Copenhagen, Denmark, WMT'17, pages 11–19.

Carolina Scarton and Lucia Specia. 2015. A Quantitative Analysis of Discourse Phenomena in Machine Translation. *Discours [online]* (16). https://doi.org/10.4000/discours.9047.

Rico Sennrich. 2017. How Grammatical is Character-level Neural Machine Translation? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. Valencia, Spain, EACL'17, pages 376–382.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio, Miceli Barone, Jozef Mokry, and Maria Nădejde. 2017. Nematus: a Toolkit for Neural Machine Translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. Valencia, Spain, EACL'17, pages 65–68.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Berlin, Germany, ACL'16, pages 1715–1725.

Jörg Tiedemann and Yves Scherrer. 2017. Neural Machine Translation with Extended Context. In *Proceedings of the 3rd Workshop on Discourse in Machine Translation*. Copenhagen, Denmark, DISCOMT'17, pages 82–92.

Longyue Wang, Zhaopeng Tu, Andy Way, and Qun Liu. 2017. Exploiting Cross-Sentence Context for Neural Machine Translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Denmark, Copenhagen, EMNLP'17, pages 2816–2821.

Barret Zoph and Kevin Knight. 2016. Multi-source Neural Translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics*. San Diego, California, USA, NAACL'16, pages 30–34.

Fast Lexically Constrained Decoding with Dynamic Beam Allocation for Neural Machine Translation

Matt Post and **David Vilar**
Amazon Research
Berlin, Germany

Abstract

The end-to-end nature of neural machine translation (NMT) removes many ways of manually guiding the translation process that were available in older paradigms. Recent work, however, has introduced a new capability: *lexically constrained* or *guided* decoding, a modification to beam search that forces the inclusion of pre-specified words and phrases in the output. However, while theoretically sound, existing approaches have computational complexities that are either linear (Hokamp and Liu, 2017) or exponential (Anderson et al., 2017) in the number of constraints. We present an algorithm for lexically constrained decoding with a complexity of $\mathcal{O}(1)$ in the number of constraints. We demonstrate the algorithm's remarkable ability to properly place these constraints, and use it to explore the shaky relationship between model and BLEU scores. Our implementation is available as part of SOCKEYE.

1 Introduction

One appeal of the phrase-based statistical approach to machine translation (Koehn et al., 2003) was that it provided control over system output. For example, it was relatively easy to incorporate domain-specific dictionaries, or to force a translation choice for certain words. These kinds of interventions were useful in a range of settings, including interactive machine translation or domain adaptation. In the new paradigm of neural machine translation (NMT), these kinds of manual interventions are much more difficult, and a lot of time has been spent investigating how to restore them (cf. Arthur et al. (2016)).

At the same time, NMT has also provided new capabilities. One interesting recent innovation is *lexically constrained decoding*, a modification to beam search that allows the user to specify words and phrases that must appear in the system output (Figure 1). Two algorithms have been proposed for this: *grid beam search* (Hokamp and Liu, 2017, GBS) and *constrained beam search* (Anderson et al., 2017, CBS). These papers showed that these algorithms do a good job automatically placing constraints and improving results in tasks such as simulated post-editing, domain adaptation, and caption generation.

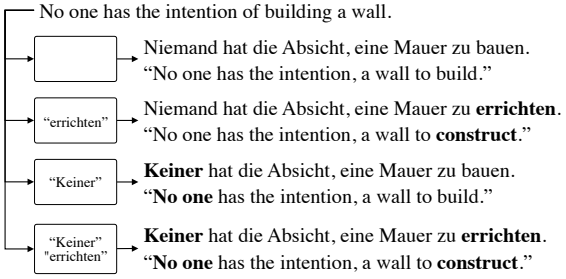


Figure 1: An example translating from English to German. The first translation is unconstrained, whereas the remaining ones have one or two constraints imposed. A word-for-word translation of the German output has been provided for the convenience of non-German speaking readers.

A downside to these algorithms is their runtime complexity: linear (GBS) or exponential (CBS) in the number of constraints. Neither paper reported decoding speeds, but the complexities alone suggest a large penalty in runtime. Beyond this, other factors of these approaches (a variable sized beam, finite-state machinery) change the decoding procedure such that it is difficult to integrate with other operations known to increase throughput, like batch decoding.

We propose and evaluate a new algorithm, *dynamic beam allocation* (DBA), that is *constant* in the number of provided constraints (Table 1). Our algorithm works by grouping together hypotheses that have met the same number of constraints into

Proceedings of NAACL-HLT 2018, pages 1314–1324
New Orleans, Louisiana, June 1 - 6, 2018.

work	complexity
Anderson et al. (2017)	$\mathcal{O}(Nk2^C)$
Hokamp and Liu (2017)	$\mathcal{O}(NkC)$
This work	$\mathcal{O}(Nk)$

Table 1: Complexity of decoding (sentence length N, beam size k, and constraint count C) with target-side constraints under various approaches.

banks (similar in spirit to the grouping of hypotheses into stacks for phrase-based decoding (Koehn et al., 2003)) and dynamically dividing a fixed-size beam across these banks at each time step. As a result, the algorithm scales easily to large constraint sets that can be created when words and phrases are expanded, for example, by sub-word processing such as BPE (Sennrich et al., 2016). We compare it to GBS and demonstrate empirically that it is significantly faster, making constrained decoding with an arbitrary number of constraints feasible with GPU-based inference. We also use the algorithm to study beam search interactions between model and metric scores, beam size, and pruning.

2 Beam Search and Grid Beam Search

Inference in statistical machine translation seeks to find the output sequence, $\hat{y}$, that maximizes the probability of a function parameterized by a model, θ, and an input sequence, x:

$$\hat{y} = \text{argmax}_{y \in \mathcal{Y}} \, p_\theta(y \mid x)$$

The space of possible translations, $\mathcal{Y}$, is the set of all sequences of words in the target language vocabulary, V_T. It is impossible to explore this entire space. Models decompose this problem into a sequence of time steps, t. At each time step, the model produces a distribution over V_T. The simplest approach to translation is therefore to run the steps of the decoder, choosing the most-probable token at each step, until either the end-of-sentence token, $\langle/\text{s}\rangle$, is generated, or some maximum output length is reached. An alternative, which explores a slightly larger portion of the search space, is beam search.

In beam search (Lowerre, 1976; Sutskever et al., 2014), the decoder maintains a *beam* of size k containing a set of active hypotheses (Algorithm 1). At each time step t, the decoder model is used to produce a distribution over the target-language vocabulary, V_T, for each of these hypotheses. This produces a large matrix of dimensions $k \times |V_T|$, that can be computed quickly with modern GPU hardware. Conceptually, a (row, column) entry (i, j) in this matrix contains the state obtained from starting from the ith state in the beam and generating the target word corresponding to the jth word of V_T. The beam for the next time step is filled by taking the states corresponding to the k-best items from this entire matrix and sorting them.

Algorithm 1 Beam search. *Inputs:* max output length N, beam size k. *Output:* highest-scoring hypothesis.

```
1: function BEAM-SEARCH(N, k)
2:     beam ← DECODER-INIT(k)
3:     for time step t in 1..N do
4:         scores = DECODER-STEP(beam)
5:         beam ← KBEST(scores)
6:     return beam[0]
7: function KBEST(scores)
8:     beam = ARGMAX_K(k, scores)
9:     return beam
```

A principal difference between beam search for phrase-based and neural MT is that in NMT, there is no recombination: each hypothesis represents a complete history, back to the first word generated. This makes it easy to record properties of the history of each hypothesis that were not possible with dynamic programming. Hokamp and Liu (2017) introduced an algorithm for forcing certain words to appear in the output called *grid beam search* (GBS). This algorithm takes a set of constraints, which are words that must appear in the output, and ensures that hypotheses have met all these constraints before they can be considered to be completed. For C constraints, this is accomplished by maintaining $C + 1$ separate beams or *banks*, $B_0, B_1, \ldots, B_C$, where B_i groups together hypotheses that have generated (or *met*) i of the constraints. Decoding proceeds as with standard beam decoding, but with the addition of bookkeeping that tracks the number of constraints met by each hypothesis, and ensures that new candidates are generated, such that each bank is filled at each time step. When beam search is complete, the hypothesis returned is the highest-scoring one in bank B_C. Conceptually, this can be thought of as adding an additional dimension to the beam, since we multiply out some base beam size b by (one plus) the number of constraints.

We note two problems with GBS:

- Decoding complexity is linear in the number of constraints: The effective beam size, $k \cdot (C + 1)$, varies with the number of constraints.
- It is impractical. The beam size changes for every sentence, whereas most decoders specify the beam size at model load time in order to optimize computation graphs, specially when running on GPUs. It also complicates beam search optimizations that increase throughput, such as batching.

Our extension, fast lexically-constrained decoding via dynamic beam allocation (DBA), addresses both of these issues. Instead of maintaining $C + 1$ beams, we maintain a single beam of size k, as with unconstrained decoding. We then dynamically allocate the slots of this beam across the constraint banks at each time step. There is still bookkeeping overhead, but this cost is constant in the number of constraints, instead of linear. The result is a practical algorithm for incorporating arbitrary target-side constraints that fits within the standard beam-decoding paradigm.

3 Dynamic Beam Allocation (DBA)

Our algorithm (Algorithm 2) is based on a small but important alteration to GBS. Instead of *multiplying* the beam by the number of constraints, we *divide*. A fixed beam size is therefore provided to the decoder, just as in standard beam search. As different sentences are processed with differing numbers of constraints, the beam is dynamically allocated to these different banks. In fact, the allocation varies not just by sentence, but across time steps in processing each individual sentence.

We need to introduce some terminology. A *word constraint* provided to the decoder is a single token in the target language vocabulary. A *phrasal constraint* is a sequence of two or more contiguous tokens. Phrasal constraints come into play when the user specifies a multi-word phrase directly (e.g., *high-ranking member*), or when a word gets broken up by subword splitting (e.g., *thou@@ ghtful*). The total number of constraints is the sum of the number of tokens across all word and phrasal constraints. It is easier for the decoder to place multiple sequential tokens in a phrasal constraint (where the permutation is fixed) compared to placing separate, independent constraints (see discussion at the end of §5), but the algorithm does not distinguish them when counting.

DBA fits nicely within standard beam decoding; we simply replace the `kbest` implementation from Algorithm 1 with one that involves a bit more bookkeeping. Instead of selecting the top-k items from the $k \times V_T$ scores matrix, the new algorithm must consider two important matters.

1. Generating a list of candidates (§3.1). Whereas the baseline beam search simply takes the top-k items from the scores matrix (a fast operation on a GPU), we now need to ensure that candidates progress through the set of provided constraints.
2. Allocating the beam across the constraint banks (§3.2). With a fixed-sized beam and an arbitrary number of constraints, we need to find an allocation strategy for dividing the beam across the constraint banks.

3.1 Generating the candidate set

We refer to Figure 2 for discussion of the algorithm. The set of candidates for the beam at time step $t + 1$ is generated from the hypotheses in the current beam at step t, which are sorted in decreasing order, with the highest-scoring hypothesis at position 1. The DECODER-STEP function of beam search generates a matrix, $scores$, where each row r corresponds to a probability distribution over all target words, expanding the hypothesis in position r in the beam. We build a set of candidates from the following items:

1. The best k tokens across all rows of $scores$ (i.e., normal top-k);
2. for each hypothesis in the beam, all unmet constraints (to ensure progress through the constraints); and
3. for each hypothesis in the beam, the single-best token (to ensure consideration of partially-completed hypotheses).

Each of these candidates is denoted by its coordinates in `scores`. The result is a set of candidates which can be grouped into banks according to how many constraints they have met, and then sorted within those banks. The new beam for timestep $t + 1$ is then built from this list according to an allocation policy (next section).

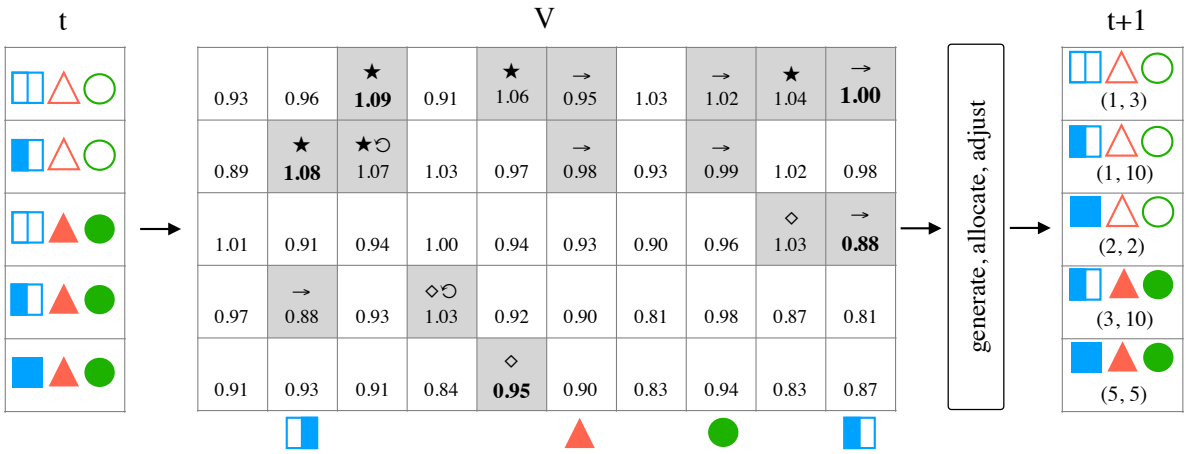


Figure 2: A single step of the constrained decoder. Along the left is the beam ($k = 5$) at time step t. The shapes in this beam represent constraints, both met (filled) and unmet (outlined). The blue square represents a phrasal constraint of length 2, which must be completed in order (left half, then right half). A step of the decoder produces a $k \times V_T$ matrix of scores. Each constraint corresponds to a single token in the vocabulary, and is marked along the bottom. Gray squares denote the set of candidates that are produced (§3.1) from the k best items (★), from extending each hypothesis with all unfilled constraints (→), and from its single-best next token (◇). Items that violate a phrasal constraint (↺) require the phrasal constraint from that hypotheses to be unwound (set to unmet). From these fifteen candidates, the beam at time step $t+1$ is filled, according to the bank allocation strategy, which here assigns one slot in the beam to each bank. The final beam includes coordinates indicating the provenance of chosen items (which are also indicated in bold in the grid).

For hypotheses partially through a phrasal constraint, special care must be taken. If a phrasal constraint has been begun, but not finished, and a token is chosen that does not match the next word of the constraint, we must reset or "unwind" those tokens in this constraint that are marked as having been met. This permits the decoder to abort the generation of a phrasal constraint, which is important in situations where a partial prefix of a phrasal constraint appears in the decoded sentence earlier than the entire phrase.

3.2 Allocating the beam

The task is to allocate a size-k beam across $C+1$ constraint banks, where C may be greater than k. We use the term *bank* to denote the portion of the beam reserved for items having met the same number of constraints (including one bank for hypotheses with zero constraints met). We use a simple allocation strategy, setting each bin size to $\lfloor k/C \rfloor$, irrespective of the timestep. Any remaining slots are assigned to the "topmost" or maximally constrained bank, C.

This may at first appear wasteful. For example, space allocated at timestep 1 to a bank representing candidates having met more than one constraint cannot be used, and similarly, for later timesteps, it seems wasteful to allocate space to bank 1. Additionally, if the number of candidates in a bank is smaller than the allocation for that bank, the beam is in danger of being underfilled. These problems are mitigated by bank adjustment (Figure 3). We provide here only a sketch of this procedure. An overfilled bank is one that has been allocated more slots than it has candidates to fill. Each such overfilled bank, in turn, gives its extra allotments to banks that have more candidates than slots, looking first to its immediate neighbors, and moving outward until it has distributed all of its extra slots. In this way, the beam is filled, up to the minimum of the beam size or the number of candidates.

3.3 Finishing

Hypotheses are not allowed to generate the end-of-sentence token, $\langle/\mathrm{s}\rangle$, unless they have met all of their constraints. When beam search is finished, the highest-scoring completed item is returned.

4 Experimental Setup

Our experiments were done using SOCKEYE (Hieber et al., 2017). We used an English–German model trained on the complete WMT'17 training corpora (Bojar et al., 2017), which we pre-

Algorithm 2 k-best extraction with DBA. *Inputs:* A $k \times |V_T|$ matrix of model states.

```
function KBEST-DBA(beam, scores)
    constraints ← [hyp.constraint for hyp in beam]
    candidates ← [(i, j, constraints[i].add(j)] for i, j in ARGMAX_K(k, scores)      ▷ Top overall k
    for 1 ≤ h ≤ k do                                                                  ▷ Go over current beam
        for all w ∈ V_T that are unmet constraints for beam[h] do                     ▷ Expand new constraints
            candidates.append( (h, w, constraints[h].add(w) ) )
        w = ARGMAX(scores[h, :])
        candidates.append( (h, w,  constraints[h].add(w)) )                           ▷ Best single word
    selected ← ALLOCATE(candidates, k)
    newBeam ← [candidates[i] for i in selected]
    return newBeam
```

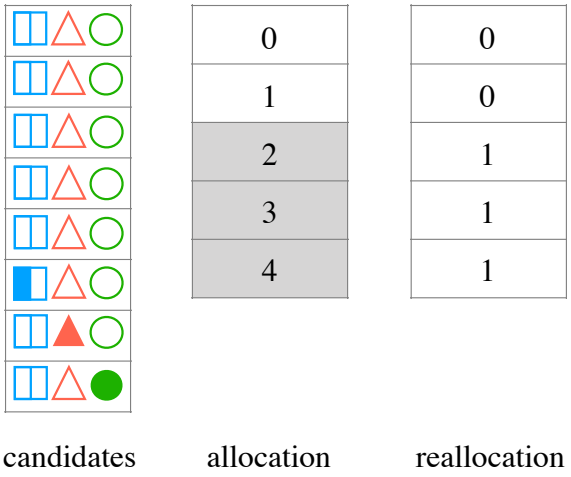


Figure 3: Beam reallocation for $k = 5$ with 4 constraints at timestep t. There are eight candidates, each having met only 0 or 1 constraint. The allocation policy gives one slot of the beam to each bank. However, there are no candidates for banks 2–4 (greyed), so their slots are redistributed to banks 0 and 1.

processed with the Moses tokenizer (preserving case) and with a joint byte-pair-encoded vocabulary with 32k merge operations (Sennrich et al., 2016). The model was a 4 layer RNN with attention. We trained using the Adam optimizer with a batch size of 80 until cross-entropy on the development data (newstest2016) stopped increasing for 10 consecutive iterations.

For decoding, we normalize completed hypotheses (those that have generated ⟨/s⟩), dividing the cumulative sentence score by the number of words. Unless otherwise noted, we apply threshold pruning to the beam, removing hypotheses whose log probability is not within 20 compared to the best completed hypothesis. This pruning is applied to all hypotheses, whether they are complete or not. (We explore the importance of this pruning in §6.3). Decoding stops when either all hypotheses still on the beam are completed or the maximum length, N, is reached. All experiments were run on a single a Volta P100 GPU. No ensembling or batching were used.

For experiments, we used the newstest2014 English–German test set (the developer version, with 2,737 sentences). All BLEU scores are computed on detokenized output using SACREBLEU (Post, 2018),[1] and are thus directly comparable to scores reported in the WMT evaluations.

5 Validation Experiment

We center our exploration of DBA by experimenting with constraints randomly selected from the references. We extract five sets of constraints: from one to four randomly selected words from the reference (`rand1` to `rand4`), and a randomly selected four-word phrase (`phr4`). We then apply BPE to these sets, which often yields a much larger number of token constraints. Statistics about these extracted phrases can be found in Table 2.

We simulate the GBS baseline within our framework. After applying BPE, We group together translations with the same number of constraints, C, and then translate them as a group, with the beam set for that group set to $b(C + 1)$, where b is the "base beam" parameter. We use $b = 10$ as reported in Hokamp et al., but also try smaller values of $b = 5$ and 1. Finally, we disable beam adjustment (§3.2), so that the space allocated to each constraint bank does not change.

Table 4 compares speeds and BLEU scores (in the legend) as a function of the number of post-

[1]The signature is BLEU+case.mixed+lang.en-de+numrefs.1+smooth.exp+test.wmt14+tok.13a+v.1.2.6

num	rand1	rand2	rand3	rand4	phr4
1	2,182	0	0	0	0
2	548	3,430	0	0	0
3	516	1,488	4,074	0	0
4	272	1,128	2,316	4,492	4,388
5	150	765	1,860	3,275	2,890
6	30	306	1,218	2,520	2,646
7	42	133	805	1,736	1,967
8	0	112	488	1,096	1,280
9	0	36	171	702	720
10	0	10	140	400	430
11+	0	22	189	417	575
total	3,726	7,477	11,205	14,885	14,926
mean	1.36	2.73	4.09	5.43	5.45

Table 2: Histogram of the number of token constraints for some constraint sets after applying BPE (model trained with 32k merge operations). *mean* denotes the mean number of constraints per sentence in the 2,737-sentence test set.

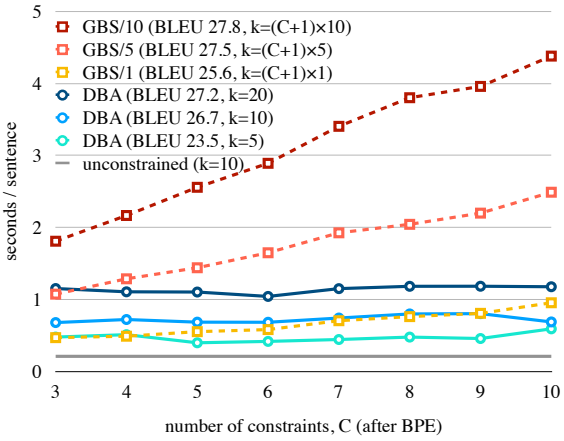


Figure 4: Running time (seconds / sentence, lower is better) as a function of the number of constraints, C (after applying BPE) on the `rand3` dataset. The unconstrained baselines have BLEU scores of 22.3, 22.3, and 22.1 for $k = 5, 10$, and 20, respectively.

BPE constraints for the `rand3` dataset. We plot all points for which there were at least 10 sentences. The times are decoding only, and exclude model loading and other setup. The linear trend in C is clear for GBS, as is the constant trend for DBA. In terms of absolute runtimes, DBA improves considerably over GBS, whose beam sizes quickly become quite large with a non-unit base beam size. On the Tesla V100 GPU, DBA ($k = 10$) takes about 0.6 seconds/sentence, regardless of the number of constraints.[2] This is about 3x slower than unconstrained decoding.

It is difficult to compare these algorithms exactly because of GBS's variable beam size. An important comparison is that between DBA ($k = 10$) and GBS/1 (base beam of 1). A beam of $k = 10$ is a common setting for decoding in general, and GBS/1 has a beam size of $k \geq 10$ for $C \geq 9$. At this setting, DBA finds better translations (BLEU 26.7 vs. 25.6) with the same runtime and with a fixed, instead of variable-sized, beam.

We note that the bank adjustment correction of the DBA algorithm allows it to work when $C >= k$. The DBA ($k = 5$) plot demonstrates this, while still finding a way to increase the BLEU score over GBS (23.5 vs. 22.3). However, while possible, low k relative to C reduces the observed improvement considerably. Looking at Figure 5 across different constraint sets, we can get a better feel for this relationship. DBA is still always able to meet the constraints even with a beam size of 5, but the quality suffers. This should not be too surprising; correctly placing independent constraints is at least as hard as finding their correct permutation, which is exponential in the number of independent constraints. But it is remarkable that the only failure to beat the baseline in terms of BLEU is when the algorithm is tasked with placing four random constraints (before BPE) with a beam size of 5. In contrast, DBA never has any trouble placing phrasal constraints (dashed lines).

[2]On a K80, it is about 1.4 seconds / sentence

6 Analysis

6.1 Placement

It's possible that the BLEU gains result from a boost in n-gram counts due to the mere presence of the reference constraints in the output, as opposed to their correct placement. This appears not to be the case. Experience examining the outputs shows its uncanny ability to sensibly place constrained words and phrases. Figure 6 contains some examples from translating a German sentence into English, manually identifying interesting phrases in the target, choosing paraphrases of those words, and then decoding with them as constraints. Note that the word weak, which doesn't fit in the semantics of the reference, is placed haphazardly.

We also confirm this correct placement quantitatively by comparing the location of the first word of each constraint in (a) the reference and (b) the output of the constrained decoder, represented as a percentage of the respective sentence lengths (Figure 7). We would not expect these numbers to

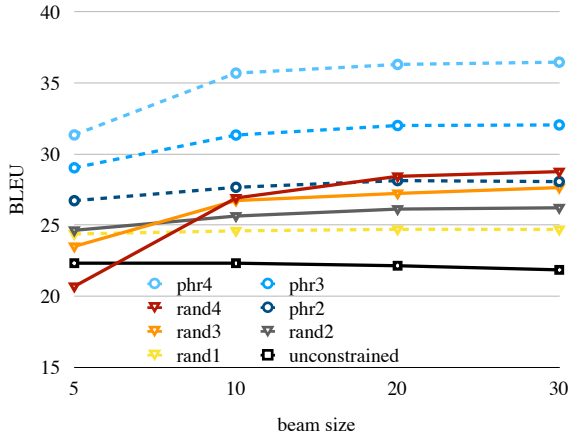


Figure 5: BLEU score as a function of beam size under DBA. All constraint sets improve as the beam gets larger (recall that the actual number of constraints increases after BPE and varies by sentence). `rand4` performs under the unconstrained baseline if the beam is too low.

	0	3	5	10	20	30
none	24.4	24.5	24.5	24.4	24.5	24.4
rand1	25.2	25.1	25.2	25.6	25.5	25.3
rand2	26.0	25.3	25.6	26.1	26.7	26.4
rand3	26.5	24.7	24.9	25.7	26.9	27.2
rand4	26.2	23.7	23.9	24.6	26.0	26.9
phr4	35.1	33.5	33.5	34.0	35.0	35.9

Table 3: BLEU scores decoding with a beam size of 10. Runtimes for unpruned systems (column 0) are nearly twice those of the other columns. But it is only at large thresholds that BLEU scores are higher than the unpruned setting.

be perfectly matched, but the strong correlation is pretty apparent (Pearson's $r = 0.82$). Together, Figures 6 and 7 provide confidence that DBA is intelligently placing the constraints.

6.2 Reference Aversion

The inference procedure in SOCKEYE maximizes the length-normalized version of the sentence's log probability. While there is no explicit training towards the metric, BLEU, modeling in machine translation assumes that better model scores correlate with better BLEU scores. However, a general repeated observation from the NMT literature is the disconnect between model score and BLEU score. For example, work has shown that opening up the beam to let the decoder find better hypotheses results in lower BLEU score (Koehn and Knowles, 2017), even as the model score rises. The phenomenon is not well understood, but it seems that NMT models have learned to travel a path straight towards their goal; as soon as they get off this path, they get lost, and can no longer function (Ott et al., 2018).

Another way to look at this problem is to ask what the neural model thinks of the references. Scoring against complete references is easy with NMT (Sennrich, 2017), but lexically-constrained decoding allows us to investigate this in finer-grained detail by including just portions of the references. We observe that forcing the decoder to include even a single word from the reference imposes a cost in model score that is inversely correlated with BLEU score, and that this grows with the number of constraints that are added (Figure 8). The NMT system seems quite averse to the references, even in small pieces, and even while it improves the BLEU score. At the same time, the hypotheses it finds in this reduced space are still good, and become better as the beam is enlargened (Figure 5). This provides a complementary finding to that of Koehn and Knowles (2017): in that setting, higher model scores found by a larger beam produce lower BLEU scores; here, lower model scores are associated with significantly higher BLEU scores.

6.3 Effects of Pruning

In the results reported above, we used a pruning threshold of 20, meaning that any hypothesis whose log probability is not within 20 of the best completed hypothesis is removed from the beam. This pruning threshold is far greater than those explored in other papers; for example, Wu et al. (2016) use 3. However, we observed two things: first, without pruning, running time for constrained decoding is nearly doubled. This increased runtime applies to both DBA and GBS in Figure 4. Second, low pruning thresholds are harmful to BLEU scores (Table 3). It is only once the thresholds reach 20 that the algorithm is able to find better BLEU scores compared to the unpruned baseline (column 0).

6.4 Garbage Generation

Why is the algorithm so slow without pruning? One might suspect that the outputs are longer, but mean output length with all constraint sets is roughly the same. The reason turns out to be that the the decoder never quits before the maximum

constraint	score	output
source		*Einer soll ein hochrangiges Mitglied aus Berlin gewesen sein .*
no constraints	-0.217	One should have been a high-ranking member from Berlin .
is said to	-0.551	One is said to have been a high-ranking member from Berlin .
of them	-0.577	One of them was to be a high-ranking member from Berlin .
participant	-0.766	One should have been a high-ranking participant from Berlin .
is thought to	-0.792	One is thought to have been a high-ranking member from Berlin .
considered	-0.967	One is considered to have been a high-ranking member from Berlin .
Hamburg	-1.165	One should have been a high-ranking member from Hamburg .
powerful	-1.360	One is to have been a powerful member from Berlin .
powerful, *is said to*	-1.496	One is said to have been a powerful member from Berlin .
powerful, *is said to*, *participant*	-1.988	One is said to have been a powerful participant from Berlin .
weak	-1.431	One weak point was to have been a high-ranking member from Berlin .
reference		*One is said to have been a high-ranking member from Berlin.*

Figure 6: Example demonstrating the correct placement of manually chosen constraints (beam size 10). The unnatural placement of the constraint *weak* demonstrates what the model does when forced to include a word that is not a semantic fit.

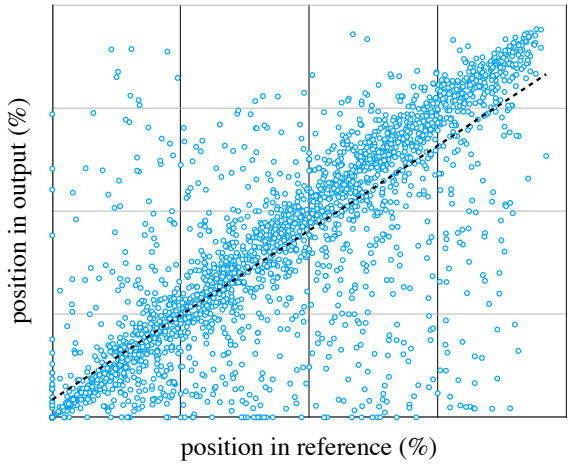


Figure 7: Location of the first word of each constraint from `phr3` in the reference versus the constrained output (Pearson's $r = 0.82$). DBA correctly places its constraints, even though no source word or alignment information is provided.

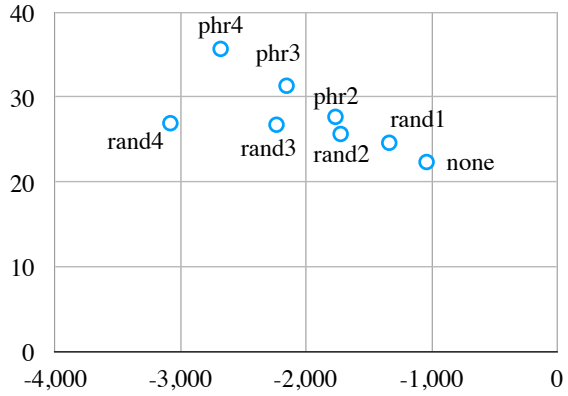


Figure 8: BLEU score as a function of model score (summed over the corpus). The reference model score is -4,396.

timestep, N. SOCKEYE's stopping criterium is to wait until all hypotheses on the beam are finished. Without pruning, the decoder generates a finished hypotheses, but continues on until the maximum timestep N, populating the rest of the beam with low-cost garbage. An example can be found in Figure 9. This may be an example of the well-attested phenomenon where NMT systems become unhinged from the source sentence, switching into "language model" mode and generating high-probable output with no end. But strangely, this doesn't seem to affect the best hypotheses, but only the rest of the beam. This seems to be more evidence of reference aversion, where the decoder, having been forced into a place it doesn't like, does not know how to generate good competing hypotheses.

An alternative to pruning is *early stopping*, which is to stop when the first complete hypothesis is generated. In our experiments, while this did fix the problem of increasing runtimes, the BLEU scores were lower.

6.5 Conclusions

By setting a large pruning threshold, we produced large speedups over GBS, and demonstrated a constant overhead in the number of constraints. Compared to GBS, our DBA algorithm makes lexically constrained decoding possible, requiring less than half a second on average on a Volta GPU with a 4-layer RNN.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	…	41
-0.51	⟨s⟩	Er	und	Kerr	lieben	einander	noch	immer	,	betonte	die	36-Jährige	.	⟨/s⟩		
-0.52	⟨s⟩	Er	und	Kerr	lieben	einander	noch	immer	,	betonte	der	36-Jährige	.	⟨/s⟩		
-0.56	⟨s⟩	Er	und	Kerr	lieben	einander	noch	immer	,	betonte	die	36-jährige	.	⟨/s⟩		
-0.57	⟨s⟩	Er	und	Kerr	lieben	einander	noch	immer	,	betonte	den	36-jährigen	.	⟨/s⟩		
-25.11	⟨s⟩	Er	und	Kerr	lieben	sich	weiterhin	einander	,	betonte	die	36-Jährige	.		…	
-27.92	⟨s⟩	Er	und	Kerr	lieben	sich	weiterhin	einander	,	betonte	die	36-Jährige	.		…	

Figure 9: The sentence *He and Kerr still love each other , emphasised the 36-year-old .* translated with the constraint *noch immer , betonte* (BPE removed for readability). The first column is the log probability, which is normalized only for finished hypotheses. The decoder completes a few hypotheses well before the maximum timestep, but then fills the lower beam with garbage until forced to stop.

7 Related Work

Hokamp and Liu (2017) was novel in that it allowed the specification of arbitrary target-side words as hard constraints, implemented entirely as a restructuring of beam search, and without reference to the source. A related approach was that of Anderson et al. (2017), who extended beam search with a finite state machine whose states marked completed subsets of the set of constraints, at an exponential cost in the number of constraints.

Lexically-constrained decoding also generalizes prefix decoding (Knowles and Koehn, 2016; Wuebker et al., 2016), since the ⟨s⟩ symbol can easily be included as the first word of a constraint.

Our work here has not explored where to get lexical constraints, but considering that question naturally brings to mind attempts to improve NMT by using lexicons and phrase tables (Arthur et al., 2016; Tang et al., 2016).

Finally, another approach which shares the hard-decision made by lexically constrained decoding is the *placeholder approach* (Crego et al., 2016), wherein identifiable elements in the input are transformed to masks during preprocessing, and then replaced with their original source-language strings during postprocessing.

8 Summary

Neural machine translation removes many of the knobs from phrase-based MT that provided fine-grained control over system output. Lexically-constrained decoding restores one of these tools, providing a powerful and interesting way to influence NMT output. It requires only the specification of the target-side constraints; without any source word or alignment information, it correctly places the constraints. Although we have only tested it here with RNNs, the code works without modification with other architectures generate target-side words one-by-one, such as the Transformer (Vaswani et al., 2017).

This paper has introduced a fast and practical solution. Building on previous approaches, constrained decoding with DBA does away with linear and exponential complexity (in the number of constraints), imposing only a constant overhead. On a Volta GPU, lexically-constrained decoding with DBA is practical, requiring about 0.6 seconds per sentence on average even with 10+ constraints, well within the realm of feasibility even for applications with strict lattency requirements, like post-editing tasks. We imagine that there are further optimizations in reach that could improve this even further.

Acknowledgments We thank Felix Hieber for valuable discussions.

References

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2017. Guided open vocabulary image captioning with constrained beam search. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 936–945. Association for Computational Linguistics.

Philip Arthur, Graham Neubig, and Satoshi Nakamura. 2016. Incorporating discrete translation lexicons into neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1557–1567. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 conference on machine translation (wmt17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214. Association for Computational Linguistics.

Josep Crego, Jungi Kim, Guillaume Klein, Anabel Rebollo, Kathy Yang, Jean Senellart, Egor Akhanov, Patrice Brunelle, Aurelien Coquard, Yongchao Deng, Satoshi Enoue, Chiyo Geiss, Joshua Johanson, Ardas Khalsa, Raoum Khiari, Byeongil Ko, Catherine Kobus, Jean Lorieux, Leidiana Martins, Dang-Chuan Nguyen, Alexandra Priori, Thomas Riccardi, Natalia Segal, Christophe Servan, Cyril Tiquet, Bo Wang, Jin Yang, Dakun Zhang, Jing Zhou, and Peter Zoldan. 2016. Systran's pure neural machine translation systems. *CoRR*.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2017. Sockeye: A toolkit for neural machine translation. *Computing Research Repository*, abs/1712.05690.

Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1546. Association for Computational Linguistics.

Howard Johnson, Joel Martin, George Foster, and Roland Kuhn. 2007. Improving translation quality by discarding most of the phrasetable. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Rebecca Knowles and Philipp Koehn. 2016. Neural interactive translation prediction. In *Proceedings of the Association for Machine Translation in the Americas*, pages 107–120.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39. Association for Computational Linguistics.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.

Pierre Lison and Jörg Tiedemann. 2016. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Bruce T. Lowerre. 1976. *The HARPY speech recognition system*. Ph.D. thesis, Carnegie-Mellon University.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60. Association for Computational Linguistics.

Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Analyzing uncertainty in neural machine translation. *CoRR*, abs/1803.00047.

Matt Post. 2018. A call for clarity in reporting bleu scores. *Computing Research Repository*, abs/1804.08771.

Rico Sennrich. 2017. How grammatical is character-level neural machine translation? assessing mt quality with contrastive translation pairs. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 376–382. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Yaohua Tang, Fandong Meng, Zhengdong Lu, Hang Li, and Philip L. H. Yu. 2016. Neural machine translation with external phrase memory. *CoRR*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*.

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

Joern Wuebker, Spence Green, John DeNero, Sasa Hasan, and Minh-Thang Luong. 2016. Models and inference for prefix-constrained machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75. Association for Computational Linguistics.

A Appendix: Failed Experiments

The main contribution of this paper is a fast, practical algorithmic improvement to lexically-constrained decoding. While we did not attempt to corroborate the experiments in interactive translation and domain adaptation experiments reported in (Hokamp and Liu, 2017), the gains discovered there only become more salient with this faster algorithm. We did try to apply lexical constraints in a few other settings, but without success. In the spirit of open scientific inquiry and reporting, we provide here a brief report on these experiments.

A.1 Automatic Constraint Selection

Our validation experiments (§5) demonstrate the large potential gains in BLEU score when including random phrases from the reference. Even including just a single random word from the reference increased BLEU score by a point; another point was gained from including two random words, and a four-word phrase yielded 10+ point gains. Only about 18% of these random unigrams were present in the unconstrained output (less for longer n-grams). This raises the question of whether we can automatically identify words that are likely to be in the reference and include them as constraints, in order to improve translation quality.

In order to do that, we first extracted a phrase table using Moses and filtered it with the significance testing approach proposed by Johnson et al. (2007) in order to keep only high quality phrases. We then selected the best phrase for each input sentence according to different criteria (longest phrase, higher significance, highest probability, combination of those). Unfortunately, adding such phrases as constraints when translating WMT or IWSLT data did not help.

A.2 Name Entity Translation

One topic that has received attention in the literature is the tendency of NMT systems to do poorly with rare words, and in particular, named entities (e.g., Arthur et al. (2016)). BPE helps address this by breaking down words into pieces and allowing all words to be represented in the decoder's vocabulary. But even with BPE, many times the correct translation does not follow any pattern, even at the subword level. This is specially true for named entities; e.g. "Aachen" in German is translated as "Aquisgrán" in Spanish or "Aix-la-Chapelle" in French, which bear little resemblance to the original form except for the starting letter. Named entities (NEs) also have the advantage that in several languages they are not inflected; therefore a simple lookup in a dictionary, if available, should produce the correct translation.

To the best of our knowledge, there is no publicly available parallel corpus of named entities. In order to create one, we downloaded the OpenSubtitles database (Lison and Tiedemann, 2016) for German and English and applied a simple method for extracting named entity correspondences. We first tagged the source and target sides with the Stanford NER system (Manning et al., 2014). We then selected a subset of the tags that were produced by both systems ("Person", "Location" and "Organization") and selected those sentences where they appeared only once for each language. From those we extracted the corresponding NEs, selecting the most frequent target side at the corpus level as the translation of a given source NE.

Given such a dictionary, we can add the translation of a NE found in new sentences to translate as decoding constraints. It didn't help. Manual inspection showed that the dictionary extracted with this simple method was still too noisy. We think that a manual, high-quality dictionary may provide a way to produce improvements.

Guiding Neural Machine Translation with Retrieved Translation Pieces

Jingyi Zhang[1,2], **Masao Utiyama**[1], **Eiichro Sumita**[1]
Graham Neubig[3,2], **Satoshi Nakamura**[2]
[1]National Institute of Information and Communications Technology, Japan
[2]Graduate School of Information Science, Nara Institute of Science and Technology, Japan
[3]Language Technologies Institute, Carnegie Mellon University, USA
jingyizhang/mutiyama/eiichiro.sumita@nict.go.jp
gneubig@cs.cmu.edu, s-nakamura@is.naist.jp

Abstract

One of the difficulties of neural machine translation (NMT) is the recall and appropriate translation of low-frequency words or phrases. In this paper, we propose a simple, fast, and effective method for recalling previously seen translation examples and incorporating them into the NMT decoding process. Specifically, for an input sentence, we use a search engine to retrieve sentence pairs whose source sides are similar with the input sentence, and then collect n-grams that are both in the retrieved target sentences and aligned with words that match in the source sentences, which we call "translation pieces". We compute pseudo-probabilities for each retrieved sentence based on similarities between the input sentence and the retrieved source sentences, and use these to weight the retrieved translation pieces. Finally, an existing NMT model is used to translate the input sentence, with an additional bonus given to outputs that contain the collected translation pieces. We show our method improves NMT translation results up to 6 BLEU points on three narrow domain translation tasks where repetitiveness of the target sentences is particularly salient. It also causes little increase in the translation time, and compares favorably to another alternative retrieval-based method with respect to accuracy, speed, and simplicity of implementation.

1 Introduction

Neural machine translation (NMT) (Bahdanau et al., 2014; Sennrich et al., 2016a; Wang et al., 2017b) is now the state-of-the-art in machine translation, due to its ability to be trained end-to-end on large parallel corpora and capture complex parameterized functions that generalize across a variety of syntactic and semantic phenomena. However, it has also been noted that compared to alternatives such as phrase-based translation (Koehn et al., 2003), NMT has trouble with low-frequency words or phrases (Arthur et al., 2016; Kaiser et al., 2017), and also generalizing across domains (Koehn and Knowles, 2017). A number of methods have been proposed to ameliorate these problems, including methods that incorporate symbolic knowledge such as discrete translation lexicons (Arthur et al., 2016; He et al., 2016; Chatterjee et al., 2017) and phrase tables (Zhang et al., 2017; Tang et al., 2016; Dahlmann et al., 2017), adjust model structures to be more conducive to generalization (Nguyen and Chiang, 2017), or incorporate additional information about domain (Wang et al., 2017a) or topic (Zhang et al., 2016) in translation models.

In particular, one paradigm of interest is recent work that augments NMT using *retrieval*-based models, retrieving sentence pairs from the training corpus that are most similar to the sentence that we want to translate, and then using these to bias the NMT model.[1] These methods – reminiscent of translation memory (Utiyama et al., 2011) or example-based translation (Nagao, 1984; Grefenstette, 1999) – are effective because they augment the parametric NMT model with a non-parametric translation memory that allows for increased capacity to measure features of the target technical terms or domain-specific words. Currently there are two main approaches to doing so. Li et al. (2016) and Farajian et al. (2017) use the retrieved sentence pairs to fine tune the parameters of the NMT model which is pre-trained on the whole training corpus. Gu et al. (2017) uses the retrieved sentence pairs as additional inputs to the NMT model to help NMT in translating the input sen-

[1]Note that there are existing retrieval-based methods for phrase-based and hierarchical phrase-based translation (Lopez, 2007; Germann, 2015). However, these methods do not improve translation quality but rather aim to improve the efficiency of the translation models.

Proceedings of NAACL-HLT 2018, pages 1325–1335
New Orleans, Louisiana, June 1 - 6, 2018.

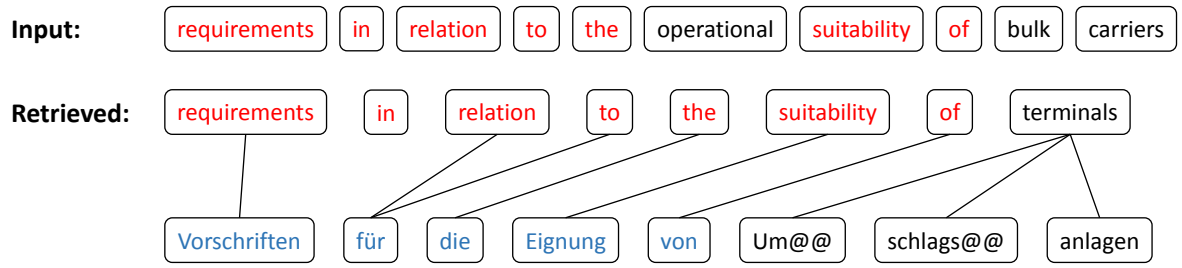


Figure 1: A word-aligned sentence pair retrieved for an input sentence. Red words are unedited words obtained by computing the edit distance between the input sentence and the retrieved source sentence. The blue part of the retrieved target sentence is collected as translation pieces for the input sentence. The target word "Umschlagsanlagen" is split into "Um@@", "schlags@@" and "anlagen" by byte pair encoding.

tence. While both of these paradigms have been proven effective, they both add significant complexity and computational/memory cost to the decoding process, and also to the training procedure. The first requires the running of several training iterations and rolling back of the model, which is costly at test time, and the second requires entirely changing the model structure which requires training the model separately, and also increases test-time computational cost by adding additional encoders.

In this paper, we propose a simple and efficient model for using retrieved sentence pairs to guide an existing NMT model at test time. Specifically, the model collects n-grams occurring in the retrieved target sentences that also match words that overlap between the input and retrieved source sentences, which we will refer to as "translation pieces" (e.g., in Figure 1, the blue part of the retrieved target sentence is collected as translation pieces for the input sentence). The method then calculates a pseudo-probability score for each of the retrieved example sentence pairs and weights the translation pieces according to this value. Finally, we up-weight NMT outputs that contain the collected translation pieces. Unlike the previous methods, this requires no change of the underlying NMT model and no updating of the NMT parameters, making it both simple and efficient to apply at test time.

We show our method improved NMT translation results up to 6 BLEU points on three translation tasks and caused little increase in the translation time. Further, we find that accuracies are comparable with the model of Gu et al. (2017), despite being significantly simpler to implement and faster at test time.

2 Attentional NMT

Our baseline NMT model is similar to the attentional model of Bahdanau et al. (2014), which includes an encoder, a decoder and an attention (alignment) model. Given a source sentence $X = \{x_1, ..., x_L\}$, the encoder learns an annotation $h_i = \left[\vec{h}_i; \overleftarrow{h}_i\right]$ for x_i using a bi-directional recurrent neural network.

The decoder generates the target translation from left to right. The probability of generating next word y_t is,[2]

$$P_{NMT}\left(y_t|y_1^{t-1}, X\right) = softmax\left(g\left(y_{t-1}, z_t, c_t\right)\right) \quad (1)$$

where z_t is a decoding state for time step t, computed by,

$$z_t = f\left(z_{t-1}, y_{t-1}, c_t\right) \quad (2)$$

c_t is a source representation for time t, calculated as,

$$c_t = \sum_{i=1}^{L} \alpha_{t,i} \cdot h_i \quad (3)$$

where $\alpha_{t,i}$ scores how well the inputs around position i and the output at position t match, computed as,

$$\alpha_{t,i} = \frac{\exp\left(a\left(z_{t-1}, h_i\right)\right)}{\sum_{j=1}^{L} \exp\left(a\left(z_{t-1}, h_j\right)\right)} \quad (4)$$

The standard decoding algorithm for NMT is beam search. That is, at each time step t, we keep n-best hypotheses. The probability of a complete

[2] g, f and a in Equation 1, 2 and 4 are nonlinear, potentially multi-layered, functions.

hypothesis is computed as,

$$\log P_{NMT}\left(Y|X\right) = \sum_{t=1}^{|Y|} \log P_{NMT}\left(y_t|y_1^{t-1}, X\right) \tag{5}$$

Finally, the translation score is normalized by sentence length to avoid too short outputs.

$$\log S_{NMT}\left(Y|X\right) = \frac{\log P_{NMT}\left(Y|X\right)}{|Y|} \tag{6}$$

3 Guiding NMT with Translation Pieces

This section describes our approach, which mainly consists of two parts:

1. retrieving candidate translation pieces from a parallel corpus for the new source sentence that we want to translate, and then
2. using the collected translation pieces to guide an existing NMT model while translating this new sentence.

At training time, we first prepare the parallel corpus that will form our database used in the retrieval of the translation pieces. Conceivably, it could be possible to use a different corpus for translation piece retrieval and NMT training, for example when using a separate corpus for domain adaptation, but for simplicity in this work we use the same corpus that was used in NMT training. As pre-processing, we use an off-the-shelf word aligner to learn word alignments for the parallel training corpus.

3.1 Retrieving Translation Pieces

At test time we are given an input sentence X. For this X, we first use the off-the-shelf search engine Lucene to search the word-aligned parallel training corpus and retrieve M source sentences $\{X^m : 1 \leq m \leq M\}$ that are similar to X. Y^m indicates the target sentence that corresponds to source sentence X^m and $\mathcal{A}^m$ is word alignments between X^m and Y^m.

For each retrieved source sentence X^m, we compute its edit distance with X as $d\left(X, X^m\right)$ using dynamic programming. We record the unedited words in X^m as $\mathcal{W}^m$, and also note the words in the target sentence Y^m that correspond to source words in $\mathcal{W}^m$, which we can presume are words that will be more likely to appear in the translated sentence for X. According to Algorithm 1, we collect n-grams (up to 4-grams) from the retrieved target sentence Y^m as possible translation pieces G_X^m for X, using word-level alignments to select n-grams that are related to X and discard n-grams that are not related to X. The final translation pieces G_X collected for X are computed as,[3]

n-grams	G_X^m
Vorschriften für die Eignung	Yes
die Eignung von	Yes
von Um@@ schlags@@ anlagen	No
Um@@ schlags@@ anlagen	No

Table 1: Examples of the collected translation pieces.

$$G_X = \bigcup_{m=1}^{M} G_X^m \tag{7}$$

Table 1 shows a few n-gram examples contained in the retrieved target sentence in Figure 1 and whether they are included in G_X^m or not. Because the retrieved source sentence in Figure 1 is highly similar with the input sentence, the translation pieces collected from its target side are highly likely to be correct translation pieces of the input sentence. However, when a retrieved source sentence is not very similar with the input sentence (e.g. only one or two words match), the translation pieces collected from its target side will be less likely to be correct translation pieces for the input sentence.

We compute a score for each $u \in G_X$ to measure how likely it is a correct translation piece for X based on sentence similarity between the retrieved source sentences and the input sentence as following,

$$\begin{aligned} &S\left(u, X, \bigcup_{m=1}^{M} \{(X^m, G_X^m)\}\right) \\ &= \max_{1 \leq m \leq M \wedge u \in G_X^m} simi\left(X, X^m\right) \end{aligned} \tag{8}$$

where $simi\left(X, X^m\right)$ is the sentence similarity computed as following (Gu et al., 2017),

$$simi\left(X, X^m\right) = 1 - \frac{d\left(X, X^m\right)}{\max\left(|X|, |X^m|\right)} \tag{9}$$

[3]Note that the extracted translation pieces are target phrases, but the target words contained in one extracted translation piece may be aligned to discontiguous source words, which is different from how phrase-based translation extracts phrase-based translation rules.

Algorithm 1 Collecting Translation Pieces

Require: $X = x_1^L$, $X^m = k_1^{L'}$, $Y^m = v_1^{L''}$, $\mathcal{A}^m$, $\mathcal{W}^m$
Ensure: G_X^m
  $G_X^m = \emptyset$
  for $i = 1$ to L'' **do**
    for $j = i$ to L'' **do**
      if $j - i = 4$ **then**
        break
      if $\exists p : (p, j) \in \mathcal{A}^m \wedge p \notin \mathcal{W}^m$ **then**
        break
      add v_i^j into G_X^m

Algorithm 2 Guiding NMT by Translation Pieces

Require: Output layer $\log P_{NMT}\left(y_t|y_1^{t-1}, X\right)$, $\mathcal{L}_X, \mathcal{D}_X$
Ensure: Updated output layer
  for u in $\mathcal{L}_X$ **do**
    $\log P_{NMT}\left(u|y_1^{t-1}, X\right) += \lambda \mathcal{D}_X\left(u\right)$
    for $i = 1$ to 3 **do**
      if $t - i < 1$ **then**
        break
      if $y_{t-i}^{t-1}, u \notin \mathcal{D}_X$ **then**
        break
      $\log P_{NMT}\left(u|y_1^{t-1}, X\right) += \lambda \mathcal{D}_X\left(y_{t-i}^{t-1}, u\right)$

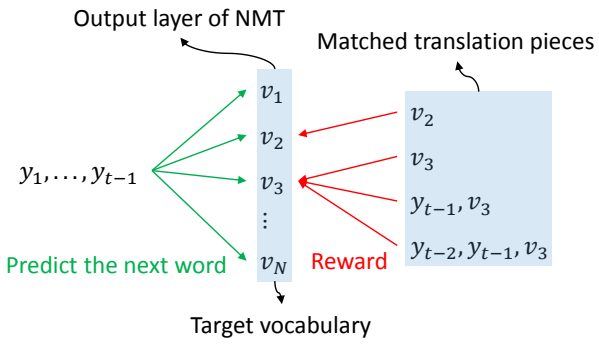


Figure 2: A simple demonstration of adding rewards for matched translation pieces into the NMT output layer.

3.2 Guiding NMT with Retrieved Translation Pieces

In the next phase, we use our NMT system to translate the input sentence. Inspired by Stahlberg et al. (2017) which rewards n-grams from syntactic translation lattices during NMT decoding, we add an additional reward for n-grams that occur in the collected translation pieces. That is, as shown in Figure 2, at each time step t, we update the probabilities over the output vocabulary and increase the probabilities of those that result in matched n-grams according to

$$\begin{aligned} &\log S_{NMT\_updated}\left(y_t|y_1^{t-1}, X\right) \\ &= \log P_{NMT}\left(y_t|y_1^{t-1}, X\right) + \\ &\lambda \sum_{n=1}^{4} \delta\left(y_{t-n+1}^t, X, \bigcup_{m=1}^{M}\{(X^m, G_X^m)\}\right), \end{aligned} \tag{10}$$

where λ can be tuned on the development set and $\delta\left(\cdot\right)$ is computed as Equation 8 if $y_{t-n+1}^t \in G_X$, otherwise $\delta\left(\cdot\right) = 0$.

To implement our method, we use a dictionary $\mathcal{D}_X$ to store translation pieces G_X and their scores for each input sentence X. At each time step t, we update the output layer probabilities by checking $\mathcal{D}_X$. However, it is inefficient to traverse all target words in the vocabulary and check whether they belong to G_X or not, because the vocabulary size is large. Instead, we only traverse target words that belong to G_X and update the corresponding output probabilities as shown in Algorithm 2. Here, $\mathcal{L}_X$ is a list that stores 1-grams contained in G_X.[4]

As we can see, our method only up-weights NMT outputs that match the retrieved translation pieces in the NMT output layer. In contrast, Li et al. (2016) and Farajian et al. (2017) use the retrieved sentence pairs to run additional training iterations and fine tune the NMT parameters for each input sentence; Gu et al. (2017) runs the NMT model for each retrieved sentence pair to obtain the NMT encoding and decoding information of the retrieved sentences as key-value memory to guide NMT for translating the new input sentence. Compared to their methods, our method adds little computational/memory cost and is simple to implement.

4 Experiment

4.1 Settings

Following Gu et al. (2017), we use version 3.0 of the JRC-Acquis corpus for our translation experiments. The JRC-Acquis corpus contains the total body of European Union (EU) law applicable in the EU Member States. It can be used as a narrow domain to test the effectiveness of our proposed method. We did translation experiments on three

[4]Note that our method does not introduce new states during decoding, because the output layer probabilities are simply updated based on history words and the next word.

		en-de		en-fr		en-es	
		BLEU	METEOR	BLEU	METEOR	BLEU	METEOR
dev	NMT	44.08	36.69	57.26	43.51	55.76	42.53
	Ours	50.81	39.50	62.60	45.83	60.51	44.58
test	NMT	43.76	36.57	57.67	43.66	55.78	42.55
	Ours	50.15	39.18	63.27	46.24	60.54	44.64

Table 2: Translation results.

	en-de	en-fr	en-es
TRAIN	674K	665K	663K
DEV	1,636	1,733	1,662
TEST	1,689	1,710	1,696
Average Length	31	29	29

Table 3: Data sets. The last line is the average length of English sentences.

		en-de	en-fr	en-es
dev	NMT	1.000	0.990	0.997
	Ours	1.005	0.991	1.001
test	NMT	0.995	0.990	0.990
	Ours	1.004	0.989	0.993

Table 5: Ratio of translation length to reference length.

directions: English-to-German (en-de), English-to-French (en-fr) and English-to-Spanish (en-es).

We cleaned the data by removing repeated sentences and used the `train-truecaser.perl` script from Moses (Koehn et al., 2007) to truecase the corpus. Then we selected 2000 sentence pairs as development and test sets, respectively. The rest was used as the training set. We removed sentences longer than 80 and 100 from the training and development/test sets respectively. The final numbers of sentence pairs contained in the training, development and test sets are shown in Table 3.[5] We applied *byte pair encoding* (Sennrich et al., 2016b) and set the vocabulary size to be 20K.

For translation piece collection, we use GIZA++ (Och and Ney, 2003) and the `grow-diag-final-and` heuristic (Koehn et al., 2003) to obtain symmetric word alignments for the training set.

We trained an attentional NMT model as our baseline system. The settings for NMT are shown in Table 4. We also compared our method with the search engine guided NMT model (SGNMT, Gu et al. (2017)) in Section 4.5.

Word embedding	512
GRU dimension	1024
Optimizer	adam
Initial learning rate	0.0001
Beam size	5

Table 4: NMT settings.

[5]We put the datasets used in our experiments on Github https://github.com/jingyiz/Data-sampled-preprocessed

For each input sentence, we retrieved 100 sentence pairs from the training set using Lucene as our preliminary setting. We analyze the influence of the retrieval size in Section 4.4. The weights of translation pieces used in Equation 10 are tuned on the development set for different language pairs, resulting in weights of 1.5 for en-de and en-fr, and a weight of 1 for en-es.

4.2 Results

Table 2 shows the main experimental results. We can see that our method outperformed the baseline NMT system up to 6 BLEU points. As large BLEU gains in neural MT can also often be attributed to changes in output length, we examined the length (Table 5) and found that it did not influence the translation length significantly.

In addition, it is of interest whether how well the retrieved sentences match the input influences the search results. We measure the similarity between a test sentence X and the training corpus D_{train} by computing the sentence similarities between X and the retrieved source sentences as

$$simi\left(X, D_{train}\right) = \max_{1 \leq m \leq M} simi\left(X, X^m\right). \tag{11}$$

The similarity between the test set D_{test} and the training corpus D_{train} is measured as,

$$simi\left(D_{test}, D_{train}\right) = \frac{\sum_{X \in D_{test}} simi\left(X, D_{train}\right)}{|D_{test}|} \tag{12}$$

Our analysis demonstrated that, expectedly, the performance of our method is highly influenced by the similarity between the test set and the training set. We divided sentences in the test set into two

	whole	half-H	half-L
en-de	0.56	0.80	0.32
en-fr	0.57	0.81	0.33
en-es	0.57	0.81	0.32

Table 6: Similarities between the training set and the whole/divided test sets.

		whole	half-H	half-L
en-de	NMT	43.76	60.93	32.25
	Ours	50.15	73.26	34.28
en-fr	NMT	57.67	72.64	47.38
	Ours	63.27	82.76	49.81
en-es	NMT	55.78	69.32	46.26
	Ours	60.54	78.37	47.93

Table 7: Translation results (BLEU) for the whole/divided test sets.

parts: half has higher similarities with the training corpus (half-H) and half has lower similarities with the training corpus (half-L). Table 6 shows the similarity between the training corpus and the whole/divided test sets. Table 7 shows translation results for the whole/divided test sets. As we can see, NMT generally achieved better BLEU scores for half-H and our method improved BLEU scores for half-H much more significantly than for half-L, which shows our method can be quite useful for narrow domains where similar sentences can be found.

We also tried our method on WMT 2017 English-to-German News translation task. However, we did not achieve significant improvements over the baseline attentional NMT model, likely because the test set and the training set for the WMT task have a relatively low similarity as shown in Table 8 and hence few useful translation pieces can be retrieved for our method. In contrast, the JRC-Acquis corpus provides test sentences that have much higher similarities with the training set, i.e., much more and longer translation pieces exist.

To demonstrate how the retrieved translation pieces help NMT to generate appropriate outputs, Figure 3 shows an input sentence with reference, the retrieved sentence pair with the highest sentence similarity and outputs by different systems for this input sentence with detailed scores: log NMT probabilities for each target word in T_1 and T_2; scores for matched translation pieces contained in T_1 and T_2. As we can see, NMT assigns higher probabilities to the incorrect translation T_1, even though the retrieved sentence pair whose source side is very similar with the input sentence was used for NMT training.

	WMT		JRC-Acquis	
Similarity	Sent	Percent	Sent	Percent
$[0, 0.1)$	0	0%	4	0.2%
$[0.1, 0.2)$	415	13.8%	141	8.3%
$[0.2, 0.3)$	1399	46.5%	238	14.0%
$[0.3, 0.4)$	740	24.6%	194	11.4%
$[0.4, 0.5)$	281	9.3%	154	9.1%
$[0.5, 0.6)$	113	3.7%	156	9.2%
$[0.6, 0.7)$	29	0.9%	157	9.2%
$[0.7, 0.8)$	10	0.3%	156	9.2%
$[0.8, 0.9)$	10	0.3%	252	14.9%
$[0.9, 1)$	0	0%	237	14.0%
1	7	0.2%	0	0%

Table 8: Statistics for similarities between each test sentence and the training set as computed by Equation 11 for the WMT 2017 en-de task (3004 sentences) and our JRC-Acquis en-de task (1689 sentences).

		en-de	en-fr	en-es
dev	NMT	44.08	57.26	55.76
	Ours	50.81	62.60	60.51
	1/0 reward	47.70	61.15	58.92
test	NMT	43.76	57.67	55.78
	Ours	50.15	63.27	60.54
	1/0 reward	47.13	62.14	58.66

Table 9: Translation results (BLEU) of 1/0 reward.

However, T_2 contains more and longer translation pieces with higher scores. The five translation pieces contained only in T_2 are collected from the retrieved sentence pair shown in Figure 3, which has high sentence similarity with the input sentence. The three translation pieces contained only in T_1 are also translation pieces collected for the input sentence, but have lower scores, because they are collected from sentence pairs with lower similarities with the input sentence. This shows that computing scores for translation pieces based on sentence similarities is important for the performance of our method. If we assign score 1 to all translation pieces contained in G_X, i.e., use 1/0 reward for translation pieces and non-translation pieces, then the performance of our method decreased significantly as shown in Table 9, but still outperformed the NMT baseline significantly.

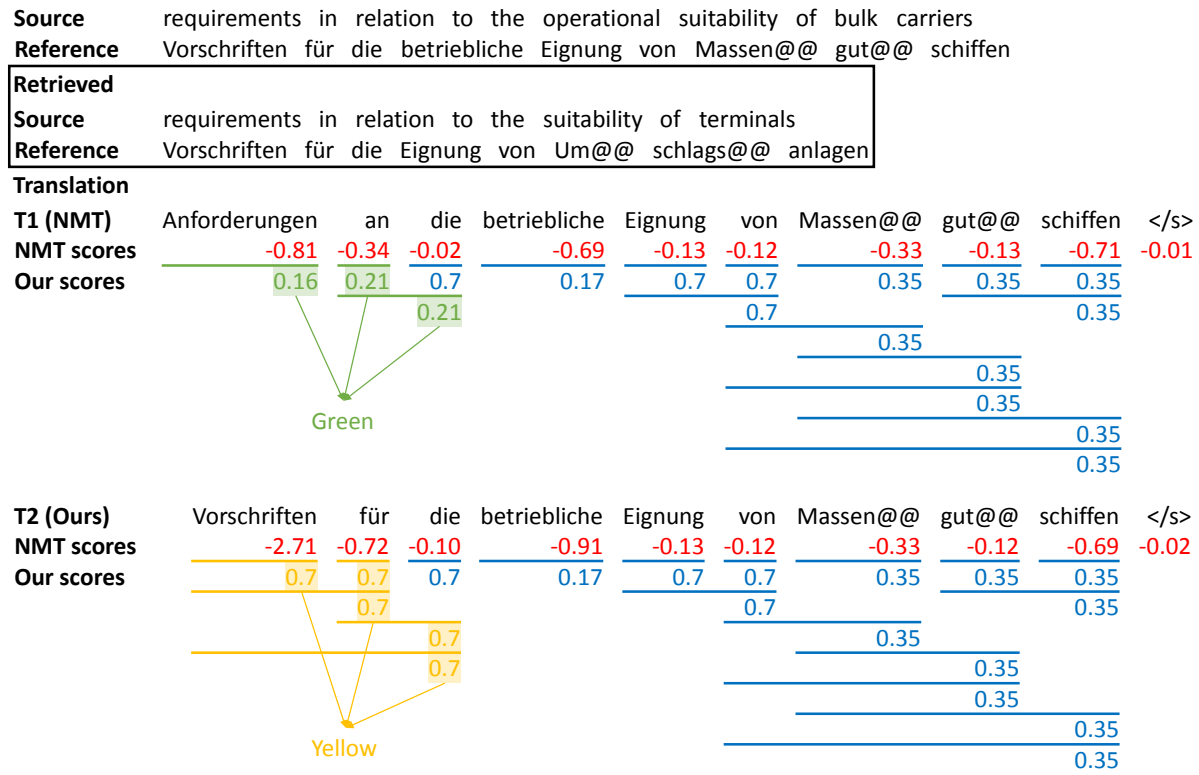


Figure 3: Translation examples. Red scores are log NMT probabilities. Green, yellow and blue scores are scores of matched translation pieces contained only in T_1, contained only in T_2, contained in both T_1 and T_2, respectively.

γ		0	1	2	5	10	20	50	100
en-de	NMT	5834	3193	1988	1196	717	370	157	75
	Ours	5843	5433	3153	1690	933	458	193	86
	Ratio (Ours/NMT)	1.00	1.70	1.58	1.41	1.30	1.23	1.22	1.14
en-fr	NMT	6983	3743	2637	1563	812	493	210	118
	Ours	7058	5443	3584	1919	968	581	214	134
	Ratio (Ours/NMT)	1.01	1.45	1.35	1.22	1.19	1.17	1.01	1.13
en-es	NMT	6500	3430	2292	1346	772	437	182	95
	Ours	6516	4589	2970	1652	895	500	196	97
	Ratio (Ours/NMT)	1.00	1.33	1.29	1.22	1.15	1.14	1.07	1.02

Table 10: $Count_\gamma$

4.3 Infrequent n-grams

The basic idea of our method is rewarding n-grams that occur in the training set during NMT decoding. We found our method is especially useful to help the translation for infrequent n-grams. First, we count how many times a target n-gram u occurs in the training set D_{train} as,

$$Occur\left(u\right) = \left|\left\{Y : \langle X, Y\rangle \in D_{train} \wedge u \in uniq\left(Y\right)\right\}\right| \tag{13}$$

where $uniq\left(Y\right)$ is the set of uniq n-grams (up to 4-grams) contained in Y.

Given system outputs $\left\{Z^k : 1 \leq k \leq K\right\}$ for the test set $\left\{X^k : 1 \leq k \leq K\right\}$ with reference $\left\{Y^k : 1 \leq k \leq K\right\}$, we count the number of correctly translated n-grams that occur γ times in the training set as,

$$Count_\gamma = \sum_{k=1}^{K} \left|\psi\left(\gamma, Z^k, Y^k\right)\right| \tag{14}$$

where

$$\psi\left(\gamma, Z^k, Y^k\right) = \left\{u : u \in \left(uniq\left(Z^k\right) \cap uniq\left(Y^k\right)\right) \wedge Occur\left(u\right) = \gamma\right\} \tag{15}$$

Table 10 shows $Count_\gamma$ for different system outputs. As we can see, our method helped little for the translation of n-grams that do not occur

	en-de	en-fr	en-es
Base NMT decoding	0.215	0.224	0.227
Search engine retrieval	0.016	0.017	0.016
TP collection	0.521	0.522	0.520
Our NMT decoding	0.306	0.287	0.289

Table 11: Translation time (seconds).

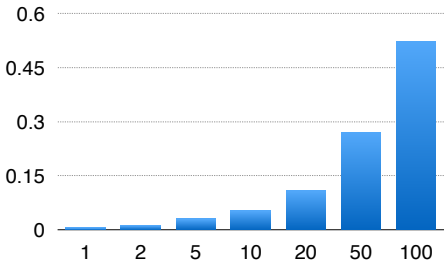


Figure 4: Translation piece collection time (seconds) with different search engine retrieval sizes.

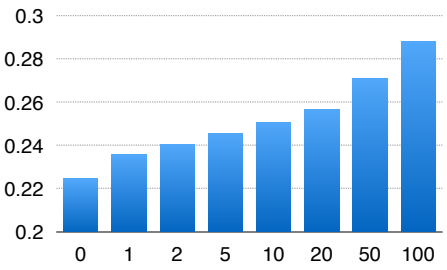


Figure 5: NMT decoding time (seconds) with different search engine retrieval sizes.

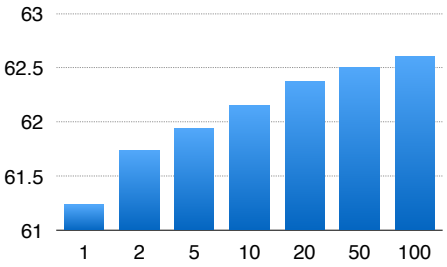


Figure 6: Translation results (BLEU) with different search engine retrieval sizes.

in the training set, which is reasonable because we only reward n-grams that occur in the training set. However, our method helped significantly for the translation of n-grams that do occur in the training set but are infrequent (occur less than 5 times). As the frequency of n-grams increases, the improvement caused by our method decreased. We analyze that the reason why our method is especially helpful for infrequent n-grams is that NMT is trained on the whole training corpus for maximum likelihood and tends to generate more frequent n-grams while our method computes scores for the collected translation pieces based on sentence similarities and does not prefer more frequent n-grams.

4.4 Computational Considerations

Our method only collects translation pieces to help NMT for translating a new sentence and does not influence the training process of NMT. Therefore, our method does not increase the NMT training time. Table 11 shows the average time needed for translating one input sentence in the development set in our experiments. The search engine retrieval and translation piece (TP) collection time is computed on a 3.47GHz Intel Xeon X5690 machine using one CPU. The NMT decoding time is computed using one GPU GeForce GTX 1080.

As we can see, the search engine retrieval time is negligible and the increase of NMT decoding time caused by our method is also small. However, collecting translation pieces needed considerable time, although our implementation was in Python and could potentially be significantly faster in a more efficient programming language. The translation piece collection step mainly consists of two parts: computing the edit distances between the input sentence and the retrieved source sentences using dynamic programming with time complexity $O\left(n^2\right)$; collecting translation pieces using Algorithm 1 with time complexity $O\left(4n\right)$.

We changed the size of sentence pairs retrieved by the search engine and analyze its influence on translation performance and time. Figure 4, 5 and 6 show the translation piece collection time, the NMT decoding time and translation BLEU scores with different search engine retrieval sizes for the en-fr task. As we can see, as the number of retrieved sentences decreased, the time needed by translation piece collection decreased significantly, the translation performance decreased much less significantly and the NMT decoding time is further reduced. In our experiments, 10 is a good setting for the retrieval size, which gave significant BLEU score improvements and caused little increase in the total translation time compared to the NMT baseline.

		en-de	en-fr	en-es
dev	$\text{NMT}_{reported}$	44.94	58.95	50.54
	$\text{SGNMT}_{reported}$	49.26	64.16	**57.62**
	NMT	45.18	59.08	50.71
	Ours	**50.61**	**65.03**	57.49
test	$\text{NMT}_{reported}$	43.98	59.42	50.48
	$\text{SGNMT}_{reported}$	48.80	64.60	**57.27**
	NMT	44.21	59.43	50.61
	Ours	**50.36**	**65.69**	57.11

Table 12: Comparison with SGNMT.

4.5 Comparison with SGNMT

We compared our method with the search engine guided NMT (SGNMT) model (Gu et al., 2017). We got their preprocessed datasets and tested our method on their datasets, in order to fairly compare our method with their reported BLEU scores.[6] Table 12 shows the results of their method and our method with the same settings for the baseline NMT system. As we can see, our method generally outperformed their method on the three translation tasks.

Considering the computational complexity, their method also performs search engine retrieval for each input sentence and computes the edit distance between the input sentence and the retrieved source sentences as our method. In addition, their method runs the NMT model for each retrieved sentence pair to obtain the NMT encoding and decoding information of the retrieved sentences as key-value memory to guide the NMT model for translating the real input sentence, which changes the NMT model structure and increases both the training-time and test-time computational cost. Specifically, at test time, running the NMT model for one retrieved sentence pair costs the same time as translating the retrieved source sentence with beam size 1. Therefore, as the number of the retrieved sentence pairs increases to the beam size of the baseline NMT model, their method doubles the translation time.

5 Conclusion

This paper presents a simple and effective method that retrieves translation pieces to guide NMT for narrow domains. We first exploit a search engine to retrieve sentence pairs whose source sides are similar with the input sentence, from which we collect and weight translation pieces for the input sentence based on word-level alignments and sentence similarities. Then we use an existing NMT model to translate this input sentence and give an additional bonus to outputs that contain the collected translation pieces. We show our method improved NMT translation results up to 6 BLEU points on three narrow domain translation tasks, caused little increase in the translation time, and compared favorably to another alternative retrieval-based method with respect to accuracy, speed, and simplicity of implementation.

[6] Only BLEU scores are reported in their paper.

Acknowledgments

We thank Jiatao Gu for providing their preprocessed datasets in Section 4.5.

References

Philip Arthur, Graham Neubig, and Satoshi Nakamura. 2016. Incorporating discrete translation lexicons into neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 1557–1567. https://aclweb.org/anthology/D16-1162.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* https://arxiv.org/pdf/1409.0473.pdf.

Rajen Chatterjee, Matteo Negri, Marco Turchi, Marcello Federico, Lucia Specia, and Frédéric Blain. 2017. Guiding neural machine translation decoding with external knowledge. In *Proceedings of the Second Conference on Machine Translation*. pages 157–168. http://www.aclweb.org/anthology/W17-4716.

Leonard Dahlmann, Evgeny Matusov, Pavel Petrushkov, and Shahram Khadivi. 2017. Neural machine translation leveraging phrase-based models in a hybrid search. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 1411–1420. https://www.aclweb.org/anthology/D17-1148.

M. Amin Farajian, Marco Turchi, Matteo Negri, and Marcello Federico. 2017. Multi-domain neural machine translation through unsupervised adaptation. In *Proceedings of the Second Conference on Machine Translation*. pages 127–137. http://www.aclweb.org/anthology/W/W17/W17-4713.pdf.

Ulrich Germann. 2015. Sampling phrase tables for the moses statistical machine translation system. *The Prague Bulletin of Mathematical Linguistics* 104(1):39–50. https://ufal.mff.cuni.cz/pbml/104/art-germann.pdf.

Gregory Grefenstette. 1999. The world wide web as a resource for example-based machine translation tasks. In *Proceedings of the ASLIB Conference on Translating and the Computer*. volume 21. http://www.mt-archive.info/Aslib-1999-Grefenstette.pdf.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor OK Li. 2017. Search engine guided non-parametric neural machine translation. *arXiv preprint arXiv:1705.07267* https://arxiv.org/pdf/1705.07267.pdf.

Wei He, Zhongjun He, Hua Wu, and Haifeng Wang. 2016. Improved neural machine translation with smt features. In *AAAI*. pages 151–157. https://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12189/11577.

Łukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. 2017. Learning to remember rare events. *arXiv preprint arXiv:1703.03129* https://arxiv.org/pdf/1703.03129.pdf.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*. pages 177–180. http://www.aclweb.org/anthology/P07-2045.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*. pages 28–39. http://www.aclweb.org/anthology/W17-3204.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*. pages 48–54. http://www.aclweb.org/anthology/N/N03/N03-1017.pdf.

Xiaoqing Li, Jiajun Zhang, and Chengqing Zong. 2016. One sentence one model for neural machine translation. *arXiv preprint arXiv:1609.06490* https://arxiv.org/pdf/1609.06490.pdf.

Adam Lopez. 2007. Hierarchical phrase-based translation with suffix arrays. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. pages 976–985. http://www.aclweb.org/anthology/D/D07/D07-1104.

Makoto Nagao. 1984. A framework of a mechanical translation between japanese and english by analogy principle. *Artificial and human intelligence* pages 351–354. http://www.mt-archive.info/Nagao-1984.pdf.

Toan Q Nguyen and David Chiang. 2017. Improving lexical choice in neural machine translation. *arXiv preprint arXiv:1710.01329* https://arxiv.org/pdf/1710.01329.pdf.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational linguistics* 29(1):19–51. http://www.aclweb.org/anthology/J/J03/J03-1002.pdf.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Edinburgh neural machine translation systems for WMT 16. In *Proceedings of the First Conference on Machine Translation*. pages 371–376. http://www.aclweb.org/anthology/W/W16/W16-2323.pdf.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. pages 1715–1725. http://www.aclweb.org/anthology/P/P16/P16-1162.pdf.

Felix Stahlberg, Adrià de Gispert, Eva Hasler, and Bill Byrne. 2017. Neural machine translation by minimising the Bayes-risk with respect to syntactic translation lattices. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. pages 362–368. http://www.aclweb.org/anthology/E/E17/E17-2058.pdf.

Yaohua Tang, Fandong Meng, Zhengdong Lu, Hang Li, and Philip LH Yu. 2016. Neural machine translation with external phrase memory. *arXiv preprint arXiv:1606.01792* https://arxiv.org/pdf/1606.01792.pdf.

Masao Utiyama, Graham Neubig, Takashi Onishi, and Eiichiro Sumita. 2011. Searching translation memories for paraphrases. In *Machine Translation Summit*. volume 13, pages 325–331. http://mt-archive.info/MTS-2011-Utiyama.pdf.

Rui Wang, Andrew Finch, Masao Utiyama, and Eiichiro Sumita. 2017a. Sentence embedding for neural machine translation domain adaptation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. pages 560–566. http://aclweb.org/anthology/P17-2089.

Yuguang Wang, Shanbo Cheng, Liyang Jiang, Jiajun Yang, Wei Chen, Muze Li, Lin Shi, Yanfeng Wang, and Hongtao Yang. 2017b. Sogou neural machine translation systems for wmt17. In *Proceedings of the Second Conference on Machine Translation*. pages 410–415. http://www.aclweb.org/anthology/W17-4742.

Jiacheng Zhang, Yang Liu, Huanbo Luan, Jingfang Xu, and Maosong Sun. 2017. Prior knowledge integration for neural machine translation using posterior regularization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. pages 1514–1523. http://www.aclweb.org/anthology/P/P17/P17-1139.pdf.

Jian Zhang, Liangyou Li, Andy Way, and Qun Liu. 2016. Topic-informed neural machine translation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. pages 1807–1817. http://aclweb.org/anthology/C16-1170.

Handling Homographs in Neural Machine Translation

Frederick Liu[*†], **Han Lu**[*‡], **Graham Neubig**
Language Technologies Institute
Carnegie Mellon University
{fliu1,hlu2,gneubig}@cs.cmu.edu

Abstract

Homographs, words with different meanings but the same surface form, have long caused difficulty for machine translation systems, as it is difficult to select the correct translation based on the context. However, with the advent of neural machine translation (NMT) systems, which can theoretically take into account global sentential context, one may hypothesize that this problem has been alleviated. In this paper, we first provide empirical evidence that existing NMT systems in fact still have significant problems in properly translating ambiguous words. We then proceed to describe methods, inspired by the word sense disambiguation literature, that model the context of the input word with context-aware word embeddings that help to differentiate the word sense before feeding it into the encoder. Experiments on three language pairs demonstrate that such models improve the performance of NMT systems both in terms of BLEU score and in the accuracy of translating homographs.[1]

1 Introduction

Neural machine translation (NMT; Sutskever et al. (2014); Bahdanau et al. (2015), §2), a method for MT that performs translation in an end-to-end fashion using neural networks, is quickly becoming the de-facto standard in MT applications due to its impressive empirical results. One of the drivers behind these results is the ability of NMT to capture long-distance context using recurrent neural networks in both the encoder, which takes the input and turns it into a continuous-space representation, and the decoder, which tracks the target-sentence state, deciding which word to output next. As a result of this ability to capture long-distance dependencies, NMT has achieved great improvements in a number of areas that have bedeviled traditional methods such as phrase-based MT (PBMT; Koehn et al. (2003)), including agreement and long-distance syntactic dependencies (Neubig et al., 2015; Bentivogli et al., 2016).

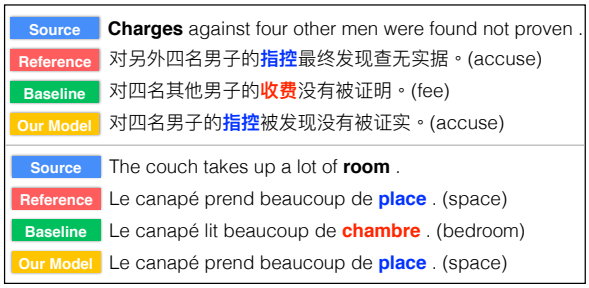


Figure 1: Homographs where the baseline system makes mistakes (red words) but our proposed system incorporating a more direct representation of context achieves the correct translation (blue words). Definitions of corresponding blue and red words are in parenthesis.

One other phenomenon that was poorly handled by PBMT was homographs – words that have the same surface form but multiple senses. As a result, PBMT systems required specific separate modules to incorporate long-term context, performing word-sense (Carpuat and Wu, 2007b; Pu et al., 2017) or phrase-sense (Carpuat and Wu, 2007a) disambiguation to improve their handling of these phenomena. Thus, we may wonder: do NMT systems suffer from the same problems when translating homographs? Or are the recurrent nets applied in the encoding step, and the strong language model in the decoding step enough to alleviate all problems of word sense ambiguity?

In §3 we first attempt to answer this question quantitatively by examining the word translation

[*]Equal contribution.
[†]Now at Snap Inc.
[‡]Now at Google
[1]Code for our translation models is available at https://goo.gl/oaiqoT

Proceedings of NAACL-HLT 2018, pages 1336–1345
New Orleans, Louisiana, June 1 - 6, 2018.

accuracy of a baseline NMT system as a function of the number of senses that each word has. Results demonstrate that standard NMT systems make a significant number of errors on homographs, a few of which are shown in Fig. 1.

With this result in hand, we propose a method for more directly capturing contextual information that may help disambiguate difficult-to-translate homographs. Specifically, we learn from neural models for word sense disambiguation (Kalchbrenner et al., 2014; Iyyer et al., 2015; Kågebäck and Salomonsson, 2016; Yuan et al., 2016; Šuster et al., 2016), examining three methods inspired by this literature (§4). In order to incorporate this information into NMT, we examine two methods: gating the word-embeddings in the model (similarly to Choi et al. (2017)), and concatenating the context-aware representation to the word embedding (§5).

To evaluate the effectiveness of our method, we compare our context-aware models with a strong baseline (Luong et al., 2015) on the English-German, English-French, and English-Chinese WMT dataset. We show that our proposed model outperforms the baseline in the overall BLEU score across three different language pairs. Quantitative analysis demonstrates that our model performs better on translating homographs. Lastly, we show sample translations of the baseline system and our proposed model.

2 Neural Machine Translation

We follow the global-general-attention NMT architecture with input-feeding proposed by Luong et al. (2015), which we will briefly summarize here. The neural network models the conditional distribution over translations $Y = (y_1, y_2, \ldots, y_m)$ given a sentence in source language $X = (x_1, x_2, \ldots x_n)$ as $P(Y|X)$. A NMT system consists of an encoder that summarizes the source sentence X as a vector representation $\boldsymbol{h}$, and a decoder that generates a target word at each time step conditioned on both $\boldsymbol{h}$ and previous words. The conditional distribution is optimized with cross-entropy loss at each decoder output.

The encoder is usually a uni-directional or bi-directional RNN that reads the input sentence word by word. In the more standard bi-directional case, before being read by the RNN unit, each word in X is mapped to an embedding in continuous vector space by a function f_e.

$$f_e(x_t) = \boldsymbol{M_e}^\top \cdot \mathbf{1}(x_t) \tag{1}$$

$\boldsymbol{M_e} \in \mathcal{R}^{|V_s| \times d}$ is a matrix that maps a one-hot representation of x_t, $\mathbf{1}(x_t)$ to a d-dimensional vector space, and V_s is the source vocabulary. We call the word embedding computed this way Lookup embedding. The word embeddings are then read by a bi-directional RNN

$$\overrightarrow{\boldsymbol{h}}_t = \overrightarrow{\text{RNN}}_e(\overrightarrow{\boldsymbol{h}}_{t-1}, f_e(x_t)) \tag{2}$$

$$\overleftarrow{\boldsymbol{h}}_t = \overleftarrow{\text{RNN}}_e(\overleftarrow{\boldsymbol{h}}_{t+1}, f_e(x_t)) \tag{3}$$

After being read by both RNNs we can compute the actual hidden state at step t, $\boldsymbol{h}_t = [\overrightarrow{\boldsymbol{h}}_t; \overleftarrow{\boldsymbol{h}}_t]$, and the encoder summarized representation $\boldsymbol{h} = \boldsymbol{h}_n$. The recurrent units $\overrightarrow{\text{RNN}}_e$ and $\overleftarrow{\text{RNN}}_e$ are usually either LSTMs (Hochreiter and Schmidhuber, 1997) or GRUs (Chung et al., 2014).

The decoder is a uni-directional RNN that decodes the tth target word conditioned on (1) previous decoder hidden state $\boldsymbol{g}_{t-1}$, (2) previous word y_{t-1} , and (3) the weighted sum of encoder hidden states $\boldsymbol{a}_t$. The decoder maintains the tth hidden state g_t as follows,

$$\boldsymbol{g}_t = \overrightarrow{\text{RNN}}_d(\boldsymbol{g}_{t-1}, f_d(y_{t-1}), \boldsymbol{a}_t) \tag{4}$$

Again, $\overrightarrow{\text{RNN}}_d$ is either LSTM or GRU, and f_d is a mapping function in target language space.

The general attention mechanism for computing the weighted encoder hidden states $\boldsymbol{a}_t$ first computes the similarity between $\boldsymbol{g}_{t-1}$ and $\boldsymbol{h}_{t'}$ for $t' = 1, 2, \ldots, n$.

$$\text{score}(\boldsymbol{g}_{t-1}, \boldsymbol{h}_{t'}) = \boldsymbol{g}_{t-1} \boldsymbol{W}_{att} \boldsymbol{h}_{t'}^\top \tag{5}$$

The similarities are then normalized through a softmax layer , which results in the weights for encoder hidden states.

$$\alpha_{t,t'} = \frac{\exp(\text{score}(\boldsymbol{g}_{t-1}, \boldsymbol{h}_{t'}))}{\sum_{k=1}^{n} \exp(\text{score}(\boldsymbol{g}_{t-1}, \boldsymbol{h}_k))} \tag{6}$$

We can then compute $\boldsymbol{a}_t$ as follows,

$$\boldsymbol{a}_t = \sum_{k=1}^{n} \alpha_{t,k} \boldsymbol{h}_k \tag{7}$$

Finally, we compute the distribution over y_t as,

$$\hat{\boldsymbol{g}}_t = \tanh(\boldsymbol{W}_1[\boldsymbol{g}_t; \boldsymbol{a}_t]) \tag{8}$$

$$p(y_t|y_{<t}, X) = \text{softmax}(\boldsymbol{W}_2 \hat{\boldsymbol{g}}_t) \tag{9}$$

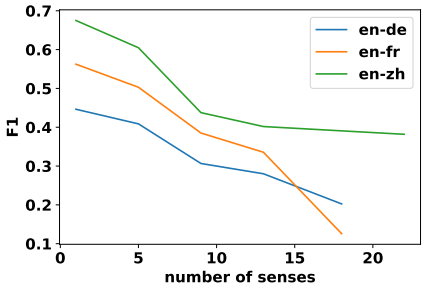


Figure 2: Translation performance of words with different numbers of senses.

3 NMT's Problems with Homographs

As described in Eqs. (2) and (3), NMT models encode the words using recurrent encoders, theoretically endowing them with the ability to handle homographs through global sentential context. However, despite the fact that they have this ability, our qualitative observation of NMT results revealed a significant number of ambiguous words being translated incorrectly, casting doubt on whether the standard NMT setup is able to appropriately learn parameters that disambiguate these word choices.

To demonstrate this more concretely, in Fig. 2 we show the translation accuracy of an NMT system with respect to words of varying levels of ambiguity. Specifically, we use the best baseline NMT system to translate three different language pairs from WMT test set (detailed in §6) and plot the F1-score of word translations by the number of senses that they have. The number of senses for a word is acquired from the Cambridge English dictionary,[2] after excluding stop words.[3]

We evaluate the translation performance of words in the source side by aligning them to the target side using `fast-align` (Dyer et al., 2013). The aligner outputs a set of target words to which the source words aligns for both the reference translation and the model translations. F1 score is calculated between the two sets of words.

After acquiring the F1 score for each word, we bucket the F1 scores by the number of senses, and plot the average score of four consecutive buckets as shown in Fig. 2. As we can see from the results, the F1 score for words decreases as the number of senses increases for three different language pairs. This demonstrates that the translation performance of current NMT systems on words with more senses is significantly decreased from that for words with fewer senses. From this result, it is evident that modern NMT architectures are not enough to resolve the problem of homographs on their own. The result corresponds to the findings in prior work (Rios et al., 2017).

[2] http://dictionary.cambridge.org/us/dictionary/english/

[3] We use the stop word list from NLTK (Bird et al., 2009).

4 Neural Word Sense Disambiguation

Word sense disambiguation (WSD) is the task of resolving the ambiguity of homographs (Ng and Lee, 1996; Mihalcea and Faruque, 2004; Zhong and Ng, 2010; Di Marco and Navigli, 2013; Chen et al., 2014; Camacho-Collados et al., 2015), and we hypothesize that by learning from these models we can improve the ability of the NMT model to choose the correct translation for these ambiguous words. Recent research tackles this problem with neural models and has shown state-of-the art results on WSD datasets (Kågebäck and Salomonsson, 2016; Yuan et al., 2016). In this section, we will summarize three methods for WSD which we will further utilize as three different *context networks* to improve NMT.

Neural bag-of-words (NBOW) Kalchbrenner et al. (2014); Iyyer et al. (2015) have shown success by representing full sentences with a context vector, which is the average of the Lookup embeddings of the input sequence

$$\boldsymbol{c}_t = \frac{1}{n}\sum_{k=1}^{n} \boldsymbol{M}_c^\top \mathbf{1}(x_k) \tag{10}$$

This is a simple way to model sentences, but has the potential to capture the global topic of the sentence in a straightforward and coherent way. However, in this case, the context vector would be the same for every word in the input sequence.

Bi-directional LSTM (BiLSTM) Kågebäck and Salomonsson (2016) leveraged a bi-directional LSTM that learns a context vector for the target word in the input sequence and predicts the word sense with a multi-layer perceptron. Specifically, we can compute the context vector c_t for tth word similarly to bi-directional encoder as follows,

$$\overrightarrow{\boldsymbol{c}}_t = \overrightarrow{\text{RNN}}_c(\overrightarrow{\boldsymbol{c}}_{t-1}, f_c(x_t)) \tag{11}$$

$$\overleftarrow{\boldsymbol{c}}_t = \overleftarrow{\text{RNN}}_c(\overleftarrow{\boldsymbol{c}}_{t+1}, f_c(x_t)) \tag{12}$$

$$\boldsymbol{c}_t = [\overrightarrow{\boldsymbol{c}}_t; \overleftarrow{\boldsymbol{c}}_t] \quad (13)$$

$\overrightarrow{\text{RNN}}_c$, $\overleftarrow{\text{RNN}}_c$ are forward and backward LSTMs repectively, and $f_c(x_t) = \boldsymbol{M}_c^\top \mathbf{1}(x_t)$ is a function that maps a word to continous embedding space.

Held-out LSTM (HoLSTM) Yuan et al. (2016) trained a LSTM language model, which predicts a held-out word given the surrounding context, with a large amount of unlabeled text as training data. Given the context vector from this language model, they predict the word sense with a WSD classifier. Specifically, we can compute the context vector c_t for tth word by first replacing tth word with a special symbol (e.g. <$>). We then feed the replaced sequence to a uni-directional LSTM:

$$\tilde{\boldsymbol{c}}_i = \overrightarrow{\text{RNN}}_c(\tilde{\boldsymbol{c}}_{i-1}, f_c(x_i)) \quad (14)$$

Finally, we can get context vector for the tth word

$$\boldsymbol{c}_t = \tilde{\boldsymbol{c}}_n \quad (15)$$

$\overrightarrow{\text{RNN}}_c$ and f_c are defined in BiLSTM paragraph, and n is the length of the sequence. Despite the fact that the context vector is always the last hidden state of the LSTM no matter which word we are targeting, the input sequence read by the HoLSTM is actually different every time.

5 Adding Context to NMT

Now that we have several methods to incorporate global context regarding a single word, it is necessary to incorporate this context with NMT. Specifically, we propose two methods to either *Gate* or *Concatenate* a context vector $\boldsymbol{c}_t$ with the Lookup embedding $\boldsymbol{M}_e^\top \cdot \mathbf{1}(x_t)$ to form a context-aware word embedding before feeding it into the encoder as shown in Fig. 3. The detail of these methods is described below.

Gate Inspired by Choi et al. (2017), as our first method for integration of context-aware word embeddings, we use a gating function as follows:

$$f'_e(x_t) = f_e(x_t) \odot \sigma(\boldsymbol{c}_t) \quad (16)$$
$$= \boldsymbol{M}_e^\top \mathbf{1}(x_t) \odot \sigma(\boldsymbol{c}_t) \quad (17)$$

The symbol $\odot$ represents element-wise multiplication, and σ is element-wise sigmoid function.

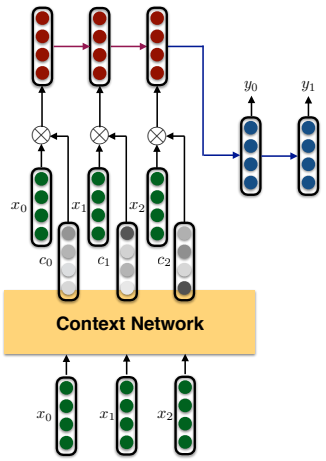


Figure 3: Illustration of our proposed model. The context network is a differentiable network that computes context vector c_t for word x_t taking the whole sequence as input. $\otimes$ represents the operation that combines original word embedding x_t with corresponding context vector c_t to form context-aware word embeddings.

Choi et al. (2017) use this method in concert with averaged embeddings from words in source language like the NBOW model above, which naturally uses the same context vectors for all time steps. In this paper, we additionally test this function with context vectors calculated using the BiLSTM and HoLSTM .

Concatenate We also propose another way for incorporating context: by concatenating the context vector with the word embeddings. This is expressed as below:

$$f'_e(x_t) = \boldsymbol{W}_3[f_e(x_t); \boldsymbol{c}_t] \quad (18)$$
$$= \boldsymbol{W}_3[\boldsymbol{M}_e^\top \mathbf{1}(x_t); \boldsymbol{c}_t] \quad (19)$$

$\boldsymbol{W}_3$ is used to project the concatenated vector back to the original d-dimensional space.

For each method can compute context vector $\boldsymbol{c}_t$ with either the NBOW, BiLSTM, or HoLSTM described in §4. We share the parameters in f_e with f_c (i.e. $\boldsymbol{M}_e = \boldsymbol{M}_c$) since the vocabulary space is the same for context network and encoder. As a result, our context network only slightly increases the number of model parameters. Details about the number of parameters of each model we use in the experiments are shown in Table 1.

6 Experiments

We evaluate our model on three different language pairs: English-French (WMT'14), and English-German (WMT'15), English-Chinese (WMT'17)

Context	Integration	uni/bi	#layers	#params	Ppl	WMT14	WMT15
None	-	$\rightarrow$	2	85M	7.12	20.49	22.95
None	-	$\leftrightarrow$	2	83M	7.20	**21.05**	**23.83**
None	-	$\leftrightarrow$	3	86M	7.50	20.86	23.14
NBOW	Concat	$\rightarrow$	2	85M	7.23	20.44	22.83
NBOW	Concat	$\leftrightarrow$	2	83M	7.28	20.76	23.61
HoLSTM	Concat	$\rightarrow$	2	87M	7.19	20.67	23.05
HoLSTM	Concat	$\leftrightarrow$	2	86M	7.04	21.15	23.53
BiLSTM	Concat	$\rightarrow$	2	87M	6.88	**21.80**	**24.52**
BiLSTM	Concat	$\leftrightarrow$	2	85M	6.87	21.33	24.37
NBOW	Gating	$\rightarrow$	2	85M	7.14	20.20	22.94
NBOW	Gating	$\leftrightarrow$	2	83M	6.92	21.16	23.52
BiLSTM	Gating	$\rightarrow$	2	87M	7.07	20.94	23.58
BiLSTM	Gating	$\leftrightarrow$	2	85M	7.11	21.33	24.05

Table 1: **WMT'14, WMT'15 English-German results** - We show perplexities (Ppl) on development set and tokenized BLEU on WMT'14 and WMT'15 test set of various NMT systems. We also show different settings for different systems. $\rightarrow$ represents uni-directional, and $\leftrightarrow$ represents bi-directional. We also highlight the best baseline model and the best proposed model in bold. The best baseline model will be referred as *base* or *baseline* and the best proposed model will referred to as *best* for further experiments.

with English as the source side. For German and French, we use a combination of Europarl v7, Common Crawl, and News Commentary as training set. For development set, newstest2013 is used for German and newstest2012 is used for French. For Chinese, we use a combination of News Commentary v12 and the CWMT Corpus as the training set and held out 2357 sentences as the development set. Translation performances are reported in case-sensitive BLEU on newstest2014 (2737 sentences), newstest2015 (2169 sentences) for German, newstest2013 (3000 sentences), newstest2014 (3003 sentences) for French, and newsdev2017 (2002 sentences) for Chinese.[4] Details about tokenization are as follows. For German, we use the tokenized dataset from Luong et al. (2015); for French, we used the moses (Koehn et al., 2007) tokenization script with the "-a" flag; for Chinese, we split sequences of Chinese characters, but keep sequences of non-Chinese characters as they are, using the script from IWSLT Evaluation 2015.[5]

We compare our context-aware NMT systems with strong baseline models on each dataset.

[4]We use the development set as testing data because the official test set hasn't been released.

[5]https://sites.google.com/site/iwsltevaluation2015/mt-track

System	BLEU	
en $\rightarrow$ de	**WMT'14**	**WMT'15**
baseline	21.05	23.83
best	**21.80**	**24.52**
en $\rightarrow$ fr	**WMT'13**	**WMT'14**
baseline	28.21	31.55
best	**28.77**	**32.39**
en $\rightarrow$ zh	**WMT'17**	
baseline	24.07	
best	**24.81**	

Table 2: **Results on three different language pairs** - The best proposed models (BiLSTM+Concat+uni) are significantly better (p-value $<$ 0.001) than baseline models using paired bootstrap resampling (Koehn, 2004).

6.1 Training Details

We limit our vocabularies to be the top 50K most frequent words for both source and target language. Words not in these shortlisted vocabularies are converted into an ⟨unk⟩ token.

When training our NMT systems, following Bahdanau et al. (2015), we filter out sentence pairs whose lengths exceed 50 words and shuffle mini-batches as we proceed. We train our model with the following settings using SGD as our optimization method. (1) We start with a learning rate of 1 and we begin to halve the learning rate every

epoch once it overfits. [6] (2) We train until the model converges. (i.e. the difference between the perplexity for the current epoch and the previous epoch is less than 0.01) (3) We batched the instances with the same length and our maximum mini-batch size is 256, and (4) the normalized gradient is rescaled whenever its norm exceeds 5. (6) Dropout is applied between vertical RNN stacks with probability 0.3. Additionally, the context network is trained jointly with the encoder-decoder architecture. Our model is built upon OpenNMT (Klein et al., 2017) with the default settings unless otherwise noted.

6.2 Experimental Results

In this section, we compare our proposed context-aware NMT models with baseline models on English-German dataset. Our baseline models are encoder-decoder models using global-general attention and input feeding on the decoder side as described in §2, varying the settings on the encoder side. Our proposed model builds upon baseline models by concatenating or gating different types of context vectors. We use LSTM for encoder, decoder, and context network. The decoder is the same across baseline models and proposed models, having 500 hidden units. During testing, we use beam search with a beam size of 5. The dimension for input word embedding d is set to 500 across encoder, decoder, and context network. Settings for three different baselines are listed below.

Baseline 1: An uni-directional LSTM with 500 hidden units and 2 layers of stacking LSTM.

Baseline 2: A bi-directional LSTM with 250 hidden units and 2 layers of stacking LSTM. Each state is summarized by concatenating the hidden states of forward and backward encoder into 500 hidden units.

Baseline 3: A bi-directional LSTM with 250 hidden units and 3 layers of stacking LSTM. This can be compared with the proposed method, which adds an extra layer of computation before the word embeddings, essentially adding an extra layer.

The context network uses the below settings.

[6] We define overfitting to be when perplexity on the dev set of the current epoch is worse than the previous epoch.

NBOW: Average word embedding of the input sequence.

BiLSTM: A single-layer bi-directional LSTM with 250 hidden units. The context vector is represented by concatenating the hidden states of forward and backward LSTM into a 500 dimensional vector.

HoLSTM: A single-layer uni-directional LSTM with 500 hidden units.

The results are shown in Table 1. The first thing we observe is that the best context-aware model (results in bold in the table) achieved improvements of around 0.7 BLEU on both WMT14 and WMT15 over the respective baseline methods with 2 layers. This is in contrast to simply using a 3-layer network, which actually degrades performance, perhaps due to the vanishing gradients problem it increases the difficulty in learning.

Next, comparing different methods for incorporating context, we can see that BiLSTM performs best across all settings. HoLSTM performs slightly better than NBOW, and NBOW obviously suffers from having the same context vector for every word in the input sequence failing to outperform the corresponding baselines. Comparing the two integration methods that incorporate context into word embeddings. Both methods improve over the baseline with BiLSTM as the context network. Concatenating the context vector and the word embedding performed better than gating. Finally, in contrast to the baseline, it is not obvious whether using uni-directional or bi-directional as the encoder is better for our proposed models, particularly when BiLSTM is used for calculating the context network. This is likely due to the fact that bi-directional information is already captured by the context network, and may not be necessary in the encoder itself.

We further compared the two systems on two different languages, French and Chinese. We achieved 0.5-0.8 BLEU improvement, showing our proposed models are stable and consistent across different language pairs. The results are shown in Table 2.

To show that our 3-layer models are properly trained, we ran a 3-layer bidirectional encoder with residual networks on En-Fr and got 27.45 for WMT13 and 30.60 for WMT14, which is similarly lower than the two layer result. It should be noted that previous work such as Britz et al. (2017) have

language	System	Homograph			All Words		
		F1	Precision	Recall	F1	Precision	Recall
en → de	baseline	0.401	0.422	0.382	0.547	0.569	0.526
	best	0.426 (+*0.025*)	0.449 (+*0.027*)	0.405 (+*0.023*)	0.553 (+*0.006*)	0.576 (+*0.007*)	0.532 (+*0.006*)
en → fr	baseline	0.467	0.484	0.451	0.605	0.623	0.587
	best	0.480 (+*0.013*)	0.496 (+*0.012*)	0.465 (+*0.014*)	0.613 (+*0.008*)	0.630 (+*0.007*)	0.596 (+*0.009*)
en → zh	baseline	0.578	0.587	0.570	0.573	0.605	0.544
	best	0.590 (+*0.012*)	0.599 (+*0.012*)	0.581 (+*0.011*)	0.581 (+*0.008*)	0.612 (+*0.007*)	0.552 (+*0.008*)

Table 3: Translation results for homographs and all words in our NMT vocabulary. We compare scores for baseline and our best proposed model on three different language pairs. Improvements are in italic. We performed bootstrap resampling for 1000 times: our best model improved more on homographs than all words in terms of either f1, precision, or recall with $p < 0.05$, indicating statistical significance across all measures.

also noted that the gains for encoders beyond two layers is minimal.

6.3 Targeted Analysis

In order to examine whether our proposed model can better translate words with multiple senses, we evaluate our context-aware model on a list of homographs extracted from Wikipedia[7] compared to the baseline model on three different language pairs. For the baseline model, we choose the best-performing model, as described in §6.2.

To do so, we first acquire the translation of homographs in the source language using `fast-align` (Dyer et al., 2013). We run `fast-align` on all the parallel corpora including training data and testing data[8] because the unsupervised nature of the algorithm requires it to have a large amount of training data to obtain accurate alignments. The settings follow the default command on fast-align github page including heuristics combining forward and backward alignment. Since there might be multiple aligned words in the target language given a word in source language, we treat a match between the aligned translation of a targeted word of the reference and the translation of a given model as true positives and use F1, precision, and recall as our metrics, and take the micro-average across all the sentence pairs. [9] We calculated the scores for the 50000 words/characters from our source vocabulary using only English words. The results are shown in Table 3. The table shows two interesting results: (1) The score for the homographs is lower than the score obtained from all the words in the vocabulary. This shows that words with more meanings are harder to translate with Chinese as the only exception.[10] (2) The improvement of our proposed model over baseline model is larger on the homographs compared to all the words in vocabulary. This shows that although our context-aware model is better overall, the improvements are particularly focused on words with multiple senses, which matches the intuition behind the design of the model.

6.4 Qualitative Analysis

We show sample translations on English-Chinese WMT'17 dataset in Table 4 with three kinds of examples. We highlighted the English homograph in bold, correctly translated words in blue, and wrongly translated words in red. (1) Target homographs are translated into the correct sense with the help of context network. For the first sample translation, "meets" is correctly translated to "会见" by our model, and wrongly translated to "符合" by baseline model. In fact, "会见" is closer to the definition "come together intentionally" and "符合" is closer to "satisfy" in the English dictionary. (2) Target homographs are translated into different but similar senses for both models in the forth example. Both models translate the word "believed" to common translations "被认为" or "相信", but these meaning are both close to reference translation "据信". (3) Target homograph is translated into the wrong sense for the baseline model, but is not translated in our model in the fifth example.

[7] https://en.wikipedia.org/wiki/List_of_English_homographs

[8] Reference translation, and all the system generated translations.

[9] The link to the evaluation script – https://goo.gl/oHYR8E

[10] One potential explanation for Chinese is that because the Chinese results are generated on the character level, the automatic alignment process was less accurate.

English-Chinese Translations	
src	Ugandan president **meets** Chinese FM , anticipates closer cooperation
ref	乌干达总统**会见** 中国外长，期待增进合作(come together intentionally)
best	乌干达总统**会见** 中国调频，预期更密切合作(come together intentionally)
base	乌干达总统**符合** 中国调频，预期更加合作(satisfy)
src	Investigators are trying to **establish** whether Kermiche and Petitjean had accomplices in France and whether they had links with Islamic State , which has claimed responsibility for the attack .
ref	调查人员正试图 **确定** 克尔米奇和帕迪让在法国是否有同谋，以及是否与伊斯兰国武装分子有联系，伊斯兰国武装分子声称对此次袭击负责。(get proof of something)
best	调查人员正试图 **确定** Kermiche 和Petitjean 在法国是否有同谋，他们是否与伊斯兰国有联系，声称对这次袭击负责。(get proof of something)
base	调查人员正在努力 **建立** 法国的同谋和他们是否与伊斯兰国有联系, 该国声称对这次袭击负有责任。(to start)
src	The decrease of transaction settlement fund **balance** in the securities market in July was smaller than that in June , while the net bank @-@ securities transfers stood at negative RMB 66.6 billion .
ref	7 月证券市场交易结算资金**余额** 减少额较6 月大幅降低，银证转账变动净额为- 666 亿元。(money left)
best	7 月份证券市场交易结算资金**余额** 的减少小于6 月份，而银行证券转让净额为negative 亿元。(money left)
base	七月证券市场交易结算基金**平衡** 的减少比六月份小，而净银行证券转让则为负元。(equal weight or force)
src	Initial reports suggest that the gunman may have shot a woman , **believed** to be his ex @-@ partner .
ref	据初步报告显示，开枪者可能击中一名妇女，**据信** 是他的前搭档。(been accepted as truth)
best	初步的报道表明，枪手可能已经射杀了一个女人，**被认为** 是他的前伙伴。(been known as)
base	最初的报道显示，枪手可能已经射杀了一名妇女，**相信** 他是他的前伙伴。(accept as truth)
src	When the game came to the last 3 ' 49 ' ' , Nigeria **closed** to 79 @-@ 81 after Aminu added a layup .
ref	比赛还有3 分49 秒时，阿米努上篮得手后，尼日利亚将比分**追成** 了79-81 。(narrow)
best	当这场比赛到了最后三个“ 49 ”时，尼日利亚在Aminu 增加了一个layup 之后**MISSING TRANSLATION**。
base	当游戏到达最后3 “ 49 ”时，尼日利亚已经**关闭** 了Aminu 。(end)

Table 4: **Sample translations** - for each example, we show sentence in source language (src), the human translated reference (ref), the translation generated by our best context-aware model (best), and the translation generated by baseline model (base). We also highlight the word with multiple senses in source language in bold, the corresponding correctly translated words in blue and wrongly translated words in red. The definitions of words in blue or red are in parenthesis.

7 Related Work

Word sense disambiguation (WSD), the task of determining the correct meaning or sense of a word in context is a long standing task in NLP (Yarowsky, 1995; Ng and Lee, 1996; Mihalcea and Faruque, 2004; Navigli, 2009; Zhong and Ng, 2010; Di Marco and Navigli, 2013; Chen et al., 2014; Camacho-Collados et al., 2015). Recent research on tackling WSD and capturing multi-senses includes work leveraging LSTM (Kågebäck and Salomonsson, 2016; Yuan et al., 2016), which we extended as a context network in our paper and predicting senses with word embeddings that capture context. Šuster et al. (2016); Kawakami and Dyer (2016) also showed that bilingual data improves WSD. In contrast to the standard WSD formulation, Vickrey et al. (2005) reformulated the task of WSD for Statistical Machine Translation (SMT) as predicting possible target translations which directly improves the accuracy of machine translation. Following this reformulation, Chan et al. (2007); Carpuat and Wu (2007a,b) integrated WSD systems into phrase-based systems. Xiong and Zhang (2014) breaks the process into two stages. First predicts the sense of the ambiguous source word. The predicted word senses together with other context features are then used to predict possible target translation. Within the framework of Neural MT, there are works that has similar motivation to ours. Choi et al. (2017) leverage the NBOW as context and gate the word-embedding on both encoder and decoder side. However, their work does not distinguish context vectors for words in the same sequence, in contrast to the method in this paper, and our results demonstrate that this is an important feature of methods that handle homographs in NMT. In addition, our quantitative analysis of the problems that homographs pose to NMT and evaluation of how context-aware models fix them was not covered in this previous work. Rios et al. (2017) tackled the problem by adding sense embedding learned with additional corpus and evaluated the performance on the sentence level with contrastive translation.

8 Conclusion

Theoretically, NMT systems should be able to handle homographs if the encoder captures the clues to translate them correctly. In this paper, we empirically show that this may not be the case; the performance of word level translation degrades as the number of senses for each word increases. We hypothesize that this is due to the fact that each word is mapped to a word vector despite them being in different contexts, and propose to integrate methods from neural WSD systems into an NMT system to alleviate this problem. We concatenated the context vector computed from the context network with the word embedding to form a context-aware word embedding, successfully improving the NMT system. We evaluated our model on three different language pairs and outperformed a strong baseline model according to BLEU score in all of them. We further evaluated our results targeting the translation of homographs, and our model performed better in terms of F1 score.

While the architectures proposed in this work do not *solve* the problem of homographs, our empirical results in Table 3 demonstrate that they do yield improvements (larger than those on other varieties of words). We hope that this paper will spark discussion on the topic, and future work will propose even more focused architectures.

References

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

Luisa Bentivogli, Arianna Bisazza, Mauro Cettolo, and Marcello Federico. 2016. Neural versus phrase-based machine translation quality: a case study. In *EMNLP*. pages 257–267.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Denny Britz, Anna Goldie, Thang Luong, and Quoc Le. 2017. Massive exploration of neural machine translation architectures. *arXiv:1703.03906* .

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. A unified multilingual semantic representation of concepts. In *ACL*. pages 741–751.

Marine Carpuat and Dekai Wu. 2007a. How phrase sense disambiguation outperforms word sense disambiguation for statistical machine translation. *TMI* pages 43–52.

Marine Carpuat and Dekai Wu. 2007b. Improving statistical machine translation using word sense disambiguation. In *EMNLP-CoNLL*. pages 61–72.

Yee Seng Chan, Hwee Tou Ng, and David Chiang. 2007. Word sense disambiguation improves statistical machine translation. In *ACL*. volume 45, pages 33–40.

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *EMNLP*. pages 1025–1035.

Heeyoul Choi, Kyunghyun Cho, and Yoshua Bengio. 2017. Context-dependent word representation for neural machine translation. *arXiv:1607.00578* .

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

Antonio Di Marco and Roberto Navigli. 2013. Clustering and diversifying web search results with graph-based word sense induction. *Computational Linguistics* 39(3):709–754.

Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. NAACL-HLT, pages 644–648.

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

Mohit Iyyer, Varun Manjunatha, Jordan L Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *ACL*. pages 1681–1691.

Mikael Kågebäck and Hans Salomonsson. 2016. Word sense disambiguation using a bidirectional lstm. *COLING 2016* page 51.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *ACL* pages 212–217.

Kazuya Kawakami and Chris Dyer. 2016. Learning to represent words in context with multilingual supervision. *ICLR workshop* .

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *arXiv:1701.02810* .

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *EMNLP*. pages 388–395.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source

toolkit for statistical machine translation. In *ACL*. pages 177–180.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *NAACL*. pages 48–54.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *EMNLP* pages 1412–1421.

Rada Mihalcea and Ehsanul Faruque. 2004. Senselearner: Minimally supervised word sense disambiguation for all words in open text. In *ACL/SIGLEX*. volume 3, pages 155–158.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)* 41(2):10.

Graham Neubig, Makoto Morishita, and Satoshi Nakamura. 2015. Neural reranking improves subjective quality of machine translation: Naist at wat2015. *WAT* .

Hwee Tou Ng and Hian Beng Lee. 1996. Integrating multiple knowledge sources to disambiguate word sense: An exemplar-based approach. In *ACL*. pages 40–47.

Xiao Pu, Nikolaos Pappas, and Andrei Popescu-Belis. 2017. Sense-aware statistical machine translation using adaptive context-dependent clustering. In *WMT*. pages 1–10.

Annette Rios, Laura Mascarell, and Rico Sennrich. 2017. Improving word sense disambiguation in neural machine translation with sense embeddings. *WMT 2017* page 11.

Simon Šuster, Ivan Titov, and Gertjan van Noord. 2016. Bilingual learning of multi-sense embeddings with discrete autoencoders. *NAACL-HLT* pages 1346–1356.

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

David Vickrey, Luke Biewald, Marc Teyssier, and Daphne Koller. 2005. Word-sense disambiguation for machine translation. In *HLT-EMNLP*. pages 771–778.

Deyi Xiong and Min Zhang. 2014. A sense-based translation model for statistical machine translation. In *ACL*. pages 1459–1469.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *ACL*. pages 189–196.

Dayu Yuan, Julian Richardson, Ryan Doherty, Colin Evans, and Eric Altendorf. 2016. Semi-supervised word sense disambiguation with neural models. *arXiv:1603.07012* .

Zhi Zhong and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *ACL*. pages 78–83.

Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets

Zhen Yang[1,2], **Wei Chen**[1]*, **Feng Wang**[1,2], **Bo Xu**[1]
[1]Institute of Automation, Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences
{yangzhen2014, wei.chen.media, feng.wang, xubo}@ia.ac.cn

Abstract

This paper proposes an approach for applying GANs to NMT. We build a conditional sequence generative adversarial net which comprises of two adversarial sub models, a generator and a discriminator. The generator aims to generate sentences which are hard to be discriminated from human-translated sentences (i.e., the golden target sentences); And the discriminator makes efforts to discriminate the machine-generated sentences from human-translated ones. The two sub models play a mini-max game and achieve the win-win situation when they reach a Nash Equilibrium. Additionally, the static sentence-level BLEU is utilized as the reinforced objective for the generator, which biases the generation towards high BLEU points. During training, both the dynamic discriminator and the static BLEU objective are employed to evaluate the generated sentences and feedback the evaluations to guide the learning of the generator. Experimental results show that the proposed model consistently outperforms the traditional RNNSearch and the newly emerged state-of-the-art Transformer on English-German and Chinese-English translation tasks.

1 Introduction

Neural machine translation (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2014) which directly leverages a single neural network to transform the source sentence into the target sentence, has drawn more and more attention in both academia and industry (Shen et al., 2015; Wu et al., 2016; Johnson et al., 2016; Gehring et al., 2017; Vaswani et al., 2017). This end-to-end NMT typically consists of two sub neural networks. The encoder network reads and encodes the source sentence into the context vector representation; and the decoder network generates the target sentence word by word based on the context vector. To dynamically generate a context vector for a target word being generated, the attention mechanism which enables the model to focus on the relevant words in the source-side sentence is usually deployed. Under the encoder-decoder framework, many variants of the model structure, such as convolutional neural network (CNN) and recurrent neural network (RNN) are proposed (Bahdanau et al., 2014; Gehring et al., 2017). Recently, (Gehring et al., 2017) propose the Transformer, the first sequence transduction model based entirely on attention, achieving state-of-the-art performance on the English-German and English-French translation tasks. Despite its success, the Transformer, similar to traditional NMT models, is still optimized to maximize the likelihood estimation of the ground word (MLE) at each time step. Such an objective poses a hidden danger to NMT models. That is, the model may generate the best candidate word for the current time step yet a bad component of the whole sentence in the long run. Minimum risk training (MRT) (Shen et al., 2015) is proposed to alleviate such a limitation by adopting the sequence level objective, i.e., the sentence-level BLEU, for traditional NMT models. Yet somewhat improved, this objective still does not guarantee the translation results to be natural and sufficient. Since the BLEU point is computed as the geometric mean of the modified n-gram precisions (Papineni et al., 2002), almost all of the existing objectives essentially train NMT models to generate sentences with n-gram precisions as high as possible (MLE can be viewed to generate sentences with high 1-gram precisions). While n-gram precisions largely tell the good sentence apart from the bad one, it is widely acknowledged that higher n-gram precisions do not guarantee better sentences (Callison-Burch and Osborne, 2006; Chatterjee et al., 2007). Additionally, the manually defined objective, i.e.,

Proceedings of NAACL-HLT 2018, pages 1346–1355
New Orleans, Louisiana, June 1 - 6, 2018.

the n-gram precision, is unable to cover all crucial aspects of the data distribution and NMT models may be trained to generate suboptimal sentences (Luc et al., 2016).

In this paper, to address the limitation mentioned above, we borrow the idea of generative adversarial training from computer vision (Goodfellow et al., 2014; Denton et al., 2015) to directly train the NMT model generating sentences which are hard to be discriminated from human translations. The motivation behind is that while we can not manually define the data distribution of golden sentences comprehensively, we are able to utilize a discriminative network to learn automatically what the golden sentences look like. Following this motivation, we build a conditional sequence generative adversarial net where we jointly train two sub adversarial models: A generator generates the target-language sentence based on the input source-language sentence; And a discriminator, conditioned on the source-language sentence, predicts the probability of the target-language sentence being a human-generated one. During the training process, the generator aims to fool the discriminator into believing that its output is a human-generated sentence, and the discriminator makes efforts not to be fooled by improving its ability to distinguish the machine-generated sentence from the human-generated one. This kind of adversarial training achieves a win-win situation when the generator and discriminator reach a Nash Equilibrium (Zhao et al., 2016; Arora et al., 2017; Guimaraes et al., 2017). Besides generating the desired distribution, we also want to directly guide the generator with a static and specific objective, such as generating sentences with high BLEU points. To this end, the smoothed sentence-level BLEU (Nakov et al., 2012) is utilized as the reinforced objective for the generator. During training, we employ both the dynamic discriminator and the static BLEU objective to evaluate the generated sentences and feedback the evaluations to guide the learning of the generator. In summary, we mainly make the following contributions:

- To the best of our knowledge, this work is among the first endeavors to introduce the generative adversarial training into NMT. We directly train the NMT model to generate sentences which are hard to be discriminated from human translations. The proposed model can be applied to any end-to-end NMT systems.

- We conduct extensive experiments on English-German and Chinese-English translation tasks and we test two different NMT models, the traditional RNNSearch (Bahdanau et al., 2014) and the state-of-the-art Transformer. Experimental results show that the proposed approach consistently achieves great success.

- Last but not least, we propose the smoothed sentence-level BLEU as the static and specific objective for the generator which biases the generation towards achieving high BLEU points. We show that the proposed approach is a weighted combination of the naive GAN and MRT.

2 Background and Related Work

2.1 RNNSearch and Transformer

The RNNSearch is the traditional NMT model which has been widely explored. We follow the de facto standard implementation by (Bahdanau et al., 2014). The encoder is a bidirectional gated recurrent units that encodes the input sequence $x = (x_1, \ldots, x_m)$ and calculates the forward sequence of hidden states $(\overrightarrow{h_1}, \ldots, \overrightarrow{h_m})$, and a backward sequence of hidden states $(\overleftarrow{h_1}, \ldots, \overleftarrow{h_m})$. The final annotation vector h_j is calculated by concatenating $\overrightarrow{h_j}$ and $\overleftarrow{h_j}$. The decoder is a recurrent neural network that predicts a target sequence $y = (y_1, \ldots, y_n)$. Each word y_i is predicted on a recurrent hidden state s_i, the previously predicted word y_{i-1} and a context vector c_i. The c_i is computed as a weighted sum of the encoded annotations h_j. The weight a_{ij} of each annotation h_j is computed by the attention mechanism, which models the alignment between y_i and x_j.

The Transformer, recently proposed by (Vaswani et al., 2017), achieves state-of-the-art results on both WMT2014 English-German and WMT2014 English-French translation tasks. The encoder of Transformer is composed of a stack of six identical layers. Each layer consists of a multi-head self-attention and a simple position-wise fully connected feed-forward network. The decoder is also composed of a stack of six identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third

sub-layer, which performs multi-head attention over the output of the encoder stack. The Transformer can be trained significantly faster than architectures based on recurrent or convolutional layers since it allows for significantly more parallelization.

2.2 Generative adversarial nets

Generative adversarial network, has enjoyed great success in computer vision and has been widely applied to image generation (Zhu et al., 2017; Radford et al., 2015). The conditional generative adversarial nets (Gauthier, 2014) apply an extension of generative adversarial network to a conditional setting, which enables the networks to condition on some arbitrary external data. Some recent works have begun to apply the generative adversarial training into the NLP area: (Chen et al., 2016) apply the idea of generative adversarial training to sentiment analysis and (Zhang et al., 2017) use the idea to domain adaptation tasks. For sequence generation problem, (Yu et al., 2016) leverage policy gradient reinforcement learning to back-propagate the reward from the discriminator, showing presentable results for poem generation, speech language generation and music generation. Similarly, (Zhang et al., 2016) generate the text from random noise via adversarial training. A striking difference from the works mentioned above is that, our work is in the conditional setting where the target-language sentence is generated conditioned on the source-language one. In parallel to our work, (Li et al., 2017) propose a similar conditional sequence generative adversarial training for dialogue generation. They use a hierarchical long-short term memory (LSTM) architecture for the discriminator. In contrast to their approach, we apply the CNN-based discriminator for the machine translation task. Furthermore, we propose to utilize the sentence-level BLEU as the specific objective for the generator. Detailed training strategies for the proposed model and extensive quantitative results are reported. We noticed that (Wu et al., 2017) is exploring the potential of GAN in NMT too. There are some differences in training strategies and experimental settings between (Wu et al., 2017) and this work. And the most significant difference is that we propose a novel BLEU-reinforced GAN for NMT [1].

3 The Approach

3.1 Model overview

In this section, we describe the architecture of the proposed BLEU reinforced conditional sequence generative adversarial net (referred to as BR-CSGAN) in detail. The sentence generation process is viewed as a sequence of actions that are taken according to a policy regulated by the generator. In this work, we take the policy gradient training strategies following (Yu et al., 2016). The whole architecture of the proposed model is depicted in figure 1. The model mainly consists of three sub modules:

Generator Based on the source-language sentences, the generator G aims to generate target-language sentences indistinguishable from human translations.

Discriminator The discriminator D, conditioned on the source-language sentences, tries to distinguish the machine-generated sentences from human translations. D can be viewed as a dynamic objective since it is updated synchronously with G.

BLEU objective The sentence-level BLEU Q serves as the reinforced objective, guiding the generation towards high BLEU points. Q is a static function which will not be updated during training.

3.2 Generator

Resembling NMT models, the generator G defines the policy that generates the target sentence y given the source sentence x. The generator takes exactly the same architecture with NMT models. Note that we do not assume the specific architecture of the generator. To verify the effectiveness of the proposed method, we take two different architectures for the generator, the RNNSearch [2] and Transformer [3].

3.3 Discriminator

Recently, the deep discriminative models such as the CNN and RNN have shown a high performance in complicated sequence classification tasks. Here, the discriminator is implemented based on the CNN architecture.

Since sentences generated by the generator have variable lengths, the CNN padding is used to transform the sentences to sequences with fixed length

[1] The previous presentation of this work can be found at https://arxiv.org/abs/1703.04887

[2] https://github.com/nyu-dl/dl4mt-tutorial

[3] https://github.com/tensorflow/tensor2tensor

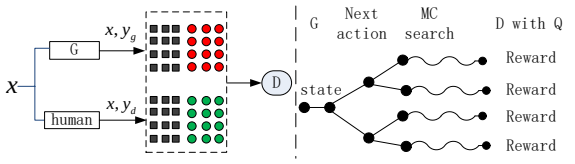


Figure 1: The Illustration of the proposed BR-CSGAN. Left: D is trained over the real sentence pairs translated by the human and the generated sentence pairs by G. Note that D is a conditional discriminator. Right: G is trained by police gradient where the final reward is provided by D and Q.

T, which is the maximum length set for the output of the generator. Given the source-language sequence $x_1, \ldots, x_T$ and target-language sequence $y_1, \ldots, y_T$, we build the source matrix $X_{1:T}$ and target matrix $Y_{1:T}$ respectively as:

$$X_{1:T} = x_1; x_2; \ldots; x_T \tag{1}$$

and

$$Y_{1:T} = y_1; y_2; \ldots; y_T \tag{2}$$

where x_t, $y_t \in R^k$ is the k-dimensional word embedding and the semicolon is the concatenation operator. For the source matrix $X_{1:T}$, a kernel $w_j \in R^{l \times k}$ applies a convolutional operation to a window size of l words to produce a series of feature maps:

$$c_{ji} = \rho(BN(w_j \otimes X_{i:i+l-1} + b)) \tag{3}$$

where $\otimes$ operator is the summation of element-wise production and b is a bias term. ρ is a non-linear activation function which is implemented as ReLu in this paper. Note that the batch normalization (Ioffe and Szegedy, 2015) which accelerates the training significantly, is applied to the input of the activation function (BN in equation 3). To get the final feature with respect to kernel w_j, a max-over-time pooling operation is leveraged over the feature maps:

$$\widetilde{c}_j = max\{c_{j1}, \ldots, c_{jT-l+1}\} \tag{4}$$

We use various numbers of kernels with different window sizes to extract different features, which are then concatenated to form the source-language sentence representation c_x. Identically, the target-language sentence representation c_y can be extracted from the target matrix $Y_{1:T}$. Finally, given the source-language sentence, the probability that the target-language sentence is being real can be computed as:

$$p = \sigma(V[c_x; c_y]) \tag{5}$$

where V is the transform matrix which transforms the concatenation of c_x and c_y into a 2-dimension embedding and σ is the logistic function.

3.4 BLEU objective

We apply the smoothed sentence-level BLEU as the specific objective for the generator. Given the generated sentence y_g and the the ground true sentence y_d, the objective Q calculates a reward $Q(y_g, y_d)$, which measures the n-gram precisions of the generated sentence y_g. Identical to the output of the discriminator, the $Q(y_g, y_d)$ also ranges from zero to one, which makes it easier to fuse Q and D.

3.5 Policy Gradient Training

Following (Yu et al., 2016), the objective of the generator G is defined as to generate a sequence from the start state to maximize its expected end reward. Formally, the objective function is computed as:

$$J(\theta) = \sum_{Y_{1:T}} G_\theta(Y_{1:T}|X) \cdot R^{G_\theta}_{D,Q}(Y_{1:T-1}, X, y_T, Y^*)$$

where θ represents the parameters in G, $Y_{1:T} = y_1, \ldots, y_T$ indicates the generated target sequence, X is the source-language sentence, Y^* represents the ground true target sentence. $R^{G_\theta}_{D,Q}$ is the action-value function of a target-language sentence given the source sentence X, i.e. the expected accumulative reward starting from the state $(Y_{1:T-1}, X)$, taking action y_T, and following the policy G_θ. To estimate the action-value function, we consider the estimated probability of being real by the discriminator D and the output of the BLEU objective Q as the reward:

$$R^{G_\theta}_{D,Q}(Y_{1:T-1}, X, y_T, Y^*) =$$
$$\lambda(D(X, Y_{1:T}) - b(X, Y_{1:T})) + (1 - \lambda)Q(Y_{1:T}, Y^*)$$

where b(X,Y) denotes the baseline value to reduce the variance of the reward. Practically, we take b(X,Y) as a constant, 0.5 for simplicity. And the λ is a hyper-parameter. The question is that, given the source sequence, D only provides a reward value for a finished target sequence. If $Y_{1:T}$ is not a finished target sequence, the value of $D(X, Y_{1:T})$ makes no sense. Therefore, we cannot get the

action-value for an intermediate state directly. To evaluate the action-value for an intermediate state, the Monte Carlo search under the policy of G_θ is applied to sample the unknown tokens. Each search ends until the end of sentence token is sampled or the sampled sentence reaches the maximum length. To obtain more stable reward and reduce the variance, we represent an N-time Monte Carlo search as:

$$\{Y^1_{1:T_1}, \ldots, Y^N_{1:T_N}\} = MC^{G_\theta}((Y_{1:t}, X), N)$$

where T_i represents the length of the sentence sampled by the i'th Monte Carlo search. $(Y_{1:t}, X) = (y_1, \ldots, y_t, X)$ is the current state and $Y^N_{t+1:T_N}$ is sampled based on the policy G_θ. The discriminator provides N rewards for the sampled N sentences respectively. The final reward for the intermediate state is calculated as the average of the N rewards. Hence, for the target sentence with the length T, we compute the reward for y_t in the sentence level as:

$$R^{G_\theta}_{D,Q}(Y_{1:t-1}, X, y_T, Y^*) =$$
$$\begin{cases} \frac{1}{N}\sum_{n=1}^{N} \lambda(D(X, Y^n_{1:T_n}) - b(X, Y^n_{1:T_n})) + (1-\lambda)Q(Y_{1:T_n}, Y*) & t < T \\ \lambda(D(X, Y_{1:t}) - b(X, Y_{1:t})) + (1-\lambda)Q(Y_{1:t}, Y^*) & t = T \end{cases}$$

Using the discriminator as a reward function can further improve the generator iteratively by dynamically updating the discriminator. Once we get more realistic generated sequences, we re-train the discriminator as:

$$\min -\mathbb{E}_{X,Y\in P_{data}}[\log D(X,Y)] - \mathbb{E}_{X,Y\in G}[\log(1 - D(X,Y))]$$

After updating the discriminator, we are ready to re-train the generator. The gradient of the objective function $J(\theta)$ w.r.t the generator's parameter θ is calculated as:

$$\nabla J(\theta) = \frac{1}{T}\sum_{t=1}^{T}\sum_{y_t} R^{G_\theta}_{D,Q}(Y_{1:t-1}, X, y_T, Y^*) \cdot \nabla_\theta(G_\theta(y_t|Y_{1:t-1}, X))$$
$$= \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{y_t\in G_\theta}[R^{G_\theta}_{D,Q}(Y_{1:t-1}, X, y_T, Y^*) \cdot \nabla_\theta \log p(y_t|Y_{1:t-1}, X)]$$

3.6 Training strategies

GANs are widely criticized for its unstable training since the generator and discriminator need to be carefully synchronized. To make this work easier to reproduce, this paper gives detailed strategies for training the proposed model.

Firstly, we use the maximum likelihood estimation to pre-train the generator on the parallel training set until the best translation performance is achieved. Then, generate the machine-generated sentences by using the generator to decode the training data. We simply use the greedy sampling method instead of the beam search method for decoding. Next, pre-train the discriminator on the combination of the true parallel data and the machine-generated data until the classification accuracy achieves at ξ. Finally, we jointly train the generator and discriminator. The generator is trained with the policy gradient training method. However, in our practice, we find that updating the generator only with the simple policy gradient training leads to unstableness. To alleviate this issue, we adopt the teacher forcing approach which is similar to (Lamb et al., 2016; Li et al., 2017). We directly make the discriminator to automatically assign a reward of 1 to the golden target-language sentence and the generator uses this reward to update itself on the true parallel example. We run the teacher forcing training for one time once the generator is updated by the policy gradient training. After the generator gets updated, we use the new stronger generator to generate η more realistic sentences, which are then used to train the discriminator. Following (Arjovsky et al., 2017), we clamp the weights of the discriminator to a fixed box ($[-\epsilon,\epsilon]$) after each gradient update. We perform one optimization step for the discriminator for each step of the generator. In our practice, we set ξ as 0.82, η as 5000, ϵ as 1.0 and the N for Monte Carlo search as 20.

4 Experiments and Results

We evaluate our BR-CSGAN on English-German and Chinese-English translation tasks and we test two different architectures for the generator, the traditional RNNSearch and the newly emerged state-of-the-art Transformer.

4.1 Data sets and preprocessing

English-German: For English-German translation, we conduct our experiments on the publicly available corpora used extensively as benchmark for NMT systems, WMT'14 En-De. This data set contains 4.5M sentence pairs [4]. Sentences are encoded using byte-pair encoding (Sennrich et al., 2015), which has a shared source-target vocabulary of about 37000 tokens. We report results on newstest2014. The newstest2013 is used as validation.

[4] http://nlp.stanford.edu/projects/nmt

Model	Chinese-English				English-German
	NIST03	NIST04	NIST05	average	newstest2014
RNNSearch (Bahdanau et al., 2014)	33.93	35.67	32.24	33.94	21.2
Transformer (Vaswani et al., 2017)	42.23	42.17	41.02	41.80	27.30
RNNSearch+BR-CSGAN($\lambda = 1.0$)	35.21	36.51	33.45	35.05	22.1
RNNSearch+BR-CSGAN($\lambda = 0.7$)	35.97	37.32	34.03	**35.77**	**22.89**
RNNSearch+BR-CSGAN($\lambda = 0$)	34.57	35.93	33.07	34.52	21.75
Transformer+BR-CSGAN($\lambda = 1.0$)	42.67	42.79	41.54	42.30	27.75
Transformer+BR-CSGAN($\lambda = 0.8$)	43.01	42.96	41.86	**42.61**	**27.92**
Transformer+BR-CSGAN($\lambda = 0$)	42.41	42.74	41.29	42.14	27.49

Table 1: BLEU score on Chinese-English and English-German translation tasks. The hyper-parameter λ is selected according to the development set. For the Transformer, following (Vaswani et al., 2017), we report the result of a single model obtained by averaging the 5 checkpoints around the best model selected on the development set.

Chinese-English: For Chinese-English translation, our training data consists of 1.6M sentence pairs randomly extracted from LDC corpora [5]. Both the source and target sentences are encoded with byte-pair encoding and the tokens in the source and target vocabulary is about 38000 and 34000 respectively [6]. We choose the NIST02 as the development set. For testing, we use NIST03, NIST04 and NIST05 data sets.

To speed up the training procedure, sentences of length over 50 words are removed when we conduct experiments on the RNNSearch model. This is widely used by previous works (Ranzato et al., 2015; Shen et al., 2015; Yang et al., 2016).

4.2 Model parameters and evaluation

For the Transformer, following the base model in (Vaswani et al., 2017), we set the dimension of word embedding as 512, dropout rate as 0.1 and the head number as 8. The encoder and decoder both have a stack of 6 layers. We use beam search with a beam size of 4 and length penalty $\alpha = 0.6$. For the RNNSearch, following (Bahdanau et al., 2014), We set the hidden units for both encoders and decoders as 512. The dimension of the word embedding is also set as 512. We do not apply dropout for training the RNNSearch. During testing, we use beam search with a beam size of 10 and length penalty is not applied.

All models are implemented in TensorFlow (Abadi et al., 2015) and trained on up to four K80 GPUs synchronously in a multi-GPU setup on a single machine [7]. We stop training when the model achieves no improvement for the tenth evaluation on the development set. BLEU (Papineni et al., 2002) is utilized as the evaluation metric. We apply the script *mteval-v11b.pl* to evaluate the Chinese-English translation and utilize the script *multi-belu.pl* for English-German translation [8].

4.3 Main results

The model of RNNSearch is optimized with the mini-batch of 64 examples. It takes about 30 hours to pre-train the RNNSearch on the Chinese-English data set and 46 hours on the English-German data set. During generative adversarial training, it takes about 35 hours on the Chinese-English data set and about 50 hours on the English-German data set. For the Transformer, each training batch contains a set of sentence pairs containing approximately 25000 source tokens and 25000 target tokens. On the Chinese-English data set, it takes about 15 hours to do pre-training and 20 hours to do generative adversarial training. On the English-German data set, it takes about 35 hours for the pre-training and 40 hours for the generative adversarial training.

Table 1 shows the BLEU score on Chinese-English and English-German test sets. On the RNNSearch model, the naive GAN (i.e., the line of RNNSearch+BR-CSGAN (λ=1) in table 1) achieves improvement up to +1.11 BLEU points averagely on Chinese-English test sets and +0.9 BLEU points on English-German test set. Armed

[5] LDC2002L27, LDC2002T01, LDC2002E18, LDC2003E07, LDC2004T08, LDC2004E12, LDC2005T10

[6] When doing BPE for Chinese, we need to do word segmentation first and the following steps are the same with BPE for English.

[7] The code we used to train and evaluate our models can be found at https://github.com/ZhenYangIACAS/NMT_GAN

[8] https://github.com/moses-smt/mosesdecoder/blob/617e8c8/scripts/generic/multi-bleu.perl;mteval-v11b.pl

with the BLEU objective, the BR-CSGAN (the line of RNNSearch+BR-CSGAN (λ=0.7)) leads to more significant improvements, +1.83 BLEU points averagely on Chinese-English translation and +1.69 BLEU points on English-German translation. We also test the translation performance when the RNNSearch is only guided by the static BLEU objective (the line of RNNSearch+BR-CSGAN (λ=0)), and we only get +0.58 BLEU points improvement on Chinese-English translation and +0.55 BLEU points improvement on English-German. Experiments on the Transformer show the same trends. While the Transformer has achieved state-of-the-art translation performances, our approach still achieves +0.81 BLEU points improvement on Chinese-English translation and +0.62 BLEU points improvement on English-German.

These results indicate that the proposed BR-CSGAN consistently outperforms the baselines and it shows better translation performance than the naive GAN and the model guided only by the BLEU objective.

5 Analysis

5.1 Compared with MRT

We show that MRT (Shen et al., 2015) is an extreme case of our approach. Considering a sentence pair (x, y), the training objective of MRT is calculated as

$$\widehat{J}(\theta') = \sum_{y^s \in S(x)} p(y^s|x;\theta')\Delta(y^s, y)$$

where $\Delta(y^s, y)$ is a loss function (i.e., the sentence-level BLEU used in this paper) that measures the discrepancy between a predicted translation y^s and the training example y, $S(x)$ represents the set which contains all of the predictions given the input x, and θ' is the parameters of the NMT model. Unfortunately, this objective is usually intractable due to the exponential search space. To alleviate this problem, a subset of the search space is sampled to approximate this objective. In this paper, when we set λ as zero, the objective for the proposed BR-CSGAN comes to

$$J(\theta)_{\lambda=0} = \sum_{Y_{1:T}} G_\theta(Y_{1:T}|X) \cdot Q(Y_{1:T}, Y^*)$$

where the $Q(Y_{1:T}, Y^*)$ is also a loss function between the predicted translation $Y_{1:T}$ and the training example Y^*. It is easy to be found that, under this condition (i.e., λ set as zero), the proposed BR-CSGAN optimizes almost the same objective with MRT. The only difference is that the reinforcement learning procedure is utilized in BR-CSGAN to maximize the total reward and MRT instead applies random sampling to approximate the risk. Actually, the BR-CSGAN is a weighted sum of the naive GAN (λ=1) and MRT (λ=0), and it incorporates the advantages of the two approaches. Specifically, compared to naive GAN which is trained without specific objective guidance, BR-CSGAN utilizes the BLEU objective to guide the generator to generate sentences with higher BLEU points. And compared to MRT which is trained only with the static objective, the BR-CSGAN applies a dynamic discriminator which updates synchronously with the generator, to feedback the dynamic rewards for the generator. Table 2 compares the translation performance between the MRT and BR-CSGAN on Chinese-English and English-German translation tasks. We only conduct experiments on the RNNSearch because we only get the open-source implementation of MRT on the RNNSearch [9]. Results show that the proposed BR-CSGAN consistently outperforms the MRT on the Chinese-English and English-German translations.

Model	Chinese-English average	English-German newstest2014
RNNSearch	33.94	21.2
MRT (Shen et al., 2015)	34.64	21.6
BR-CSGAN($\lambda = 0.7$)	35.77	22.89

Table 2: BLEU score on Chinese-English and English-German translation tasks for MRT and BR-CSGAN.

5.2 When to stop pre-training

The initial accuracy ξ of the discriminator which is viewed as a hyper-parameter, can be set carefully during the process of pre-training. A natural question is that when shall we end the pre-training. Do we need to pre-train the discriminator with the highest accuracy? To answer this question, we test the impact of the initial accuracy of the discriminator. We pre-train five discriminators which have the accuracy as 0.6, 0.7, 0.8, 0.9 and 0.95 respectively. With the five discriminators, we train five different BR-CSGAN models (with the generator as RNNSearch and λ set as 0.7) and test

[9]The open-source implementation can be found at: https://github.com/EdinburghNLP/nematus

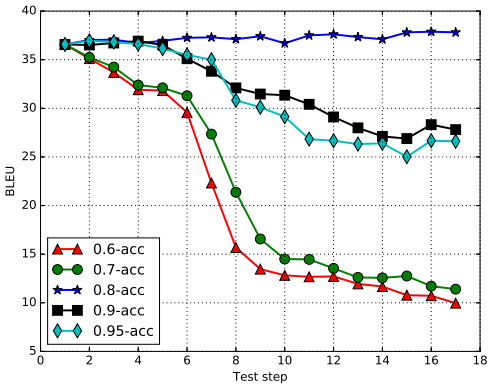


Figure 2: BLEU score on the development set for the BR-CSGAN where the discriminators have different initial accuracy. "0.6-acc" means the initial accuracy is 0.6. We report the results on the Chinese-English translation tasks. RNNSearch is taken as the generator.

their translation performances on the development set at regular intervals. Figure 2 reports the results and we can find that the initial accuracy of the discriminator shows great impacts on the translation performance of the proposed BR-CSGAN. From figure 2, we show that the initial accuracy of the discriminator needs to be set carefully and either it is too high (0.9 and 0.95) or too low (0.6 and 0.7), the model performs badly [10]. This suggests that it is important for the generator and discriminator to keep a balanced relationship at the beginning of the generative adversarial training. If the discriminator is too strong, the generator is always penalized for its bad predictions and gets no idea about right predictions. Hence, the generator is discouraged all the time and the performance gets worse and worse. On the other hand, if the discriminator is too weak, the discriminator is unable to give right guidance for the generator, i.e. the gradient direction for updating the generator is random. Empirically, we pre-train the discriminator until its accuracy reaches around 0.8.

5.3 Sample times for Monto Carol search

We are also curious about how the sample times N for Monte Carlo search affects the translation performance. Intuitively, if N is set as a small number, the intermediate reward for each word may be incorrect since there is a large variance for the Monto Carol search when the sample time is too small. And if otherwise, the computation shall be very time consuming because we need to do much more sampling. Therefore, there is a trade-off between the accuracy and computation complexity here. We investigate this problem on the Chinese-English translation task. Table 3 presents the translation performance of the BR-CSGAN on the test sets when the N are set from 5 to 30 with interval 5. From table 3, the proposed model achieves no improvement than the baseline (i.e., the pre-trained generator) when N are set less than 15 and the BLEU scores are not reported on the table. As a matter of fact, the translation performance of the model gets worse and worse. We conjecture that the approximated reward is far from the expected reward due to the large variance when N is set too small, and gives wrong gradient directions for model updating. Since the training for GAN is not stable, the wrong gradient direction exacerbates the unstableness and results in the BLEU getting worse and worse. With the increasing of N, the translation performance of the model gets improved. However, with N set larger than 20, we get little improvement than the model with N set as 20 and the training time exceeds our expectation.

[10] To make the illustration simple and clear, we only depict the results when the RNNSearch acts as the generator.

N	NIST02	NIST03	NIST04	NIST05
0	36.87	33.93	35.67	32.24
5	-	-	-	-
10	-	-	-	-
15	37.34	34.91	36.09	33.45
20	38.58	35.97	37.32	34.03
25	38.65	36.04	37.52	33.91
30	38.74	36.01	37.54	33.76

Table 3: The translation performance of the BR-CSGAN with different N for Monte Carlo search. "-" means that the proposed model shows no improvement than the pre-trained generator or it can not be trained stably. With N set as 0, it is referred to as the pre-trained generator. Similarly, we only report results on the RNNSearch and λ is set as 0.7.

6 Conclusion and Future Work

In this work, we propose the BR-CSGAN which leverages the BLEU reinforced generative adversarial net to improve the NMT. We show that the proposed approach is a weighted combination of the naive GAN and MRT. To verify the effectiveness of our approach, we test two different architectures for the generator, the traditional RNNSearch and the state-of-the-art Transformer.

Extensive experiments on Chinese-English and English-German translation tasks show that our approach consistently achieves significant improvements. In the future, we would like to try multi-adversarial framework which consists of multi discriminators and generators for GAN.

Acknowledgements

This work is supported by the National Key Research and Development Program of China under Grant No. 2017YFB1002102, and Beijing Engineering Research Center under Grant No. Z171100002217015. We would like to thank Xu Shuang for her preparing data used in this work. Additionally, we also want to thank Chen Zhineng, Wang Wenfu and Zhao Yuanyuan for their invaluable discussions on this work.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems .

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875* .

Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. 2017. Generalization and equilibrium in generative adversarial nets. *ICML* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Chris Callison-Burch and Miles Osborne. 2006. Re-evaluating the role of bleu in machine translation research .

Niladri Chatterjee, Anish Johnson, and Madhav Krishna. 2007. Some improvements over the bleu metric for measuring translation quality for hindi. In *Computing: Theory and Applications, 2007. ICCTA'07. International Conference on*. IEEE, pages 485–490.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2016. Adversarial deep averaging networks for cross-lingual sentiment classification. *arXiv preprint arXiv:1606.01614* .

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

Emily L Denton, Soumith Chintala, Rob Fergus, et al. 2015. Deep generative image models using a? laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*. pages 1486–1494.

Jon Gauthier. 2014. Conditional generative adversarial nets for convolutional face generation. *Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition, Winter semester* 2014(5):2.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning .

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

Gabriel Lima Guimaraes, Benjamin Sanchez-Lengeling, Pedro Luis Cunha Farias, and Alán Aspuru-Guzik. 2017. Objective-reinforced generative adversarial networks (organ) for sequence generation models. *arXiv preprint arXiv:1705.10843* .

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* .

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation. *arXiv preprint arXiv:1611.04558* .

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. *EMNLP* pages 1700–1709.

Alex Lamb, Anirudh Goyal, Ying Zhang, Saizheng Zhang, Aaron Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. *Advances In Neural Information Processing Systems* pages 4601–4609.

Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547* .

Pauline Luc, Camille Couprie, Soumith Chintala, and Jakob Verbeek. 2016. Semantic segmentation using adversarial networks. *arXiv preprint arXiv:1611.08408* .

Preslav Nakov, Francisco Guzman, and Stephan Vogel. 2012. Optimizing for sentence-level bleu+ 1 yields short translations. In *COLING*. volume 12, pages 1979–1994.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. *Association for Computational Linguistics* pages 311–318.

Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434* .

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

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *Computer Science* .

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2015. Minimum risk training for neural machine translation. *arXiv preprint arXiv:1512.02433* .

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need .

Lijun Wu, Yingce Xia, Li Zhao, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2017. Adversarial neural machine translation. *arXiv preprint arXiv:1704.06933* .

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* .

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2016. A character-aware encoder for neural machine translation. In *COLING*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2016. Seqgan: Sequence generative adversarial nets with policy gradient. *The Association for the Advancement of Artificial Intelligence 2017* .

Yizhe Zhang, Zhe Gan, and Lawrence Carin. 2016. Generating text via adversarial training. *NIPS* .

Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2017. Aspect-augmented adversarial networks for domain adaptation. *arXiv preprint arXiv:1701.00188* .

Junbo Zhao, Michael Mathieu, and Yann LeCun. 2016. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126* .

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593* .

Neural Machine Translation for Bilingually Scarce Scenarios: A Deep Multi-task Learning Approach

Poorya Zaremoodi **Gholamreza Haffari**
Faculty of Information Technology, Monash University, Australia
first.last@monash.edu

Abstract

Neural machine translation requires large amounts of parallel training text to learn a reasonable-quality translation model. This is particularly inconvenient for language pairs for which enough parallel text is not available. In this paper, we use monolingual linguistic resources in the source side to address this challenging problem based on a multi-task learning approach. More specifically, we scaffold the machine translation task on auxiliary tasks including semantic parsing, syntactic parsing, and named-entity recognition. This effectively injects semantic and/or syntactic knowledge into the translation model, which would otherwise require a large amount of training bitext. We empirically evaluate and show the effectiveness of our multi-task learning approach on three translation tasks: English-to-French, English-to-Farsi, and English-to-Vietnamese.

1 Introduction

Neural Machine Translation (NMT) with attentional encoder-decoder architectures (Luong et al., 2015; Bahdanau et al., 2015) has revolutionised machine translation, and achieved state-of-the-art for several language pairs. However, NMT is notorious for its need for large amounts of bilingual data (Koehn and Knowles, 2017) to achieve reasonable translation quality. Leveraging existing monolingual resources is a potential approach for compensating this requirement in bilingually scarce scenarios. Ideally, semantic and syntactic knowledge learned from existing linguistic resources provides NMT with proper inductive biases, leading to increased generalisation and better translation quality.

Multi-task learning (MTL) is an effective approach to inject knowledge into a task, which is learned from other related tasks. Various recent works have attempted to improve NMT with an MTL approach (Peng et al., 2017; Liu et al., 2017; Zhang and Zong, 2016); however, they either do not make use of curated linguistic resources (Domhan and Hieber, 2017; Zhang and Zong, 2016), or their MTL architectures are restrictive yielding mediocre improvements (Niehues and Cho, 2017). The current research leaves open how to best leverage curated linguistic resources in a suitable MTL framework to improve NMT.

In this paper, we make use of curated monolingual linguistic resources in the source side to improve NMT in bilingually scarce scenarios. More specifically, we scaffold the machine translation task on auxiliary tasks including semantic parsing, syntactic parsing, and named-entity recognition. This is achieved by casting the auxiliary tasks as sequence-to-sequence (SEQ2SEQ) transduction tasks, and tie the parameters of their encoders and/or decoders with those of the main translation task. Our MTL architectures makes use of deep stacked encoders and decoders, where the parameters of the top layers are shared across the tasks. We further make use of adversarial training to prevent contamination of common knowledge with task-specific information.

We present empirical results on translating from English into French, Vietnamese, and Farsi; three target languages with varying degree of divergence compared to English. Our extensive empirical results demonstrate the effectiveness of our MTL approach in substantially improving the translation quality for these three translation tasks in bilingually scarce scenarios.

2 Neural SEQ2SEQ Transduction

Our MTL is based on the attentional encoder-decoder architecture for SEQ2SEQ transduction. It contains an encoder to *read* the input sentence, and an attentional decoder to *generate* the output.

Proceedings of NAACL-HLT 2018, pages 1356–1365
New Orleans, Louisiana, June 1 - 6, 2018.

Encoder The encoder is a bi-directional RNN whose hidden states represent tokens of the input sequence. These representations capture information not only of the corresponding token, but also other tokens in the sequence to leverage the context. The bi-directional RNN consists of two RNNs running in the left-to-right and right-to-left directions over the input sequence:

$$\begin{aligned}\overrightarrow{\boldsymbol{h}}_i &= \mathrm{RNN}(\overrightarrow{\boldsymbol{h}}_{i-1}, \boldsymbol{E}_S[x_i]) \\ \overleftarrow{\boldsymbol{h}}_i &= \mathrm{RNN}(\overleftarrow{\boldsymbol{h}}_{i+1}, \boldsymbol{E}_S[x_i])\end{aligned}$$

where $\boldsymbol{E}_S[x_i]$ is the embedding of the token x_i from the embedding table $\boldsymbol{E}_S$ of the input (source) space, and $\overrightarrow{\boldsymbol{h}}_i$ and $\overleftarrow{\boldsymbol{h}}_i$ are the hidden states of the forward and backward RNNs which can be based on the LSTM (long-short term memory) (Hochreiter and Schmidhuber, 1997) or GRU (gated-recurrent unit) (Chung et al., 2014) units. Each source token is then represented by the concatenation of the corresponding bidirectional hidden states, $\boldsymbol{h}_i = [\overrightarrow{\boldsymbol{h}}_i; \overleftarrow{\boldsymbol{h}}_i]$.

Decoder. The backbone of the decoder is a unidirectional RNN which generates the token of the output one-by-one from left to right. The generation of each token y_j is conditioned on all of the previously generated tokens $\boldsymbol{y}_{<j}$ via the state of the RNN decoder $\boldsymbol{s}_j$, and the input sequence via a *dynamic* context vector $\boldsymbol{c}_j$ (explained shortly):

$$\begin{aligned}y_j &\sim \mathrm{softmax}(\boldsymbol{W}_y \cdot \boldsymbol{r}_j + \boldsymbol{b}_r) && (1)\\ \boldsymbol{r}_j &= \tanh(\boldsymbol{s}_j + \boldsymbol{W}_{rc} \cdot \boldsymbol{c}_j + \boldsymbol{W}_{rj} \cdot \boldsymbol{E}_T[y_{j-1}]) && (2)\\ \boldsymbol{s}_j &= \tanh(\boldsymbol{W}_s \cdot \boldsymbol{s}_{j-1} + \boldsymbol{W}_{sj} \cdot \boldsymbol{E}_T[y_{j-1}] + \boldsymbol{W}_{sc} \cdot \boldsymbol{c}_j)\end{aligned}$$

where $\boldsymbol{E}_T[y_j]$ is the embedding of the token y_j from the embedding table $\boldsymbol{E}_T$ of the output (target) space, and the $\boldsymbol{W}$ matrices and $\boldsymbol{b}_r$ vector are the parameters.

A crucial element of the decoder is the *attention* mechanism which dynamically attends to relevant parts of the input sequence necessary for generating the next token in the output sequence. Before generating the next token t_j, the decoder computes the attention vector $\boldsymbol{\alpha}_j$ over the input token:

$$\begin{aligned}\boldsymbol{\alpha}_j &= \mathrm{softmax}(\boldsymbol{a}_j)\\ a_{ji} &= \boldsymbol{v} \cdot \tanh(\boldsymbol{W}_{ae} \cdot \boldsymbol{h}_i + \mathbf{W}_{at} \cdot \boldsymbol{s}_{j-1})\end{aligned}$$

which intuitively is similar to the notion of *alignment* in word/phrase-based statistical MT (Brown et al., 1993). The attention vector is then used to compute a fixed-length dynamic representation of the source sentence

$$\boldsymbol{c}_j = \sum_i \alpha_{ji} \boldsymbol{h}_i. \qquad (3)$$

which is conditioned upon in the RNN decoder when computing the next state or generating the output word (as mentioned above).

Training and Decoding. The model parameters are trained end-to-end by maximising the (regularised) log-likelihood of the training data

$$\arg\max_{\boldsymbol{\theta}} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}} \sum_{j=1}^{|\boldsymbol{y}|} \log P_{\boldsymbol{\theta}}(y_j|\boldsymbol{y}_{<j}, \boldsymbol{x})$$

where the above conditional probability is defined according to eqn (1). Usually drop-out is employed to prevent over-fitting on the training data. In the decoding time, the best output sequence for a given input sequence is produced by

$$\arg\max_{\boldsymbol{y}} P_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) = \prod_j P_{\boldsymbol{\theta}}(y_j|\boldsymbol{y}_{<j}\boldsymbol{x}).$$

Usually greedy decoding or beam search algorithms are employed to find an approximate solution, since solving the above optimisation problem exactly is computationally hard.

3 SEQ2SEQ Multi-Task Learning

We consider an extension of the basic SEQ2SEQ model where the encoder and decoder are equipped with deep stacked layers. Presumably, deeper layers capture more abstract information about a task, hence they can be used as a mechanism to share useful generalisable information among multiple tasks.

Deep Stacked Encoder. The deep encoder consists of multiple layers, where the hidden states in layer $\ell-1$ are the inputs to the hidden states at the next layer ℓ. That is,

$$\begin{aligned}\overrightarrow{\boldsymbol{h}}_i^{\ell} &= \overrightarrow{\mathrm{RNN}}^{\ell}_{\boldsymbol{\theta}_{\ell,enc}}(\overrightarrow{\boldsymbol{h}}_{i-1}^{\ell}, \boldsymbol{h}_i^{\ell-1})\\ \overleftarrow{\boldsymbol{h}}_i^{\ell} &= \overleftarrow{\mathrm{RNN}}^{\ell}_{\boldsymbol{\theta}_{\ell,enc}}(\overleftarrow{\boldsymbol{h}}_{i-1}^{\ell}, \boldsymbol{h}_i^{\ell-1})\end{aligned}$$

where $\boldsymbol{h}_i^{\ell} = [\overrightarrow{\boldsymbol{h}}_i^{\ell}; \overleftarrow{\boldsymbol{h}}_i^{\ell}]$ is the hidden state of the ℓ'th layer RNN encoder for the i'th source sentence word. The inputs to the first layer forward/backward RNNs are the source word embeddings $\boldsymbol{E}_S[x_i]$. The representation of the source sentence is then the concatenation of the hidden states for all layers $\boldsymbol{h}_i = [\boldsymbol{h}_i^1; \ldots; \boldsymbol{h}_i^L]$ which is then used by the decoder.

Deep Stacked Decoder. Similar to the multi-layer RNN encoder, the decoder RNN has multiple layers:

$$\boldsymbol{s}_j^\ell = \text{RNN}^\ell_{\boldsymbol{\theta}_{\ell,dec}}(\boldsymbol{s}_{j-1}^\ell, \boldsymbol{s}_j^{\ell-1})$$

where the inputs to the first layer RNNs are

$$\boldsymbol{W}_{sj} \cdot \boldsymbol{E}_T[y_{j-1}] + \boldsymbol{W}_{sc} \cdot \boldsymbol{c}_j$$

in which $\boldsymbol{c}_j$ is the dynamic source context, as defined in eqn 3. The state of the decoder is then the concatenation of the hidden states for all layers: $\boldsymbol{s}_j = [\boldsymbol{s}_j^1; \ldots; \boldsymbol{s}_j^L]$ which is then used in eqn 2 as part of the "output generation module".

Shared Layer MTL. We share the deep layer RNNs in the encoders and/or decoders across the tasks, as a mechanism to share abstract knowledge and increase model generalisation.

Suppose we have a total of $M+1$ tasks, consisting of the main task plus M auxiliary tasks. Let $\boldsymbol{\Theta}^m_{enc} = \{\boldsymbol{\theta}^m_{\ell,enc}\}_{\ell=1}^{L}$ and $\boldsymbol{\Theta}^m_{dec} = \{\boldsymbol{\theta}^m_{\ell',dec}\}_{\ell'=1}^{L'}$ be the parameters of multi-layer encoder and decoder for the task m. Let $\{\boldsymbol{\Theta}^m_{enc}, \boldsymbol{\Theta}^m_{dec}\}_{m=1}^{M}$ and $\{\boldsymbol{\Theta}^0_{enc}, \boldsymbol{\Theta}^0_{dec}\}$ be the RNN parameters for the auxiliary tasks and the main task, respectively. We share the parameters of the deep-level encoders and decoders of the auxiliary tasks with those of the main task. That is,

$$\begin{aligned} \forall m \in [1,..,M]\ \forall \ell \in [L^m_{enc},..,L] \quad &: \quad \boldsymbol{\theta}^m_{\ell,enc} = \boldsymbol{\theta}^0_{\ell,enc} \\ \forall m \in [1,..,M]\ \forall \ell' \in [L'^m_{dec},..,L'] \quad &: \quad \boldsymbol{\theta}^m_{\ell',dec} = \boldsymbol{\theta}^0_{\ell',dec} \end{aligned}$$

where L^m_{enc} and L'^m_{dec} specify the deep-layer RNNs need to be shared parameters. Other parameters to share across the tasks include those of the attention module, the source/target embedding tables, and the output generation module. As an extreme case, we can share *all* the parameters of SEQ2SEQ architectures across the tasks.

Training Objective. Suppose we are given a collection of $M+1$ SEQ2SEQ transductions tasks, each of which is associated with a training set $\mathcal{D}_m := \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N_m}$. The parameters are learned by maximising the MTL training objective:

$$\mathcal{L}_{mtl}(\boldsymbol{\Theta}_{mtl}) := \sum_{m=0}^{M} \frac{\gamma_m}{|\mathcal{D}_m|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}_m} \log P_{\boldsymbol{\Theta}_m}(\boldsymbol{y}|\boldsymbol{x}) \tag{4}$$

where $\boldsymbol{\Theta}_{mtl}$ denotes all the parameters of the MTL architecture, $|\mathcal{D}_m|$ denotes the size of the training set for the task m, and γ_m balances out its influence in the training objective.

Training Schedule. Variants of stochastic gradient descent (SGD) can be used to optimise the objective in order to learn the parameters. Making the best use of tasks with different objective geometries is challenging, e.g. due to the scale of their gradients. One strategy for making an SGD update is to select the tasks from which the next data items should be chosen. In our training schedule, we randomly select a training data item from the main task, and pair it with a data item selected from a randomly selected auxiliary task for making the next SGD update. This ensures the presence of training signal from the main task in all SGD updates, and avoids the training signal being washed out by the auxiliary tasks.

4 Adversarial Training

The learned shared knowledge can be contaminated by task-specific information. We address this issue by adding an adversarial objective. The basic idea is to augment the MTL training objective with additional terms, so that the identity of a task cannot be predicted from its data items by the representations resulted from the shared encoder/decoder RNN layers.

Task Discriminator. The goal of the task discriminator is to predict the identity of a task for a data item based on the representations of the share layers. More specifically, our task discriminator consists of two RNNs with LSTM units, each of which encodes the sequence of hidden states in the shared layers of the encoder and the decoder.[1] The last hidden states of these two RNNs are then concatenated, giving rise to a fixed dimensional vector summarising the representations in the shared layers. The summary vector is passed through a fully connected layer followed by a $\mathrm{softmax}$ to predict the probability distribution over the tasks:

$$\begin{aligned} P_{\boldsymbol{\Theta}_d}(\text{task id}|\boldsymbol{h}_d) &\sim \mathrm{softmax}(\boldsymbol{W}_d\boldsymbol{h}_d + \boldsymbol{b}_d) \\ \boldsymbol{h}_d &:= \mathrm{disLSTMs}(\mathrm{shrRep}_{\boldsymbol{\Theta}_{mtl}}(\boldsymbol{x}, \boldsymbol{y})) \end{aligned}$$

where $\mathrm{disLSTMs}$ denotes the discriminator LSTMs, $\mathrm{shrRep}_{\boldsymbol{\Theta}_{mtl}}(\boldsymbol{x}, \boldsymbol{y})$ denotes the representations in the shared layer of deep encoders and decoders in the MTL architecture, and $\boldsymbol{\Theta}_d$ includes the $\mathrm{disLSTMs}$ parameters as well as $\{\boldsymbol{W}_d, \boldsymbol{b}_d\}$.

[1] When multiple layers are shared, we concatenate their hidden states at each time step, which is then input to the task discriminator's LSTMs.

Adversarial Objective. Inspired by (Chen et al., 2017), we add two additional terms to the MTL training objective in eqn 4. The first term is $\mathcal{L}_{adv1}(\mathbf{\Theta}_d)$ defined as:

$$\sum_{m=0}^{M} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}_m} \log P_{\mathbf{\Theta}_d}(m|\,\text{disLSTMs}(\text{shrRep}_{\mathbf{\Theta}_{mtl}}(\boldsymbol{x},\boldsymbol{y}))).$$

Maximising the above objective over $\mathbf{\Theta}_d$ ensures proper training of the discriminator to predict the identity of the task. The second term ensures that the parameters of the shared layers are trained so that they confuse the discriminator by maximising the entropy of its predicted distribution over the task identities. That is, we add the term $\mathcal{L}_{adv2}(\mathbf{\Theta}_{mtl})$ to the training objective defined as:

$$\sum_{m=0}^{M} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}_m} H\big[P_{\mathbf{\Theta}_d}(.|\,\text{disLSTMs}(\text{shrRep}_{\mathbf{\Theta}_{mtl}}(\boldsymbol{x},\boldsymbol{y})))\big]$$

where $H[.]$ is the entropy of a distribution. In summary, the adversarial training leads to the following optimisation

$$\arg\max_{\mathbf{\Theta}_d,\mathbf{\Theta}_{mtl}} \mathcal{L}_{mtl}(\mathbf{\Theta}_{mtl})+\mathcal{L}_{adv1}(\mathbf{\Theta}_d)+\lambda\mathcal{L}_{adv2}(\mathbf{\Theta}_{mtl}).$$

We maximise the above objective by SGD, and update the parameters by alternating between optimising $\mathcal{L}_{mtl}(\mathbf{\Theta}_{mtl}) + \lambda\mathcal{L}_{adv2}(\mathbf{\Theta}_{mtl})$ and $\mathcal{L}_{adv1}(\mathbf{\Theta}_d)$.

5 Experiments

5.1 Bilingual Corpora

We use three language-pairs, translating from English to French, Farsi, and Vietnamese. We have chosen these languages to analyse the effect of multi-task learning on languages with different underlying linguistic structures. The sentences are segmented using BPE (Sennrich et al., 2016) on the union of source and target vocabularies for English-French and English-Vietnamese. For English-Farsi, BPE is performed using separate vocabularies due to the disjoint alphabets. We use a special <UNK> token to replace unknown BPE units in the test and development sets.

Table 1 show some statistics about the bilingual corpora. Further details about the corpora and their pre-processing is as follows:

- The English-French corpus is a random subset of EuroParlv7 as distributed to WMT2014. Sentence pairs in which either the source or the target has length more than 80 (before applying BPE) have been removed. The BPE is performed with a 30k total vocabulary size. The "news-test2012" and "news-test-2013" portions are used for validation and test sets, respectively.

	Train	Dev	Test
En → Fr	98,846	5,357	5,357
En → Fa	98,158	3,000	4,000
En → vi	133,290	1,553	1,268

Table 1: The statistics of bilingual corpora.

- The English-Farsi corpus is assembled from all the parallel news text in LDC2016E93 *Farsi Representative Language Pack* from the Linguistic Data Consortium, combined with English-Farsi parallel subtitles from the TED corpus (Tiedemann, 2012). Since the TED subtitles are user-contributed, this text contained considerable variation in the encoding of its Perso-Arabic characters. To address this issue, we have normalized the corpus using the Hazm toolkit[2]. Sentence pairs in which one of the sentences has more than 80 (before applying BPE) are removed, and BPE is performed with a 30k vocabulary size. Random subsets of this corpus (3k and 4k sentences each) are held out as validation and test sets, respectively.

- The English-Vietnamese is from the translation task in IWSLT 2015, and we use the preprocessed version provided by (Luong and Manning, 2015). The sentence pairs in which at least one of their sentences had more than 300 units (after applying BPE) are removed. "tst2012" and "tst2013" parts are used for validation and test sets, respectively.

5.2 Auxiliary Tasks

We have chosen the following auxiliary tasks to provide the NMT model with syntactic and/or semantic knowledge, in order to enhance the quality of translation:

Named-Entity Recognition (NER). With a small bilingual training corpus, it would be hard for the NMT model to learn how to translate rarely occurring named-entities. Through the NER task,

[2] www.sobhe.ir/hazm

the model hopefully learns the skill to recognize named entities. Speculatively, it would then enables leaning translation patterns by masking out named entities. The NER data comes from the CONLL shared task.[3]

Syntactic Parsing. This task enables NMT to learn the phrase structure of the input sentence, which would then be useful in better re-orderings. This would be most useful for language pairs with high syntactic divergence. The parsing data comes from the Penn Tree Bank with the standard split for training, development, and test (Marcus et al., 1993). We linearise the constituency trees, in order to turn syntactic parsing as a SEQ2SEQ transduction (Vinyals et al., 2015).

Semantic Parsing. A good translation should preserve the meaning. Learning from the semantic parsing task enables the NMT model to pay attention to a meaning abstraction of the source sentence, in order to convey it to the target translation. We have made use of the Abstract Meaning Representation (AMR) corpus Release 2.0 (LDC2017T10), which pairs English sentences AMR meaning graphs. We linearise the AMR graphs, in order to convert semantic parsing as a SEQ2SEQ transduction problem (Konstas et al., 2017).

5.3 Models and Baselines

We have implemented the proposed multi-task learning architecture in C++ using DyNet (Neubig et al., 2017), on top of Mantis (Cohn et al., 2016) which is an implementation of the attentional SEQ2SEQ NMT model in (**?**). In our multi-task architecture, we do partial sharing of parameters, where the parameters of the top 2 stacked layers are shared among the encoders of the tasks. Moreover, we share the parameters of the top layer stacked decoder among the tasks. Source and target embedding tables are shared among the tasks, while the attention component is task-specific. [4] We compare against the following baselines:

- Baseline 1: The vanila SEQ2SEQ model without any multi-tasking.
- Baseline 2: The multi-tasking architecture proposed in (Niehues and Cho, 2017), which is a special case of our approach where the parameters of all 3 stacked layers are shared among the tasks.[5] They have not used deep stacked layers in encoder and decoder as we do, so we extend their work to make it comparable with ours.

The configuration of models is as follows. The encoders and decoders make use of GRU units with 400 hidden dimensions, and the attention component has 200 dimensions. For training, we used Adam algorithm (Kingma and Ba, 2014) with the initial learning rate of 0.003 for all of the tasks. Learning rates are halved when the performance on the corresponding dev set decreased. In order to speed-up the training, we use mini-batching with the size of 32. Dropout rates for both encoder and decoder are set to 0.5, and models are trained for 50 epochs where the best models is selected based on the perplexity on the dev set. λ for the adversarial training is set to 0.5. Once trained, the NMT model translates using the greedy search. We use BLEU (Papineni et al., 2002) to measure translation quality. [6]

5.4 Results

Table 2 reports the BLEU scores and perplexities for the baseline and our proposed method on the three aforementioned translation tasks. It can be seen that the performance of multi-task learning models are better than Baseline 1 (only MT task). This confirms that adding auxiliary tasks helps to increase the performance of the machine translation task.

As expected, the effect of different tasks are not similar across the language pairs, possibly due to the following reasons: (i) these translation tasks datasets come from different domains so they have various degree of domain relatedness to the auxiliary tasks, and (ii) the BLEU scores of the Baseline 1 show that the three translation models are on different quality levels which may entail that they benefit from auxiliary knowledge on different levels. In order to improve a model with low quality translations due to language divergence, syntactic knowledge can be more helpful as they help better reorderings. In a higher-quality model, however, semantic knowledge can be more useful as

[3] https://www.clips.uantwerpen.be/conll2003/ner

[4] In our experiments, models with task-specific attention components achieved better results than those sharing them.

[5] We have used their best performing architecture and changed the training schedule to ours.

[6] With "multi-bleu.perl" script from Moses (Koehn et al., 2007).

	English → French				English → Farsi				English → Vietnamese			
	Dev		Test		Dev		Test		Dev		Test	
	PPL	BLEU	PPL	BLEU	PPL	BLEU	PPL	BLEU	PPL	BLEU	PPL	BLEU
NMT	117.27	8.85	80.29	10.71	86.63	7.69	87.94	7.46	23.24	16.53	20.36	17.86
+ Semantic	71.7	10.58	51.2	12.72	56.32	8.3	57.88	8.32	14.86	19.96	12.79	21.82
+ NER	73.42	10.73	52.07	12.92	48.46	9.11	49.53	9.03	15.04	20.2	13.13	21.96
+ Syntactic	69.45	11.88	48.9	13.94	44.35	9.73	45.37	9.37	16.42	18.4	14.27	20.4
+ All Tasks	69.71	11.3	49.86	13.41	44.03	**9.68**	45.1	**9.7**	14.79	20.12	12.65	22.41
+ All+Adv.	68.44	**11.93**	48.92	**14.02**	45.25	9.55	45.87	9.19	14.19	**21.21**	12.11	**23.54**

Table 2: BLEU scores and perplexities of the baseline vs our MTL architecture with various auxiliary tasks on the full bilingual datasets.

	W/O Adaptation		W/ Adaptation		
	Partial	Full	Partial	Part.+Adv.	Full
En→Fr	13.41	9.94	14.86	15.12	11.94
En → Fa	9.7	7.89	10.31	10.08	8.6
En → Vi	22.41	20.26	23.35	24.28	21.67

Table 3: Our method (partial parameter sharing) against Baseline 2 (full parameter sharing).

a higher-level linguistic knowledge. This pattern can be seen in the reported results: syntactic parsing leads to more improvement on Farsi translation which has a low BLEU score and high language divergence to English, and semantic parsing yields more improvement on the Vietnamese translation task which already has a high BLEU score. The NER task has led to a steady improvement in all the translation tasks, as it leads to better handling of named entities.

We have further added adversarial training to ensure the shared representation learned by the encoder is not contaminated by the task-specific information. The results are in the last row of Table 2. The experiments show that adversarial training leads to further gains in MTL translation quality, except when translating into Farsi. We speculate this is due to the low quality of NMT for Farsi, where updating shared parameters with respect to the entropy of discriminator's predicted distribution may negatively affect the model.

Table 3 compares our multi-task learning approach to Baseline 2. As Table 3, our partial parameter sharing mechanism is more effective than fully sharing the parameters (Baseline 2), due to its flexibility in allowing access to private task-specific knowledge. We also applied the adaptation technique (Niehues and Cho, 2017) as follows. Upon finishing MTL training, we continue to train only on the MT task for another 20 epochs, and choose the best model based on perplexity on dev set. Adaptation has led to consistent gains in the performance of our MTL architecture and Baseline 2.

5.5 Analysis

How many layers of encoder/decoder to share? Figure 2 show the results of changing the number of shared layers in encoder and decoder based on the En→Vi translation task. The results confirm that partial sharing of stacked layers is better than full sharing. Intuitively, partial sharing provides the model with an opportunity to learn task specific skills via the private layers, while leveraging the knowledge learned from other tasks via shared layers.

Statistics of gold n-grams in MTL translations. Generating high order gold n-grams is hard. We analyse the effect of syntactic and semantic knowledge on generating gold n-grams in translations.

For each sentence, we first extract n-grams in the gold translation, and then compute the number of n-grams which are common with the generated translations. Finally, after aggregating the results over the entire test set, we compute the percentage of additional gold n-grams generated by each MTL model compared to the ones in single-task MT model. The results are depicted in Figure 1. Interestingly, the MTL models generate more correct n-grams relative to the vanilla NMT model, as n increases.

Effect of the NER task. The NMT model has difficulty translating rarely occurring named-entities, particularly when the bilingual parallel data is scarce. We expect learning from the NER task leads the MTL model to recognize named-entities and learn underlying patterns for translating them. The top part in Table 4 shows an example of such situation. As seen, the MTL is able to recognize all of the named-entities in the sentence and translate the while the single-task model

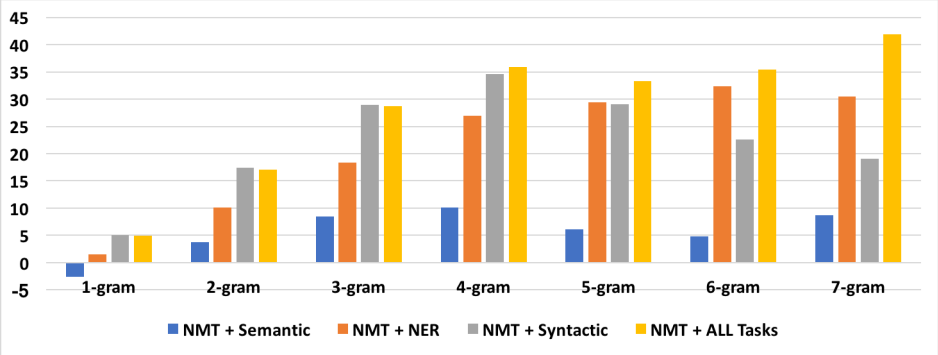


Figure 1: Percentage of more correct n-grams generated by the deep MTL models compared to the single-task model (only MT).

English	this is a building in Hyderabad , India .
Reference	this a building in Hyderabad is , in India .
MT only model	this a building in Hyderabad is .
MT+NER model	this a building in Hyderabad India is .
English	we see people on our screens .
Reference	we people on television screen or cinema see .
MT only model	we people see we people .
MT+semantic model	we people on television screen see .
English	in hospitals , for new medical instruments ; in streets for traffic control .
Reference	in hospitals , for instruments medical new ; in streets for control traffic
MT only model	in hospitals , for tools new tools for traffic controlled* [7] .
MT+syntactic model	in hospitals , for devices new , in streets for control traffic .

Table 4: Example of translations on Farsi test set. In this examples each Farsi word is replaced with its English translation, and the order of words is reversed (Farsi is written right-to-left). The structure of Farsi is Subject-Object-Verb (SOV), leading to different word orders in English and Reference sentences.

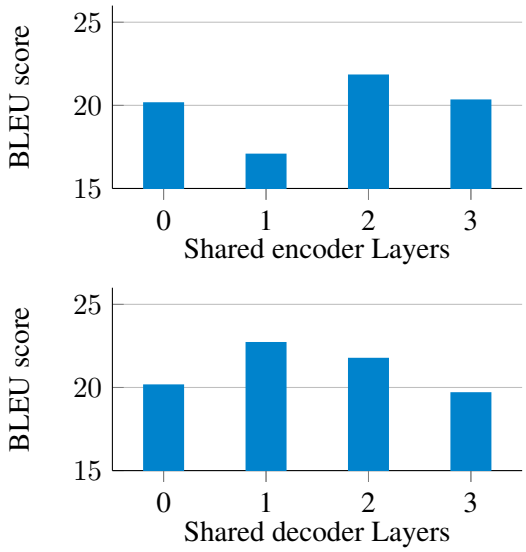


Figure 2: BLEU scores for different numbers of shared layers in (top) encoder while no layer is shared in decoder, and (bottom) decoder while no layer is shared in encoder

missed "India".

For more analysis, we have applied a Farsi POS tagger (Feely et al., 2014) to gold translations. Then, we extracted n-grams with at least one noun in them, and report the statistics of correct such n-grams, similar to what reported in Figure 1. The resulting statistics is depicted in Figure 3. As seen, the MTL model trained on MT and NER tasks leads to generation of more correct unigram noun phrases relative to the vanilla NMT, as n increases.

Effect of the semantic parsing task. Semantic parsing encourages a precise understanding of the source text, which would then be useful for conveying the correct meaning to the translation. The middle part in Table 4 is an example translation, showing that semantic parsing has helped NMT by understanding that "the subject sees the object via subject's screens".

Effect of the syntactic parsing task. Recognizing the syntactic structure of the source sentence helps NMT to better translate phrases. The bottom part of Table 4 shows an example translation demonstrating such case. The source sentence is talking about "a method for controlling the

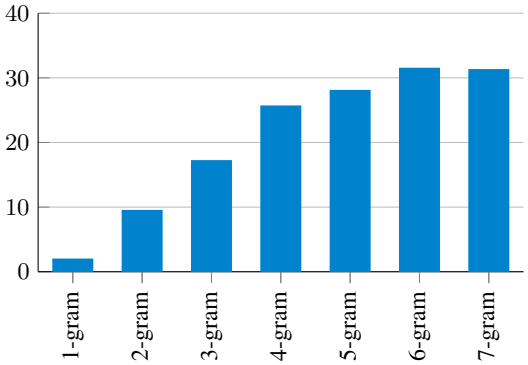


Figure 3: Percentage of more corrected n-grams with at least one noun generated by MT+NER model compared with the only MT model (only MT).

traffic", which is correctly translated by the MTL model while vanilla NMT has mistakenly translated it to "controlled traffic".

6 Related Work

Multi-task learning has attracted attention to improve NMT in recent work. (Zhang and Zong, 2016) has made use of monolingual data in the source language in a multitask learning framework by sharing encoder in the attentional encoder-decoder model. Their auxiliary task is to reorder the source text to make it close to the target language word order. (Domhan and Hieber, 2017) proposed a two-layer stacked decoder, which the bottom layer is trained on language modelling on the target language text. The next word is jointly predicted by the bottom layer language model and the top layer attentional RNN decoder. They reported only moderate improvements over the baseline and fall short against using synthetic parallel data. (Dalvi et al., 2017) investigated the amount of learned morphology and how it can be injected using MTL. Our method is related to what they call joint data-learning, where they share all of the SEQ2SEQ components among the tasks.

(Belinkov et al., 2017a; Shi et al., 2016; Belinkov et al., 2017b) investigate syntax/semantics phenomena learned as a byproduct of SEQ2SEQ NMT training. We, in turn, investigate the effect of injecting syntax/semantic on learning NMT using MTL.

The closet work to ours is (Niehues and Cho, 2017), which has made use of part-of-speech tagging and named-entity recognition tasks to improve NMT. They have used the attentional encoder-decoder with a shallow architecture, and share different parts eg the encoder, decoder, and attention. They report the best performance with fully sharing the encoder. In contrast, our architecture uses partial sharing on deep stacked encoder and decoder components, and the results show that it is critical for NMT improvement in MTL. Furthermore, we propose adversarial training to prevent contamination of shared knowledge with task specific details.

Taking another approach to MTL, (Søgaard and Goldberg, 2016) and (Hashimoto et al., 2017) have proposed architectures by stacking up tasks on top of each other according to their linguistic level, eg from lower level tasks (POS tagging) to higher level tasks (parsing). In this approach, each task uses predicted annotations and hidden states of the lower-level tasks for making a better prediction. This is contrast to the approach taken in this paper where models with shared parameters are trained jointly on multiple tasks.

More broadly, deep multitask learning has been used for various NLP problems, including graph-based parsing (Chen and Ye, 2011) and keyphrase boundary classification (Augenstein and Søgaard, 2017) . (Chen et al., 2017) has applied multi-task learning for Chinese word segmentation, and (Liu et al., 2017) applied it for text classification problem. Both of these works have used adversarial training to make sure the shared layer extract only common knowledge.

MTL has been used effectively to learn from multimodal data. (Luong et al., 2016) has proposed MTL architectures for neural SEQ2SEQ transduction for tasks including MT, image caption generation, and parsing. They fully share the encoders (many-to-one), the decoders (one-to-many), or some of the encoders and decoders (many-to-many). (Pasunuru and Bansal, 2017) have made use of an MTL approach to improve video captioning with auxiliary tasks including video prediction and logical language entailment based on a many-to-many architecture.

7 Conclusions and Future Work

We have presented an approach to improve NMT in bilingually scarce scenarios, by leveraging curated linguistic resources in the source, including semantic parsing, syntactic parsing, and named entity recognition. This is achieved via an effective MTL architecture, based on deep stacked en-

coders and decoders, to share common knowledge among the MT and auxiliary tasks. Our experimental results show substantial improvements in the translation quality, when translating from English to French, Vietnamese, and Farsi in bilingually scarce scenarios. For future work, we would like to investigate architectures which allow automatic parameter tying among the tasks (Ruder et al., 2017).

Acknowledgments

We are very grateful to the members of the JSALT–2017 workshop at CMU, particularly George Foster, Colin Cherry, Patrick Littell, David Mortensen, Graham Neubig, Ji Xin, Daniel Beck, Anna Currey, Vu Hoang, and Gaurav Kumar for the insightful discussions and data pre-processing. This work was supported by computational resources from the Multi-modal Australian ScienceS Imaging and Visualisation Environment (MASSIVE) at Monash University, Amazon, Extreme Science and Engineering Discovery Environment (supported by the NSF grant number OCI-1053575), and the Bridges system (supported by the NSF award number ACI-1445606) at the Pittsburgh Supercomputing Center. This work was supported by the Australian Research Council via DP160102686.

References

Isabelle Augenstein and Anders Søgaard. 2017. Multi-task learning of keyphrase boundary classification. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. pages 341–346.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of the International Conference on Learning Representations*.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017a. What do neural machine translation models learn about morphology? In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. pages 861–872.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2017b. Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. In *Proceedings of the International Joint Conference on Natural Language Processing*. pages 1–10.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics* 19(2):263–311.

Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-criteria learning for chinese word segmentation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. pages 1193–1203.

Y. Chen and X. Ye. 2011. Projection Onto A Simplex . *arXiv preprint arXiv:1101.6081* .

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. NIPS Workshop on Deep Learning.

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating Structural Alignment Biases into an Attentional Neural Translation Model. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics–Human Language Technologies*. pages 876–885.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, and Stephan Vogel. 2017. Understanding and improving morphological learning in the neural machine translation decoder. In *Proceedings of the International Joint Conference on Natural Language Processing*. pages 142–151.

Tobias Domhan and Felix Hieber. 2017. Using target-side monolingual data for neural machine translation through multi-task learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. pages 1501–1506.

Weston Feely, Mehdi Manshadi, Robert E Frederking, and Lori S Levin. 2014. The CMU METAL Farsi NLP Approach. In *LREC*. pages 4052–4055.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple NLP tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. pages 1923–1933.

Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9(8):1735–1780.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open

Source Toolkit for Statistical Machine Translation. In *Proceedings of the Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*. pages 177–180.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*. pages 28–39.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural amr: Sequence-to-sequence models for parsing and generation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. pages 146–157.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. *arXiv preprint arXiv:1704.05742* .

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford neural machine translation systems for spoken language domain. In *International Workshop on Spoken Language Translation*. Da Nang, Vietnam.

Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *International Conference on Learning Representations*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1412–1421.

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

G. Neubig, C. Dyer, Y. Goldberg, A. Matthews, W. Ammar, A. Anastasopoulos, M. Ballesteros, D. Chiang, D. Clothiaux, T. Cohn, K. Duh, M. Faruqui, C. Gan, D. Garrette, Y. Ji, L. Kong, A. Kuncoro, G. Kumar, C. Malaviya, P. Michel, Y. Oda, M. Richardson, N. Saphra, S. Swayamdipta, and P. Yin. 2017. DyNet: The Dynamic Neural Network Toolkit. *ArXiv preprints arXiv:1701.03980* .

Jan Niehues and Eunah Cho. 2017. Exploiting linguistic resources for neural machine translation using multi-task learning. In *Proceedings of the Second Conference on Machine Translation*. pages 80–89.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the Annual Meeting on Association for Computational Linguistics*. pages 311–318.

Ramakanth Pasunuru and Mohit Bansal. 2017. Multi-task video captioning with video and entailment generation. In *Proceedings of ACL*.

Hao Peng, Sam Thomson, and Noah A Smith. 2017. Deep multitask learning for semantic dependency parsing. *arXiv preprint arXiv:1704.06855* .

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2017. Sluice networks: Learning what to share between loosely related tasks. *arXiv preprint arXiv:1705.08142* .

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. pages 1715–1725.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural mt learn source syntax? In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. pages 1526–1534.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. pages 231–235.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In *Proceedings of the International Conference on Language Resources and Evaluation*. pages 2214–2218.

Oriol Vinyals, Ł ukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pages 2773–2781.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. pages 1535–1545.

Self-Attentive Residual Decoder for Neural Machine Translation

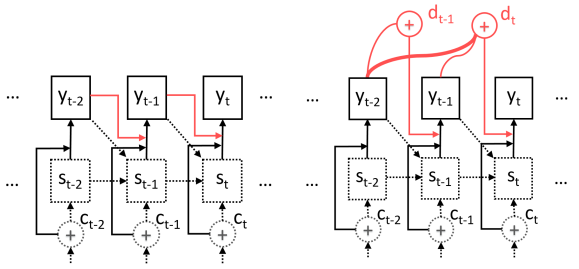
Lesly Miculicich Werlen[*,†], **Nikolaos Pappas**[*], **Dhananjay Ram**[*,†],
Andrei Popescu-Belis[‡]
[*]Idiap Research Institute, Switzerland,
[†]École polytechnique fédérale de Lausanne (EPFL), Switzerland,
[‡]HEIG-VD / HES-SO, Switzerland
{lmiculicich, npappas, dram}@idiap.ch
andrei.popescu-belis@heig-vd.ch

Abstract

Neural sequence-to-sequence networks with attention have achieved remarkable performance for machine translation. One of the reasons for their effectiveness is their ability to capture relevant source-side contextual information at each time-step prediction through an attention mechanism. However, the target-side context is solely based on the sequence model which, in practice, is prone to a recency bias and lacks the ability to capture effectively non-sequential dependencies among words. To address this limitation, we propose a target-side-attentive residual recurrent network for decoding, where attention over previous words contributes directly to the prediction of the next word. The residual learning facilitates the flow of information from the distant past and is able to emphasize any of the previously translated words, hence it gains access to a wider context. The proposed model outperforms a neural MT baseline as well as a memory and self-attention network on three language pairs. The analysis of the attention learned by the decoder confirms that it emphasizes a wider context, and that it captures syntactic-like structures.

1 Introduction

Neural machine translation (NMT) has recently become the state-of-the-art approach to machine translation (Bojar et al., 2016). Several architectures have been proposed for this task (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014; Gehring et al., 2017; Vaswani et al., 2017), but the attention-based NMT model designed by Bahdanau et al. (2015) is still considered the de-facto baseline. This architecture is composed of two recurrent neural networks (RNNs), an encoder and a decoder, and an attention mechanism between them for modeling a soft word-alignment. First, the model encodes the complete source sentence, and then decodes one word at a time. The decoder has access to all the context on the source side through the attention mechanism. However, on the target side, the contextual information is represented only through a fixed-length vector, namely the hidden state of the decoder. As observed by Bahdanau et al. (2015), this creates a bottleneck which hinders the ability of the sequential model to learn longer-term information effectively.



(a) Baseline NMT decoder (b) Self-attentive residual dec.

Figure 1: Comparison between the decoder of the baseline NMT and the proposed decoder with self-attentive residual connections.

As pointed out by Cheng et al. (2016), sequential models present two main problems for natural language processing. First, the memory of the encoder is shared across multiple words and is prone to bias towards the recent past. Second, such models do not fully capture the structural composition of language. To address these limitations, several recent models have been proposed, namely memory networks (Cheng et al., 2016; Tran et al., 2016; Wang et al., 2016) and self-attention networks (Daniluk et al., 2016; Liu and Lapata, 2018). We experimented with these methods, applying them to NMT: *memory RNN* (Cheng et al., 2016) and *self-attentive RNN* (Daniluk et al., 2016). How-

Proceedings of NAACL-HLT 2018, pages 1366–1379
New Orleans, Louisiana, June 1 - 6, 2018.

ever, we observed no significant gains in performance over the baseline architecture.

In this paper, we propose a self-attentive residual recurrent decoder, presented in Figure 1b, which, if unfolded over time, represents a densely-connected residual network. The self-attentive residual connections focus selectively on previously translated words and propagate useful information to the output of the decoder, within an attention-based NMT architecture. The attention paid to the previously predicted words is analogous to a read-only memory operation, and enables the learning of syntactic-like structures which are useful for the translation task.

Our evaluation on three language pairs shows that the proposed model improves over several baselines, with only a small increase in computational overhead. In contrast, other similar approaches have lower scores but a higher computational overhead. The contributions of this paper can be summarized as follows:

- We propose and compare several options for using self-attentive residual learning within a standard decoder, which facilitates the flow of contextual information on the target side.
- We demonstrate consistent improvements over a standard baseline, and two advanced variants, which make use of memory and self-attention on three language pairs (English-to-Chinese, Spanish-to-English, and English-to-German).
- We perform an ablation study and analyze the learned attention function, providing additional insights on its actual contributions.

2 Related Work

Several studies have been proposed to enhance sequential models by capturing longer contexts. Long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) is the most commonly used recurrent neural network (RNN), because its internal memory allows to retain information from a more distant past than a vanilla RNN. Several studies attempt to increase the memory capacity of LSTMs by using memory networks (Weston et al., 2015; Sukhbaatar et al., 2015). For instance, Cheng et al. (2016) incorporate different memory cells for each previous output representation, which are later accessed by an attention mechanism. Tran et al. (2016) include a memory block to access recent input words in a selective manner. Both methods show improvements on language modeling. For NMT, Wang et al. (2016) presented a decoder enhanced with an external shared memory. Memory networks extend the capacity of the network and have the potential to read, write, and forget information. Our method, which attends over previously predicted words, can be seen as a read-only memory, which is simpler but computationally more efficient because it does not require additional memory space.

Other studies aim to improve the modeling of source-side contextual information, for example through a context-aware encoder using self-attention (Zhang et al., 2017), or a recurrent attention NMT (Yang et al., 2017) that is aware of previously attended words on the source-side in order to better predict which words will be attended in future. Additionally, variational NMT (Zhang et al., 2016a) introduces a latent variable to model the underlying semantics of source sentences. In contrast to these studies, we focus instead on the contextual information *on the target side*.

The application of self-attention mechanisms to RNNs have been previously studied, and in general, they seem to capture syntactic dependencies among distant words (Liu and Lapata, 2018; Soltani and Jiang, 2016; Lee et al., 2017; Lin et al., 2017). Daniluk et al. (2016) explore different approaches to self-attention for language modeling, leading to improvements over a baseline LSTM and over memory-augmented methods. However, the methods do not fully utilize a longer context. The main difference with our approach is that we apply attention on the output embeddings rather than the hidden states. Thus, the connections are independent of the recurrent layer representations, which is beneficial to NMT, as we show below.

Our model relies on residual connections, which have been shown to improve the learning process of deep neural networks by addressing the vanishing gradient problem (He et al., 2016). These connections create a direct path from previous layers, helping the transmission of information. Recently, several architectures using residual connections with LSTMs have been proposed for sequence prediction (Zhang et al., 2016b; Kim et al., 2017; Zilly et al., 2017; Wang and Tian, 2016). To our knowledge, our study is the first one to use self-attentive residual connections within residual RNNs for NMT. In parallel to our study, a similar method was recently proposed for sentiment analysis (Wang, 2017).

3 Background: Neural Machine Translation

Neural machine translation aims to compute the conditional distribution of emitting a sentence in a target language given a sentence in a source language, denoted by $p_\Theta(y|x)$, where Θ is the set of parameters of the neural model, and $y = \{y_1, ..., y_n\}$ and $x = \{x_1, ..., x_m\}$ are respectively the representations of source and target sentences as sequences of words. The parameters Θ are learned by training a sequence-to-sequence neural model on a corpus of parallel sentences. In particular, the learning objective is to maximize the following conditional log-likelihood:

$$\max_{\Theta} \frac{1}{N} \sum_{n=1}^{N} \log(p_\Theta(y|x)) \quad (1)$$

The models typically use gated recurrent units (GRUs) (Cho et al., 2014) or LSTMs (Hochreiter and Schmidhuber, 1997). Their architecture has three main components: an encoder, a decoder, and an attention mechanism.

The goal of the encoder is to build meaningful representations of the source sentences. It consists of a bidirectional RNN which includes contextual information from past and future words into the vector representation h_i of a particular word vector x_i, formally defined as follows:

$$h_i = [\overrightarrow{h_i}, \overleftarrow{h_i}] \quad (2)$$

Here, $\overrightarrow{h_i} = f(x_i, h_{i-1})$ and $\overleftarrow{h_i} = f(x_i, h_{i+1})$ are the hidden states of the forward and backward passes of the bidirectional RNN respectively, and f is a non-linear function.

The decoder (see Figure 1a) is in essence a recurrent language model. At each time step, it predicts a target word y_t conditioned over the previous words and the information from the encoder using the following posterior probability:

$$p(y_t|y_1, ..., y_{t-1}, c_t) \approx g(s_t, y_{t-1}, c_t) \quad (3)$$

where g is a non-linear multilayer function. The hidden state of the decoder s_t is defined as:

$$s_t = f(s_{t-1}, y_{t-1}, c_t) \quad (4)$$

and depends on a *context vector* c_t that is computed by the attention mechanism.

The attention mechanism allows the decoder to select which parts of the source sentence are more useful to predict the next output word. This goal is achieved by considering a weighted sum over all hidden states of the encoder as follows:

$$c_t = \sum_{i=1}^{m} \alpha_i^t h_i \quad (5)$$

where α_i^t is a weight calculated using a normalized exponential function a, also known as *alignment function*, which computes how good is the match between the input at position $i \in \{1, ..., n\}$ and the output at position t:

$$\alpha_i^t = softmax(e_i^t) \quad (6)$$

$$e_i^t = a(s_{t-1}, h_i) \quad (7)$$

Different types of alignment functions have been used for NMT, as investigated by Luong et al. (2015). Here, we use the one originally defined by Bahdanau et al. (2015).

4 Self-Attentive Residual Decoder

The decoder of the attention-based NMT model uses a skip connection from the previously predicted word to the output classifier in order to enhance the performance of translation. As we can see in Eq. (3), the probability of a particular word is calculated by a function g which takes as input the hidden state of the recurrent layer s_t, the representation of the previously predicted word y_{t-1}, and the context vector c_t. Within g, these quantities are typically summed up after going through simple linear transformations, hence the addition of y_{t-1} is indeed a skip connection as in residual networks (He et al., 2016). In theory, s_t should be sufficient for predicting the next word given that it is dependent on the other two local-context components according to Eq. (4). However, the y_{t-1} quantity makes the model emphasize the last predicted word for generating the next word. How can we make the model consider a broader context?

To answer this question, we propose to include into the decoder's formula skip connections not only from the previous time step y_{t-1}, but from all previous time steps from y_0 to y_{t-1}. This defines a residual recurrent network which, unfolded over time, can be seen as a densely connected residual network. These connections are applied to all previously predicted words, and reinforce the memory of the recurrent layer towards what has been translated so far. At each time step, the model

decides which of the previously predicted words should be emphasized to predict the next one. In order to deal with the dynamic length of this new input, we use a target-side summary vector d_t that can be interpreted as the representation of the decoded sentence until the time t in the word embedding space. We therefore modify Eq. (3) replacing y_{t-1} with d_t:

$$p(y_t|y_1, ..., y_{t-1}, c_t) \approx g(s_t, d_t, c_t) \quad (8)$$

The replacement of y_{t-1} with d_t means that the number of parameters added to the model is dependent only on the calculation of d_t. Figure 1b illustrates the change made to the decoder. We define two methods for summarizing the context into d_t, which are described in the following sections.

4.1 Mean Residual Connections

One simple way to aggregate information from multiple word embeddings is by averaging them. This average can be seen as the sentence representation until time t. We hypothesize that this representation is more informative than using only the embedding of the previous word. Formally:

$$d_t^{avg} = \frac{1}{t-1} \sum_{i=1}^{t-1} y_i \quad (9)$$

4.2 Self-Attentive Residual Connections

Averaging is a simple and cheap way to aggregate information from multiple words, but may not be sufficient for all kinds of dependencies. Instead, we propose a dynamic way to aggregate information in each sentence, such that different words have different importance according to their relation with the prediction of the next word. We propose to use a shared self-attention mechanism to obtain a summary representation of the translation, i.e. a *weighted average representation* of the words translated from y_0 to y_{t-1}. This mechanism aims to model, in part, important non-sequential dependencies among words, and serves as a complementary memory to the recurrent layer.

$$d_t^{cavg} = \sum_{i=1}^{t-1} \alpha_i^t y_i \quad (10)$$

$$\alpha_i^t = softmax(e_i^t) \quad (11)$$

The weights of the attention model are computed by a scoring function e_i^t that predicts how important each previous word (y_0, ..., or y_{t-1}) is for the current prediction y_t.

We experiment with two different scoring functions, as follows:

$$e_i^t = v^\intercal tanh(W_y y_i + W_s s_t) \ \textit{(content+scope)} \quad (12)$$

$$\text{or } e_i^t = v^\intercal tanh(W_y y_i) \ \textit{(content)} \quad (13)$$

where $v \in \mathbb{R}^e$, $W_y \in \mathbb{R}^{e \times e}$, and $W_s \in \mathbb{R}^{e \times d}$ are weight matrices, e and d are the dimensions of the embeddings and hidden states respectively. Firstly, we study the scoring function noted *content+scope*, as proposed by Bahdanau et al. (2015) for NMT. Secondly, we explore a scoring function noted as *content*, which is calculated based only on the previous hidden states of the decoder, as proposed by Pappas and Popescu-Belis (2017). In contrast to the first attention function, which makes use of the hidden vector s_t, the second one is based only on the previous word representations, therefore, it is independent of the current prediction representation. However, the normalization of this function still depends on t.

5 Other Self-Attentive Networks

To compare our approach with similar studies, we adapted two representative self-attentive networks for application to NMT.

5.1 Memory RNN

The *Memory RNN* decoder is based on the proposal by Cheng et al. (2016) to modify an LSTM layer to include a memory with different cells for each previous output representation. Thus at each time step, the hidden layer can select past information dynamically from the memory. To adapt it to our framework, we modify Eq. (4) as:

$$s_t = f(\tilde{s}_t, y_{t-1}, c_t) \quad (14)$$

$$\text{where} \quad \tilde{s}_t = \sum_{i=1}^{t-1} \alpha_i^t s_i \quad (15)$$

$$\alpha_i^t = softmax(e_i^t) \quad (16)$$

$$e_i^t = a(h_i, y_{t-1}, \tilde{s}_{t-1}) \quad (17)$$

5.2 Self-Attentive RNN

The *Self-Attentive RNN* is the simplest one proposed by Daniluk et al. (2016), and incorporates a summary vector from past predictions calculated with an attention mechanism. Here, the attention is applied over previous hidden states. This decoder is formulated as follows:

$$p(y_t|y_1, ..., y_{t-1}, c_t) \approx g(s_t, y_{t-1}, c_t, \tilde{s}_t) \quad (18)$$

$$\text{where} \quad \tilde{s}_t = \sum_{i=1}^{t-1} \alpha_i^t s_i \tag{19}$$

$$\alpha_i^t = softmax(e_i^t) \tag{20}$$

$$e_i^t = a(s_i, s_t) \tag{21}$$

Additional details of the formulations in Sections 3, 4, and 5 are described in the Appendix A.

6 Experimental Settings

6.1 Datasets

To evaluate the proposed MT models in different conditions, we select three language pairs with increasing amounts of training data: English-Chinese (0.5M sentence pairs), Spanish-English (2.1M), and English-German (4.5M).

For English-to-Chinese, we use a subset of the UN parallel corpus (Rafalovitch and Dale, 2009)[1], with 0.5M sentence pairs for training, 2K for development, and 2K for testing. For training Spanish-to-English MT, we use a subset of WMT 2013 (Bojar et al., 2013), corresponding to Europarl v7 and News Commentary v11 with ca. 2.1M sentence pairs. Newstest2012 and Newstest2013 were used for development and testing respectively. Finally, we use the complete English-to-German set from WMT 2016 (Bojar et al., 2016) with a total of ca. 4.5M sentence pairs. The development set is Newstest2013, and the testing set is Newstest2014. Additionally, we include as testing sets Newstest2015 and Newstest2016, for comparison with the state of the art. We report translation quality using (a) BLEU over *tokenized* and *truecased* texts, and (b) NIST BLEU over *detokenized* and *detruecased* texts[2].

6.2 Model Configuration

We use the implementation of the attention-based NMT baseline provided in `dl4mt-tutorial`[3] developed in Python using Theano (Theano Development Team, 2016). The system implements an attention-based NMT model, described above, using one layer of GRUs (Cho et al., 2014). The vocabulary size is 25K for English-to-Chinese NMT, and 50K for Spanish-to-English and English-German. We use the byte pair encoding (BPE) strategy for out-of-vocabulary words (Sennrich et al., 2016b). For all cases, the maximum sentence length of the training samples is 50, the dimension of the word embeddings is 500, and the dimension of the hidden layers is 1,024. We use dropout with a probability of 0.5 after each layer. The parameters of the models are initialized randomly from a standard normal distribution scaled to a factor of 0.01. The loss function is optimized using Adadelta (Zeiler, 2012) with $\epsilon = 10^{-6}$ and $\rho = 0.95$ as in the original paper. The systems were trained in 7–12 days for each model on a Tesla K40 GPU at the speed of about 1,000 words/sec.

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

[2] Scrips from Moses toolkit (Koehn et al., 2007): BLEU *multi-bleu*, NIST BLEU *mteval-v13a.pl*, *tokenizer.perl*, *truecase.perl*.

[3] https://github.com/nyu-dl/dl4mt-tutorial

Models	|Θ|	BLEU En–Zh	BLEU Es–En
SMT baseline	–	21.6	25.2
NMT baseline	108.7M	22.6	25.4
+ Memory RNN	109.7M	22.5	25.5
+ Self-attentive RNN	110.2M	22.0	25.1
+ Mean residual connections	108.7M	23.6	25.7
+ Self-attentive residual connections	108.9M	**24.0**	**26.3**

Table 1: BLEU score (multi-bleu) on *tokenized* text. The highest score per dataset is marked in bold. The self-attentive residual connections make use of the *content* attention function. $|\Theta|$ indicates the number of parameters per model.

7 Analysis of the Results

Table 1 shows the BLEU scores and the number of parameters used by the different NMT models. Along with the NMT baseline, we included a statistical machine translation (SMT) model based on Moses (Koehn et al., 2007) with the same training/tuning/test data as the NMT. The performance of *memory RNN* is similar to the baseline and, as confirmed later, its focus of attention is mainly on the prediction at $t-1$. The *self-attentive RNN* method is inferior to the baseline, which can be attributed to the overhead on the hidden vectors that have to learn the recurrent representations and the attention simultaneously. The proposed models outperform the baseline, and the best scores are obtained by the NMT model with *self-attentive residual connections*. Despite their simplicity, the *mean residual connections* already improve the translation, without increasing the number of parameters.

Tables 2 and 3 show further experiments with the proposed methods on various English-German test sets, compared to several previous systems.

Models	BLEU NT14	NT15
NMT (unk. word repl.) (Luong et al., 2015)	20.9	–
Context-aware NMT (Zhang et al., 2017)	22.57	–
Recurrent attention NMT (Yang et al., 2017)	22.1	25.0
Variational NMT (Zhang et al., 2016a)	–	25.49
NMT baseline	22.3	24.8
+ Memory RNN	22.6	24.9
+ Self-attentive RNN	22.0	24.3
+ Mean residual connections	22.9	24.9
+ Self-attentive residual connections	**23.2**	**25.5**

Table 2: BLEU score (multi-bleu) on *tokenized* text for English-to-German on *Newstest (NT) 2014, and 2015*. The highest score per dataset is marked in bold. The self-attentive residual connections makes use of the *content* attention function.

Models	BLEU (NIST) NT14	NT15	NT16
Winning WMT	20.1	24.4	**34.2**
NMT (BPE) (Sennrich et al., 2016b)	–	22.8	–
Syntax NMT (Nadejde et al., 2017)	–	–	29.0
NMT Baseline	21.0	24.4	28.8
+ Mean residual connections*	21.4	24.7	29.6
+ Self-attentive residual connections**	**21.7**	**25.0**	29.7

Table 3: NIST BLEU scores on *detokenized* and *detruecased* text for English-to-German on *Newstest (NT) 2014, 2015, 2016*. Significance test: * $p < 0.05$, ** $p < 0.01$. The Winning WMT systems are listed in the text below.

Table 2 shows BLEU values calculated by *multi-bleu*, and includes the NMT system proposed by Luong et al. (2015) which replaces unknown predicted words with the most strongly aligned word on the source sentence. Also, the table includes other systems described in Section 2. Additionally, Table 3 shows values calculated by the NIST BLEU scorer, as well as results reported by the "Winning WMT" systems for each test set respectively: UEDIN-SYNTAX (Williams et al., 2014), UEDIN-SYNTAX (Williams et al., 2015), and UEDIN-NMT (Sennrich et al., 2016a). Also, we include the results reported by Sennrich et al. (2016b) for a baseline encoder-decoder NMT with BPE for unknown words similar to our configuration, and finally the system proposed by Nadejde et al. (2017), an explicit syntax-aware NMT that introduces combinatory categorial grammar (CCG) supertags on the target side by predicting words and tags alternately. The comparison with this work is relevant for the analysis described later in Section 8.2. The results confirm that the *self-attentive residual connections* improve significantly the translations. To evaluate the significance of the improvements against the NMT baseline, we performed a one-tailed paired t-test.

Attention function	BLEU En-Zh	Es-En
Content+Scope	23.1	25.6
Content	**24.0**	**26.3**

Table 4: BLEU scores for two scoring variants of the attention function of the proposed decoder.

7.1 Impact of the Attention Function

We now examine the two scoring functions that can be used for the *self-attentive residual connections* model presented in Eq. (12), considering English-to-Chinese and Spanish-to-English. The BLEU scores are presented in Table 4: the best option is the *content* matching function, which depends only on the word embeddings. The *content+scope* function, which depends additionally on the hidden representation of the current prediction is better than the baseline but scores lower than *content*. The idea that the importance of the context depends on the current prediction is appealing, because it can be interpreted as learning internal dependencies among words. However, the experimental results show that it does not necessarily lead to the best translation. On the contrary, the *content* attention function may be extracting representations of the whole sentence which are easier to learn and generalize.

7.2 Performance According to Human Evaluation

Manual evaluation on samples of 50 sentences for each language pair helped to corroborate the conclusions obtained from the BLEU scores, and to provide a qualitative understanding of the improvements brought by our model. For each language, we employed one evaluator who was a native speaker of the target language and had good knowledge of the source language. The evaluators ranked three translations of the same source sentence – one from each of our models: *baseline*, *mean residual connections*, and *self-attentive residual connections* – according to their translation quality. The three translations were presented in a random order, so that the system that had generated them could not be identified. To integrate

	Ranking (%)								
System	**En–Zh**			**Es–En**			**En–De**		
	>	=	<	>	=	<	>	=	<
Mean vs. Baseline	26	56	18	20	64	16	28	58	24
Self-attentive vs. Baseline	28	60	12	28	56	16	32	54	14
Self-attentive vs. Mean	24	62	14	28	58	14	32	56	12

Table 5: Human evaluation of sentence-level translation quality on three language pairs. We compare the models in pairs, indicating the percentages of sentences that were ranked higher (>), equal to (=), or lower (<) for the first system with respect to the second one. The values correspond to percentages (%).

Systems	d	**Perplexity**
LSTM (Daniluk et al., 2016)	300	85.2
LSTM + Attention (Daniluk et al., 2016)	296	82.0
LSTM + 4-gram (Daniluk et al., 2016)	968	75.9
LSTM + Mean residual connections	296	80.2
LSTM + Self-attentive residual connections	296	80.4

Table 6: Evaluation of the proposed methods on language modeling. The number of parameter for all models is 47M.

the judgments, we proceed in pairs, and count the number of times each system was ranked higher, equal to, or lower than another competing system. The results shown in Table 5 indicate that the *self-attentive residual connections* model outperforms the one with *mean residual connections*, and both outperform the baseline, for all three language pairs. The rankings are thus identical to those obtained using BLEU in Tables 1 and 3.

7.3 Performance on Language Modeling

To examine whether language modeling (LM) can benefit from the proposed method, we incorporate the residual connections into a neural LM. We use the same setting as Daniluk et al. (2016) for a corpus of Wikipedia articles (22.5M words), and we compare with two methods proposed in the same paper, namely attention LSTM and 4-gram LSTM. As shown in Table 6, the proposed models outperform the LSTM baseline as well as the self-attention model, but not the 4-gram LSTM. Experiments using 4-gram LSTM for NMT showed poor performance (13.9 BLEU points for English-Chinese) which can be attributed to the difference between the LM and NMT tasks. Both tasks predict one word at a time conditioned over previous words, however, in NMT the previous target-word-inputs are not given, they have to be generated by the decoder. Thus, the output could be conditioned over previous erroneous predictions

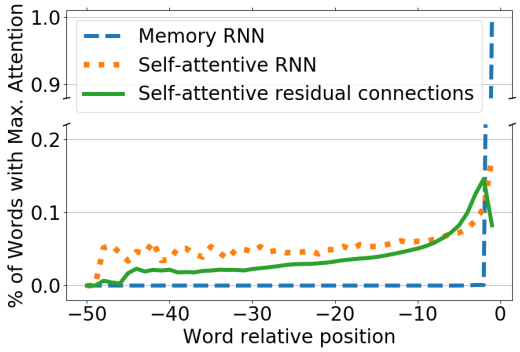


Figure 2: Percentage of words that received maximum attention at a given relative position, ranging from -1 to -50 (maximum length).

affecting in higher proportion the 4-gram LSTM model. This shows that even if a model improves language modeling, it does not necessarily improve machine translation.

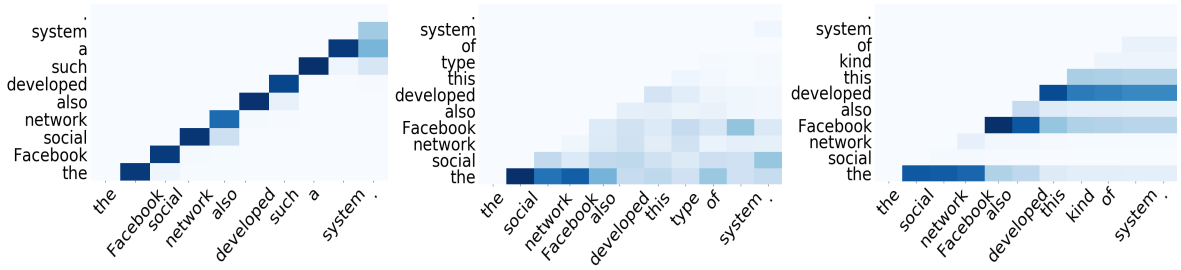
8 Qualitative Analysis

8.1 Distribution of Attention

Figure 2 shows a comparison of the distribution of attention of the different self-attentive models described in this paper, on Spanish-to-English NMT (the other two language pairs exhibit similar distributions). The values correspond to the number of words which received maximal attention for each relative position (x-axis). We selected, at each prediction, the preceding word with maximal weight, and counted its relative position. We normalized the count by the number of previous words at the time of each prediction.

We observe that the *memory RNN* almost always selects the immediately previous word ($t-1$) and ignores the rest of the context. On the contrary, the other two models distribute attention more evenly among all previous words. In particular, the *self-attentive RNN* uses a longer context than the *self-attentive residual connections* but, as the performance on BLEU score shows, this fact does not necessarily mean better translation.

Figure 3 shows the attention to previous words generated by each model for one sentence translated from Spanish to English. The matrices present the target-side attention weights, with the vertical axis indicating the previous words, and the color shades at each position (cell) representing the attention weights. The weights of the *memory RNN* are concentrated on the diagonal, indicating that the attention is generally located on



(a) Memory RNN (b) Self-attentive RNN (c) Self-attentive residual connections

Figure 3: Matrix of distribution of the attention weights to previous words. The vertical axis represents the previous words. A darker shade indicates a higher attention weight.

Algorithm 1 Binary Parse Tree

Require: **A** matrix of attention of size $N \times N$
Require: **s** sentence as list of words of size N
function SPLIT($tree$, **A**, **s**)
  $n \leftarrow length(s)$
  $i \leftarrow 0$
  while $max(\mathbf{A}[:][i]) = 0$ or $i < n$ **do**
    $i \leftarrow i + 1$
  end while
  $tree.addChild(\mathbf{s}[0:i])$
  if $i < n$ **then**
    $subtree \leftarrow newTree()$
    SPLIT($subtree$, $\mathbf{A}[i:n][i:n]$, $\mathbf{s}[i:n]$))
    $tree.addChild(subtree)$
  end if
end function
$tree \leftarrow newTree()$; SPLIT($tree$, **A**, **s**)

the previous word, which makes the model almost equivalent to the baseline. The weights of the *self-attentive RNN* show that attention is more distributed towards the distant past, and they vary for each word because the attention function depends on the current prediction. This model tries to find dependencies among words, although complex relations seem difficult to learn. On the contrary, the proposed *self-attentive residual connections* model strongly focuses on particular words, and we present a wider analysis of it in the following section.

8.2 Structures Learned by the Model

When visualizing the matrix of attention weights generated by our model (Figure 3c), we observed the formation of sub-phrases which are grouped depending on their attention to previous words. To build the sub-phrases in a deterministic fashion, we implemented Algorithm 1, which iteratively splits the sentence into two sub-phrases every time the focus of attention changes to a new word, from left-to-right. The results are binary tree structures containing the sub-phrases, exemplified in Figure 4.

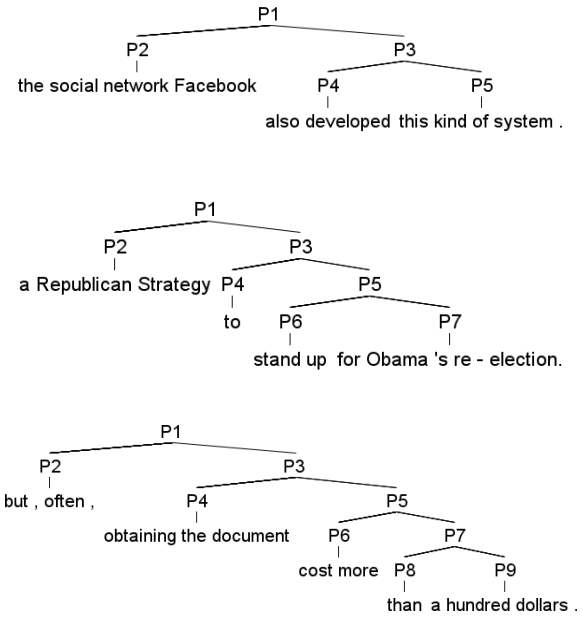


Figure 4: Examples of hypothesized syntactic structures obtained with Algorithm 1.

We formally evaluate the syntactic properties of the binary tree structures by comparing them with the results of an automatic constituent parser (Manning et al., 2014), using the ParsEval approach (Black et al., 1991), i.e. by counting the precision and recall of constituents, excluding single words. Our models reaches a precision of 0.56, which is better than the precision of 0.45 obtained by a trivial right-branched tree model[4]. Note that these structures were neither optimized for parsing nor learned using part-of-speech tagging as most parsers do. Our interpretation of the results is that they are "syntactic-like" structures. However, given the simplicity of the model, they could

[4]A model constructed by dividing iteratively one word and the rest of the sentence, from left-to-right.

Better than baseline	
S:	Estudiantes y profesores se están tomando a la ligera la fecha.
R:	Students and teachers are taking the date lightly.
B:	Students and teachers are *being taken lightly to the date*.
O:	Students and teachers are **taking the date lightly**.
S:	No porque compartiera su ideología, sino porque para él los Derechos Humanos son indivisibles.
R:	Not because he shared their world view, but because for him, human rights are indivisible.
B:	Not because *I* share his ideology, but because *he is indivisible by human rights*.
O:	Not because **he** shared his ideology, but because **for him human rights are indivisible**.
Worse than baseline	
S:	El gobierno intenta que no se construyan tantas casas pequeñas.
R:	The Government is trying not to build so many small houses.
B:	The government is trying *not to build so many small houses*.
O:	The government is trying **to ensure that so many small houses are not built**.
S:	Otras personas pueden tener niños .
R:	Other people can have children.
B:	*Other people can* have children.
O:	**Others may** have children.

Table 7: Examples from Spanish to English.

also be viewed as more limited structures, similar to sentence chunks.

8.3 Translation Examples

Table 7 shows examples of translations produced with the baseline and the *self-attentive residual connections* model. The first part shows examples for which the proposed model reached a higher BLEU score than the baseline. Here, the structure of the sentences, or at least the word order, are improved. The second part contains examples where the baseline achieved better BLEU score than our model. In the first example, the structure of the sentence is different but the content and quality are similar, while in the second one lexical choices differ from the reference.

9 Conclusion

We presented a novel decoder which uses self-attentive residual connections to previously translated words in order to enrich the target-side contextual information in NMT. To cope with the variable lengths of previous predictions, we proposed two methods for context summarization: *mean residual connections* and *self-attentive residual connections*. Additionally, we showed how similar previous proposals, designed for language modeling, can be adapted to NMT. We evaluated the methods over three language pairs: Chinese-to-English, Spanish-to-English, and English-to-German. In each case, we improved the BLEU score compared to the NMT baseline and two variants with memory-augmented decoders. A manual evaluation over a small set of sentences for each language pair confirmed the improvement. Finally, a qualitative analysis showed that the proposed model distributes weights throughout an entire sentence, and learns structures resembling syntactic ones.

As future work, we plan to enrich the present attention mechanism with the *key-value-prediction* technique (Daniluk et al., 2016; Miller et al., 2016) which was shown to be useful for language modeling. Moreover, we will incorporate relative positional information to the attention function. To encourage further research in *self-attentive residual connections* for NMT an other similar tasks, our code is made publicly available[5].

Acknowledgments

We are grateful for support to the European Union under the Horizon 2020 SUMMA project (grant n. 688139, see www.summa-project.eu). We would also like to thank James Henderson for his valuable feedback and suggestions.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*. San Diego, USA.

Ezra W. Black, Steven Abney, Daniel P. Flickenger, Claudia Gdaniec, Ralph Grishman, Philip Harrison, Donald Hindle, Robert J. P. Ingria, Frederick Jelinek, Judith L. Klavans, Mark Y. Liberman, Mitchell P. Marcus, Salim Roukos, Beatrice Santorini, and Tomek Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove*. California, USA. http://www.aclweb.org/anthology/H91-1060.

[5]This work is part of the project *Towards Document-Level Neural Machine Translation* (Miculicich Werlen, 2017). The code is available at https://github.com/idiap/Attentive_Residual_Connections_NMT

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Sofia, Bulgaria, pages 1–44. http://www.aclweb.org/anthology/W13-2201.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 Conference on Machine Translation. In *Proceedings of the First Conference on Machine Translation*. Association for Computational Linguistics, Berlin, Germany, pages 131–198. http://www.aclweb.org/anthology/W16-2301.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 551–561. https://aclweb.org/anthology/D16-1053.

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

Michał Daniluk, Tim Rocktäschel, Johannes Welbl, and Sebastian Riedel. 2016. Frustratingly short attention spans in neural language modeling. In *Proceedings of the International Conference on Learning Representations*. San Juan, Puerto Rico.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*. PMLR, International Convention Centre, Sydney, Australia, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252. http://proceedings.mlr.press/v70/gehring17a.html.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. volume 00, pages 770–778. https://doi.org/10.1109/CVPR.2016.90.

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

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 1700–1709. http://www.aclweb.org/anthology/D13-1176.

Jaeyoung Kim, Mostafa El-Khamy, and Jungwon Lee. 2017. Residual LSTM: design of a deep recurrent architecture for distant speech recognition. *CoRR* abs/1701.03360. http://arxiv.org/abs/1701.03360.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*. Association for Computational Linguistics, Prague, Czech Republic, pages 177–180. http://www.aclweb.org/anthology/P07-2045.

Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2017. Recurrent additive networks. *CoRR* abs/1705.07393. http://arxiv.org/abs/1705.07393.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *Proceedings of the International Conference on Learning Representations*. Toulon, France.

Yang Liu and Mirella Lapata. 2018. Learning structured text representations. *Transactions of the Association for Computational Linguistics* 6:63–75. https://transacl.org/ojs/index.php/tacl/article/view/1185/280.

Thang Luong, Hieu Pham, and D. Christopher Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1412–1421. https://doi.org/10.18653/v1/D15-1166.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, Baltimore, Maryland, pages 55–60. http://www.aclweb.org/anthology/P14-5010.

Lesly Miculicich Werlen. 2017. Towards document-level neural machine translation. Idiap-RR Idiap-RR-25-2017, Idiap.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1400–1409. https://aclweb.org/anthology/D16-1147.

Maria Nadejde, Siva Reddy, Rico Sennrich, Tomasz Dwojak, Marcin Junczys-Dowmunt, Philipp Koehn, and Alexandra Birch. 2017. Predicting target language CCG supertags improves neural machine translation. In *Proceedings of the Second Conference on Machine Translation*. Association for Computational Linguistics, Copenhagen, Denmark, pages 68–79. http://www.aclweb.org/anthology/W17-4707.

Nikolaos Pappas and Andrei Popescu-Belis. 2017. Explicit document modeling through weighted multiple-instance learning. *Journal of Artificial Intelligence Research* 58:591–626.

Alexandre Rafalovitch and Robert Dale. 2009. United Nations General Assembly resolutions: A six-language parallel corpus. In *Proceedings of the MT Summit*. volume 12, pages 292–299.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Edinburgh Neural Machine Translation Systems for WMT 16. In *Proceedings of the First Conference on Machine Translation*. Association for Computational Linguistics, Berlin, Germany, pages 371–376. http://www.aclweb.org/anthology/W/W16/W16-2323.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the ACL (Vol. 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1715–1725. http://www.aclweb.org/anthology/P16-1162.

Rohollah Soltani and Hui Jiang. 2016. Higher order recurrent neural networks. *CoRR* abs/1605.00064. http://arxiv.org/abs/1605.00064.

Sainbayar Sukhbaatar, arthur szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, Curran Associates, Inc., pages 2440–2448. http://papers.nips.cc/paper/5846-end-to-end-memory-networks.pdf.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, Curran Associates, Inc., pages 3104–3112.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* abs/1605.02688. http://arxiv.org/abs/1605.02688.

Ke Tran, Arianna Bisazza, and Christof Monz. 2016. Recurrent memory networks for language modeling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 321–331. http://www.aclweb.org/anthology/N16-1036.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, Curran Associates, Inc., pages 5998–6008. http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Cheng Wang. 2017. RRA: recurrent residual attention for sequence learning. *CoRR* abs/1709.03714. http://arxiv.org/abs/1709.03714.

Mingxuan Wang, Zhengdong Lu, Hang Li, and Qun Liu. 2016. Memory-enhanced decoder for neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 278–286. https://aclweb.org/anthology/D16-1027.

Yiren Wang and Fei Tian. 2016. Recurrent residual learning for sequence classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 938–943. https://aclweb.org/anthology/D16-1093.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *Proceedings of the International Conference on Learning Representations*. San Diego, USA.

Philip Williams, Rico Sennrich, Maria Nadejde, Matthias Huck, Eva Hasler, and Philipp Koehn. 2014. Edinburgh's Syntax-Based Systems at WMT 2014. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Baltimore, Maryland, USA,

pages 207–214. http://www.aclweb.org/anthology/W/W14/W14-3324.

Philip Williams, Rico Sennrich, Maria Nadejde, Matthias Huck, and Philipp Koehn. 2015. Edinburgh's Syntax-Based Systems at WMT 2015. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Lisbon, Portugal, pages 199–209. http://aclweb.org/anthology/W15-3024.pdf.

Zichao Yang, Zhiting Hu, Yuntian Deng, Chris Dyer, and Alex Smola. 2017. Neural machine translation with recurrent attention modeling. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, Valencia, Spain, pages 383–387. http://www.aclweb.org/anthology/E17-2061.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR* abs/1212.5701. http://arxiv.org/abs/1212.5701.

Biao Zhang, Deyi Xiong, Jinsong Su, and Hong Duan. 2017. A context-aware recurrent encoder for neural machine translation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 25(12):2424–2432.

Biao Zhang, Deyi Xiong, jinsong su, Hong Duan, and Min Zhang. 2016a. Variational neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 521–530. https://aclweb.org/anthology/D16-1050.

Yu Zhang, Guoguo Chen, Dong Yu, Kaisheng Yaco, Sanjeev Khudanpur, and James Glass. 2016b. Highway long short-term memory RNNs for distant speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. pages 5755–5759. https://doi.org/10.1109/ICASSP.2016.7472780.

Julian Georg Zilly, Rupesh Kumar Srivastava, Jan Koutník, and Jürgen Schmidhuber. 2017. Recurrent highway networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*. PMLR, International Convention Centre, Sydney, Australia, volume 70 of *Proceedings of Machine Learning Research*, pages 4189–4198. http://proceedings.mlr.press/v70/zilly17a.html.

A Detailed Architecture

This appendix describes in detail the implementation of the *self-attentive residual decoder* for NMT, which builds on the attention-based NMT implementation of `dl4mt-tutorial`[6].

The input of the model is a source sentence denoted as 1-of-k coded vector, where each element of the sequence corresponds to a word:

$$x = (x_1, x_2, ..., x_m), x_i \in \mathbb{R}^V$$

and the output is a target sentence denoted as well as 1-of-k coded vector:

$$y = (y_1, y_2, ..., y_n), y_i \in \mathbb{R}^V$$

where V is the size of the vocabulary of target and source side, m and n are the lengths of the source and target sentences respectively. We omit the bias vectors for simplicity.

A.1 Encoder

Each word of the source sentence is embedded in a e-dimensional vector space using the embedding matrix $\bar{E} \in \mathbb{R}^{e \times V}$. The hidden states are $2d$-dimensional vectors modeled by a bi-directional GRU. The forward states $\overrightarrow{h} = (\overrightarrow{h}_1, ..., \overrightarrow{h}_m)$ are computed as:

$$\overrightarrow{h}_i = \overrightarrow{z}_i \odot \overrightarrow{h}_{i-1} + (1 - \overrightarrow{z}_i) \odot \overrightarrow{h}'_i$$

where

$$\overrightarrow{h}'_i = tanh(\overrightarrow{W}\bar{E}x_i + \overrightarrow{U}[\overrightarrow{r}_i \odot \overrightarrow{h}_{i-1}])$$
$$\overrightarrow{z}_i = \sigma(\overrightarrow{W}_z\bar{E}x_i + \overrightarrow{U}_z\overrightarrow{h}_{i-1})$$
$$\overrightarrow{r}_i = \sigma(\overrightarrow{W}_r\bar{E}x_i + \overrightarrow{U}_r\overrightarrow{h}_{i-1})$$

Here, $\overrightarrow{W}, \overrightarrow{W}_z, \overrightarrow{W}_r \in \mathbb{R}^{d \times e}$ and $\overrightarrow{U}, \overrightarrow{U}_z, \overrightarrow{U}_r \in \mathbb{R}^{d \times d}$ are weight matrices. The backward states $\overleftarrow{h} = (\overleftarrow{h}_1, ..., \overleftarrow{h}_m)$ are computed in similar manner. The embedding matrix $\bar{E}$ is shared for both passes, and the final hidden states are formed by the concatenation of them:

$$h_i = \begin{bmatrix} \overrightarrow{h}_i \\ \overleftarrow{h}_i \end{bmatrix}$$

A.2 Attention Mechanism

The *context vector* at time t is calculated by:

$$c_t = \sum_{i=1}^{m} \alpha_i^t h_i$$

[6] https://github.com/nyu-dl/dl4mt-tutorial

where

$$\alpha_i^t = \frac{exp(e_i^t)}{\sum_j exp(e_j^t)}$$
$$e_i^t = v_a^\intercal tanh(W_d s_{t-1} + W_e h_i)$$

Here, $v_a \in \mathbb{R}^d$, $W_d \in \mathbb{R}^{d \times d}$ and $W_e \in \mathbb{R}^{d \times 2d}$ are weight matrices.

A.3 Decoder

The input of the decoder are the previous word y_{t-1} and the *context vector* c_t, the objective is to predict y_t. The hidden states of the decoder $s = (s_1, ..., s_n)$ are initialized with the mean of the *context vectors*:

$$s_0 = tanh(W_{init} \frac{1}{m} \sum_{i=1}^{m} c_i)$$

where $W_{init} \in \mathbb{R}^{d \times 2d}$ is a weight matrix, m is the size of the source sentence. The following hidden states are calculated with a GRU conditioned over the *context vector* at tine t as follows:

$$s_t = z_t \odot s'_t + (1 - z_t) \odot s''_t$$

where

$$s''_t = tanh(Ey_{t-1} + U[r_t \odot s_{t-1}] + Cc_t)$$
$$z_i = \sigma(W_z Ey_{t-1} + U_z s_{t-1} + C_z c_t)$$
$$r_i = \sigma(W_r Ey_{t-1} + U_r s_{t-1} + C_r c_t)$$

Here, $E \in \mathbb{R}^{e \times V}$ is the embedding matrix for the target language. $W, W_z, W_r \in \mathbb{R}^{d \times e}$, $U, U_z, U_r \in \mathbb{R}^{d \times d}$, and $C, C_z, C_r \in \mathbb{R}^{d \times 2d}$ are weight matrices. The intermediate vector s'_t is calculated from a simple GRU:

$$s'_t = GRU(y_{t-1}, s_{t-1})$$

In the attention-based NMT model, the probability of a target word y_t is given by:

$$p(y_t|s_t, y_{t-1}, c_t) = softmax(W_o tanh(W_{st} s_t + W_{yt} y_{t-1} + W_{ct} c_t))$$

Here, $W_o \in \mathbb{R}^{V \times e}$, $W_{st} \in \mathbb{R}^{e \times d}$, $W_{yt} \in \mathbb{R}^{e \times e}$, $W_{ct} \in \mathbb{R}^{e \times 2d}$ are weight matrices.

A.3.1 Self-Attentive Residual Connections

In our model, the probability of a target word y_t is given by:

$$p(y_t|s_t, d_t, c_t) = softmax(W_o tanh(W_{st} s_t + W_{dt} d_t + W_{ct} c_t))$$

Here, $W_o \in \mathbb{R}^{V \times e}$, $W_{st} \in \mathbb{R}^{e \times d}$, $W_{dt}, W_{yt} \in \mathbb{R}^{e \times e}$, $W_{ct} \in \mathbb{R}^{e \times 2d}$ are weight matrices. The summary vector d_t can be calculated in different manners based on previous words y_1 to y_{t-1}. First, a simple average:

$$d_t^{avg} = \frac{1}{t-1} \sum_{i=1}^{t-1} y_i$$

The second, by using an attention mechanism:

$$d_t^{cavg} = \sum_{i=1}^{t-1} \alpha_i^t y_i$$
$$\alpha_i^t = \frac{exp(e_i^t)}{\sum_{j=1}^{t-1} exp(e_j^t)}$$
$$e_i^t = v^\intercal tanh(W_y y_i)$$

where $v \in \mathbb{R}^e$, $W_y \in \mathbb{R}^{e \times e}$ are weight matrices.

A.3.2 Memory RNN

This model modifies the recurrent layer of the decoder as follows:

$$s_t = z_t \odot s'_t + (1 - z_t) \odot s''_t$$

where

$$s''_t = tanh(Ey_{t-1} + U[r_t \odot \tilde{s}_t] + Cc_t)$$
$$z_i = \sigma(W_z Ey_{t-1} + U_z \tilde{s}_t + C_z c_t)$$
$$r_i = \sigma(W_r Ey_{t-1} + U_r \tilde{s}_t + C_r c_t)$$

Here, $E \in \mathbb{R}^{e \times V}$ is the embedding matrix for the target language. $W, W_z, W_r \in \mathbb{R}^{d \times e}$, $U, U_z, U_r \in \mathbb{R}^{d \times d}$, and $C, C_z, C_r \in \mathbb{R}^{d \times 2d}$ are weight matrices. The intermediate vector s'_t is calculated from a simple GRU:

$$s'_t = GRU(y_{t-1}, \tilde{s}_t)$$

The recurrent vector $\tilde{s}_t$ is calculated as following:

$$\tilde{s}_t = \sum_{i=1}^{t-1} \alpha_i^t s_i$$

where

$$\alpha_i^t = \frac{exp(e_i^t)}{\sum_{j=1}^{t-1} exp(e_j^t)}$$
$$e_i^t = v^\intercal tanh(W_m s_i + W_s s_t)$$

where $v \in \mathbb{R}^d$, $W_m \in \mathbb{R}^{d \times d}$, and $W_s \in \mathbb{R}^{d \times d}$ are weight matrices.

A.3.3 Self-Attentive RNN

The formulation of this decoder is as following:

$$p(y_t|y_1, ..., y_{t-1}, c_t) \approx softmax(W_o tanh(\\ W_{st}s_t + W_{yt}y_{t-1} + W_{ct}c_t + W_{mt}\tilde{s}_t))$$

Here, $W_o \in \mathbb{R}^{V \times e}$, $W_{st} \in \mathbb{R}^{e \times d}$, $W_{yt} \in \mathbb{R}^{e \times e}$, $W_{ct} \in \mathbb{R}^{e \times 2d}$, and $W_{mt} \in \mathbb{R}^{e \times d}$ are weight matrices.

$$\tilde{s}_t = \sum_{i=1}^{t-1} \alpha_i^t s_i$$
$$\alpha_i^t = \frac{exp(e_i^t)}{\sum_{j=1}^{t-1} exp(e_j^t)}$$
$$e_i^t = v^\intercal tanh(W_m s_i + W_s s_t)$$

where $v \in \mathbb{R}^d$, $W_m \in \mathbb{R}^{d \times d}$, and $W_s \in \mathbb{R}^{d \times d}$ are weight matrices.

Target Foresight based Attention for Neural Machine Translation*

Xintong Li†, Lemao Liu‡, Zhaopeng Tu‡, Shuming Shi‡, Max Meng†

†The Chinese University of Hong Kong
{xtli, qhmeng}@ee.cuhk.edu.hk
‡Tencent AI Lab
{redmondliu, zptu, shumingshi}@tencent.com

Abstract

in neural machine translation, an attention model is used to identify the aligned source words for a target word (target foresight word) in order to select translation context, but it does not make use of any information of this target foresight word at all. previous work proposed an approach to improve the attention model by explicitly accessing this target foresight word and demonstrated the substantial gains in alignment task. however, this approach is useless in machine translation task on which the target foresight word is unavailable. in this paper, we propose a new attention model enhanced by the implicit information of target foresight word oriented to both alignment and translation tasks. empirical experiments on chinese-to-english and japanese-to-english datasets show that the proposed attention model delivers significant improvements in terms of both alignment error rate and bleu.

1 Introduction

Since neural machine translation (NMT) was proposed (Bahdanau et al., 2014), it has been attracted increasing interests in machine translation community (Luong et al., 2015b; Tu et al., 2016; Feng et al., 2016; Cohn et al., 2016). NMT not only yields impressive translation performance in practice, but also has appealing model architecture in essence. Compared with traditional statistical machine translation (Koehn et al., 2003; Chiang, 2005), one of advantages in NMT is that its architecture combines language model, translation model and alignment between source and target words in a unified manner rather than a pipeline manner, and it thereby has the potential to alleviate the issue of error propagation.

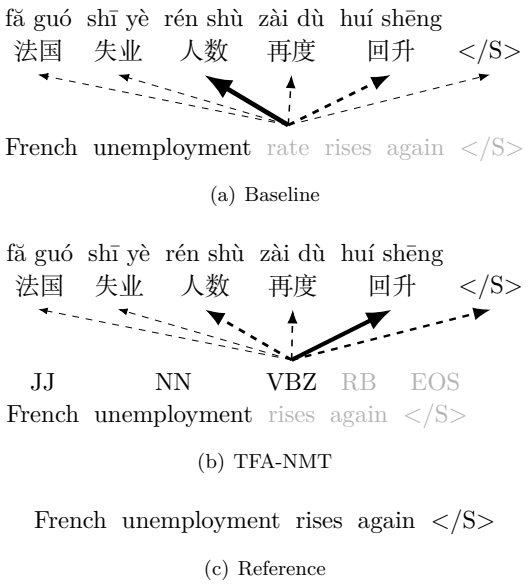


Figure 1: A running example to motivate the proposed model. (a) The baseline obtains a translation error due to the incorrect attention. (b) With the help of the target foresight information "VBZ", TFA-NMT is likely to figure out the exact translation as the reference in (c). The light font denotes the target words to be translated in future. Both dashed or solid arrowed lines denote the alignments and solid one denotes the 1-best alignment.

In NMT, the attention mechanism plays an important role. It calculates the alignments of a target word with respect to the source words for translation context selection. Although the source words are always available in inference, the target word, called target foresight word, [1]

*Work done when X. Li interning at Tencent AI Lab. L. Liu is the corresponding author.

[1]Note that the concept of foresight word in our translation task is not exactly the same as the original concept in alignment task (Peter et al., 2017). However, both of them share a common idea that foresight word should be at a later time step, and thus we respect the work in Peter et al. (2017) and maintain the same concept for easier understanding.

Proceedings of NAACL-HLT 2018, pages 1380–1390
New Orleans, Louisiana, June 1 - 6, 2018.

i.e. the first light color word in Figure 1(a), is not known but to be translated at the next time step. Therefore, this may lead to inadequate modeling for attention mechanism (Liu et al., 2016a; Peter et al., 2017). Regarding to this, Peter et al. (2017) explicitly feed this target word into the attention model, and demonstrate the significant improvements in alignment accuracy. Unfortunately, this approach relies on the premise that the target foresight word is available in advance in its alignment scenario, and thus it can not be used in the translation scenario.

To address this issue, in this paper, we propose a target foresight based attention (TFA) model oriented to both alignment and translation tasks. Its basic idea includes two steps: it firstly designs an auxiliary mechanism to predict some information for the target foresight word which is helpful for alignment; and then it feeds the predicted result into the attention model for translation. For the sake of efficiency, instead of predicting the target foresight word with large vocabulary size, we only predict its partial information, i.e. part-of-speech tag, which is proved to be helpful for word alignment (Liu et al., 2005). Figure 1(b) shows the main idea of TFA based on NMT. In order to remit the negative effects due to the prediction errors, we feed the distribution of the prediction result instead of the maximum a posteriori result into the attention model. In addition, since the target foresight words are available during the training, we jointly learn the prediction model for the target foresight words and the translation model in a supervised manner.

This paper makes the following contributions:

- It proposes a novel TFA-NMT for neural machine translation by using an auxiliary mechanism to predict the target foresight word which is subsequently used to enhance the attention model.
- It empirically shows that the proposed TFA-NMT can lead to better alignment accuracy, and achieves significant improvements on both Chinese-to-English and Japanese-to-English translation tasks.

2 Background

Given a source sentence $\mathbf{x} = \{x_1, \ldots, x_m\}$ with length m and a target sentence $\mathbf{y} = \{y_1, \ldots, y_n\}$ with length n, neural machine translation aims to model the conditional probability $P(\mathbf{y} \mid \mathbf{x})$:

$$P(\mathbf{y} \mid \mathbf{x}) = \prod_{i=1}^{n} P(y_i \mid \mathbf{y}_{<i}, \mathbf{x}), \quad (1)$$

where $\mathbf{y}_{<i} = \{y_1, \ldots, y_{i-1}\}$ denotes a prefix of $\mathbf{y}$ with length $i-1$.

To achieve this, neural machine translation adopts recurrent neural network (RNN) under the encoder-decoder framework (Bahdanau et al., 2014). In encoding, an encoder reads the source sentence $\mathbf{x}$ into a sequence of representation vectors by a bidirectional recurrent neural network. Suppose h_i denotes the representation vector for x_i, and let $\mathbf{h} = \{h_1, \ldots, h_m\}$. In decoding, a decoder sequentially generates a target word according to $P(y_i \mid \mathbf{y}_{<i}, \mathbf{x})$ by using another RNN.

In Eq.(1), the distribution $P(y_i \mid \mathbf{y}_{<i}, \mathbf{x})$ is used to generate y_i as follows:

$$P(y_i \mid \mathbf{y}_{<i}, \mathbf{x}) = \mathrm{softmax}(\phi(y_{i-1}, s_i, c_i)), \quad (2)$$

where ϕ represents a feedforward neural network, c_i is the context vector from $\mathbf{h}$ to infer y_i, and s_i denotes the hidden state at timestamp i via the decoding RNN represented by f:

$$s_i = f(s_{i-1}, y_{i-1}, c_i). \quad (3)$$

Bahdanau et al. (2014) propose an attention model to define the context c_i, inspired by the alignment model in statistical machine translation.

Given the last hidden state s_{i-1} and the encoding vectors $\mathbf{h}$, an attention model is based on a distribution consisting of α_{ij} as follows:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{m} \exp(e_{ik})},$$

where e_{ij} is computed by a feedforward neural network represented by a:

$$e_{ij} = a(s_{i-1}, h_j). \quad (4)$$

The quantity α_{ij} denotes the possibility of target word y_i aligns to the source word x_j encoded by h_j. According to α_{ij}, the context

vector c_i is defined as the weighted sum of $\mathbf{h}$:

$$c_i = \sum_{j=1}^{m} \alpha_{ij} h_j. \quad (5)$$

In this way, when translating the target word y_i, the decoder will pay more attention to its aligned source words with respect to the distribution $\alpha_i = \{\alpha_{i1}, \cdots, \alpha_{im}\}$. Figure 2 shows a slice of the entire architecture for NMT at timestamp i.

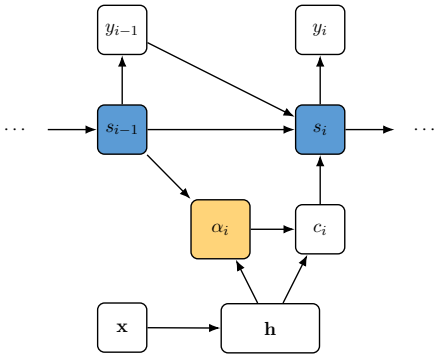


Figure 2: One slice of the architecture of Neural Machine Translation based on a generic attention.

Unfortunately, even though the entire translation $\mathbf{y}$ is available in training, during the inference it is unknown in advance but to be generated sequentially. Specifically, when calculating α_i, one can make use of the information only from $\mathbf{x}$ and $\mathbf{y}_{<i}$ but nothing from y_i. Therefore, it is difficult to certainly specify which source words should be aligned to an unknown target word y_i. This might lead to the inadequacy of the attention model (Liu et al., 2016a; Peter et al., 2017), as explained in Figure 1(a).

3 Target Foresight Attention

In order to alleviate the issue of inadequate modeling for attention in NMT, in this section, we propose the target foresight attention for NMT, which foresees some related information of the unknown target foresight word to improve its alignments regarding to source words. The basic idea of the proposed attention model includes two steps as following:

- It firstly introduce a model to predict some information of the target foresight word. (§3.1)
- It then feeds the predicted result about the foresight target word into the attention as an additional input. (§3.2)

Therefore, as shown in Figure 1(b), when translating the third word, if the prediction model shows it to be a "VBZ", the attention model is likely to align it to the verb words such as "huí shēng" rather than "rén shù" in the source side, and then the corrected word "rises" will be translated.

3.1 Target Foresight Prediction

Ideally, it is possible to build a model to directly predict the target foresight word itself. In practice, it will be inefficient due to its large vocabulary size. As a result, we instead build a model to predict the partial information of the target foresight word, such as part-of-speech (POS) tag or word cluster, which has limited vocabulary size. In this paper, we use the POS tag as the partial information of a target foresight word because POS tag is helpful to word alignment proved by Liu et al. (2005). Furthermore, predicting a POS tag is easier than a target foresight word, so the predicted result will be more reliable for the downstream application on attention.

Suppose u_i denotes a variable indicating the POS tag of a target foresight word y_i. Our aim is to define a prediction model of u_i prior to calculate the attention probability. For simplicity, this prediction model is generally represented as $\beta_i = P(u_i \mid \mathbf{y}_{<i}, \mathbf{x})$. We consider three variant prediction models in a coarse-to-fine manner as follows.

3.1.1 Model 1

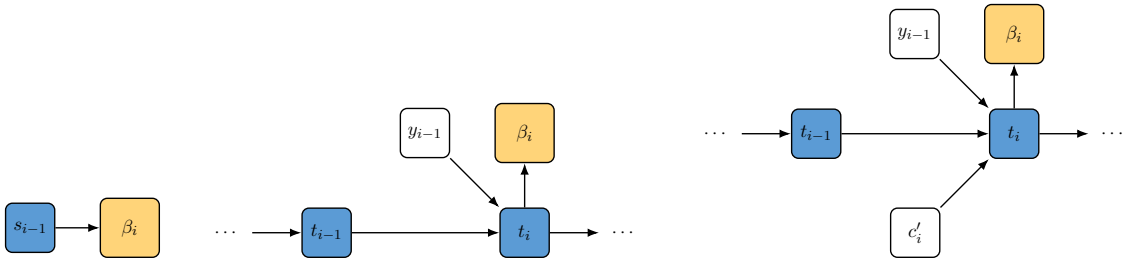
It is straightforward to define this prediction model directly based on the hidden states of the RNN in decoder by using a neural network. Formally, one can use the following equation:

$$\beta_i = P\left(u_i \mid \mathbf{y}_{<i}, \mathbf{x}\right) = \text{softmax}\left(\psi(y_{i-1}, s_{i-1})\right), \quad (6)$$

where ψ is implemented by a feedforward neural network. Note that Eq.(6) only depends on the decoding RNN hidden state s_{i-1} and it is very simple to implementation. Figure 3(a) shows its architecture.

3.1.2 Model 2

Unlike Eq.(6) relying on the same hidden s_{i-1} as the decoder, we design a specialized RNN



(a) model 1 (b) model 2 (c) model 3

Figure 3: The prediction coarse-to-fine models for target foresight information: (a) Model 1 using only the decoding hidden state s_{i-1}. (b) Model 2 using a hidden state t_i from a specialized RNN. (c) Models using a hidden state from a specialized RNN enhanced by the representation vector c'_i of $\mathbf{x}$ similar to Eq.(5).

to provide a particular hidden state for prediction of u_i. This improved prediction model is defined as follows:

$$\beta_i = P\left(u_i \mid \mathbf{y}_{<i}, \mathbf{x}\right) = \text{softmax}\left(\psi(y_{i-1}, t_i)\right), \tag{7}$$

where t_i is the hidden state of the specialized RNN defined by a GRU unit, i.e. $t_i = g(t_{i-1}, y_{i-1})$. This prediction model architecture is shown in Figure 3(b).

3.1.3 Model 3

In model 2, the specialized RNN for u_i only cares about the target sentence $\mathbf{y}$ and ignores the information from the source sentence $\mathbf{x}$. We define a fine-grained model by taking a context vector c'_i from $\mathbf{x}$ as an additional input:

$$\beta_i = P\left(u_i \mid \mathbf{y}_{<i}, \mathbf{x}\right) = \text{softmax}\left(\psi(y_{i-1}, t_i, c'_i)\right), \tag{8}$$

where c'_i is a context vector extracted from $\mathbf{x}$ in a way similar to c_i in Eq.(5),[2] and $t_i = g(t_{i-1}, y_{i-1}, c'_i)$ is the hidden state of the specialized RNN. The architecture of this model is shown in Figure 3(c).

3.2 Feeding the Prediction Model

Suppose we have the prediction result $P(u_i \mid \mathbf{y}_{<i}, \mathbf{x})$, then we consider to feed it into the attention model. Firstly, it is natural to feed the prediction into attention by using maximum a posteriori (MAP) strategy:

$$e_{ij} = a(s_{i-1}, h_j, z_i), \tag{9}$$

where a is the function for attention similar to Eq.(4) but includes an additional input z_i, which is the MAP result of $P(u_i \mid \mathbf{y}_{<i}, \mathbf{x})$:

$$z_i = \mathbf{z}\big(\operatorname*{argmax}_{u_i} P\left(u_i \mid \mathbf{y}_{<i}, \mathbf{x}\right)\big), \tag{10}$$

where $\mathbf{z}$ denotes the embeddings of the POS tags of target foresight words, and $\mathbf{z}(u_i)$ returns the embedding of a particular POS tag u_i.

Note that in Eq.(10) the accuracy of $P(u_i \mid \mathbf{y}_{<i}, \mathbf{x})$ is important to the attention model. For example, suppose at timestamp i, the ground-truth POS tag is "NN", but one has $P(u_i = \text{NN} \mid \mathbf{y}_{<i}, \mathbf{x}) = 0.4$ and $P(u_i = \text{VV} \mid \mathbf{y}_{<i}, \mathbf{x}) = 0.41$. In this case, the prediction model selects "VV" as the POS tag of the target foresight word and ignores the ground-truth tag "NN". Then the attention model takes this error signal and may align the target foresight word to a verb word. Subsequently, this might lead to a translation error.

Therefore, we propose another method to integrate the expected embedding of u_i according to $P(u_i \mid \mathbf{y}_{<i}, \mathbf{x})$ into attention as follows:

$$z_i = \sum_{u_i} \mathbf{z}(u_i) P\left(u_i \mid \mathbf{y}_{<i}, \mathbf{x}\right). \tag{11}$$

In this way, z_i can take into account all possible POS tags u_i including the ground-truth result.

Until now, we can obtain the entire architecture of the proposed target foresight attention based NMT (TFA-NMT), as shown in Figure 4. Comparing Figure 4 with Figure 2, the only difference is the variable z_i, which is

[2] In our preliminary experiments, we tried c_i, but we found c'_i performs better.

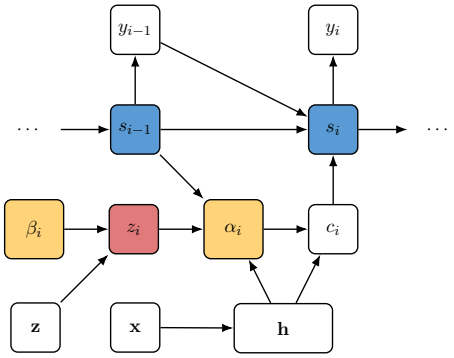


Figure 4: Neural machine translation with target Foresight attention. β_i is derived from Figure 3, z_i is from Eq.(10-11), and other nodes are similar to ones in Figure 2.

obtained from Eq.(10-11) and the prediction model as shown in Figure 3.

Note that the proposed TFA-NMT models the target foresight word, which is a future word regarding to the current time step, to conduct attention calculation. In this sense, it employs the idea of modeling future and thus resembles to the work in (Zheng et al., 2017). The main difference is that TFA-NMT models the future from the target side whereas Zheng et al. (2017) models the future from the source side. In addition, Weng et al. (2017) imposes a regularization term by using future words during training. Unlike our approach, their approach does not use future words during the inference because these words are unavailable. Anyway, it is possible to put both their approach and our approach together for further improvements.

3.3 Learning and Inference

Suppose a set of training data is denoted by $\{\langle \mathbf{x}^k, \mathbf{y}^k, \mathbf{u}^k \rangle \mid k = 1, \cdots, K\}$. Here $\mathbf{x}^k$, $\mathbf{y}^k$ and $\mathbf{u}^k$ denotes a source sentence, a target sentence and a POS tag sequence of $\mathbf{y}^k$, respectively. Then one can jointly train both the translation model for $\mathbf{y}^k$ and the prediction model for $\mathbf{u}^k$ by minimizing the loss function:

$$\ell = -\sum_k \sum_i \big(\log P(y_i^k \mid \mathbf{y}_{<i}^k, \mathbf{x}^k) + \lambda \log P(u_i^k \mid \mathbf{y}_{<i}^k, \mathbf{x}^k) \big), \quad (12)$$

where $P(y_i^k \mid \mathbf{y}_{<i}^k, \mathbf{x}^k)$ is the translation model similar to Eq.(2) with target foresight attention, and $P(u_i^k \mid \mathbf{y}_{<i}^k, \mathbf{x}^k)$ is the target foresight prediction model as defined in Eq.(6-8), respectively. $\lambda \geq 0$ is a hyper-parameter that balances the preference between the translation model and target foresight prediction model.

According to the training objective, the proposed TFA-NMT resembles to the multi-task learning, since it jointly learns two tasks similar to (Evgeniou and Pontil, 2004; Luong et al., 2015a). The difference of our approach is obviously: in this work the prediction result of one model is integrated into the other model, while in their works, two models only share some common hidden states.

In inference, we implement two different decoding methods according two different ways to integrate the foresight prediction model into attention as described in §3.2. For the MAP feeding style, we optimize u_i according to the loss function in Eq.(12) by beam search besides optimizing y_i. However, for the expectation feeding style, we maintain the standard beam search algorithm only regarding to the translation model, i.e. by setting $\lambda = 0$.

4 Experiments

We conduct experiments on Chinese-to-English and Japanese-to-English translation tasks. The specific analyses are based on Chinese-to-English task, and the generalization ability is shown by Japanese-to-English task. Case-insensitive 4-gram BLEU is used to evaluate translation quality, and the multi-bleu.perl is adopted as its implementation.

4.1 Setup

Data The training data for Chinese-to-English task consists of 1.8M sentence pairs from NIST2008 Open Machine Campaign, with 40.1M Chinese words and 48.3M English words respectively. The development set is chosen as NIST2002 (878 sentences) and the test sets are NIST2005 (1082 sentences), NIST2006 (1664 sentences), and NIST2008 (1357 sentences).

For Japanese-to-English translation, we adopt the data sets from NTCIR-9 patent translation task (Goto et al., 2013). The training data consists of 2.0M sentence pairs with 53.4M Japanese words and 49.3M English words, the development and test sets respectively contain 2000 sentences with a single ref-

Model	# Para.	Speed		Performance	
		Train	*Decode*	*BLEU*	*FPA*
NEMATUS	105M	2858.8	86.6	38.65	–
+2-LAYER	+6M	2522.5	84.1	38.57	–
+MODEL1	+2M	1844.9	72.0	38.83	69.03
+MODEL2	+12M	1666.1	70.1	39.26	69.95
+MODEL3	+27M	1485.2	59.1	**40.63**	**71.91**

Table 1: Speeds and performances of the proposed models. "Speed" is measured in words/second for both training and decoding, and performances are measured in terms of BLEU scores ("BLEU") and foresight prediction accuracy ("FPA") on the development set. Higher BLEU and FPA scores denote better performance.

erence, following (Goto et al., 2013; Liu et al., 2016b) for further comparison.

Implementation We compare the proposed models with two strong baselines from SMT and NMT:

- MOSES (Koehn et al., 2007): an open source phrased based translation system with default configuration.
- NEMATUS (Sennrich et al., 2017): an generic attention based NMT.

We implement the proposed models on top of NEMATUS. We use Stanford Log-linear Part-Of-Speech Tagger (Toutanova et al., 2003) to produce POS tags for the English side. For both Chinese-to-English and Japanese-to-English tasks, we limit the vocabularies to the most frequent 30K words for both sides. All the out-of-vocabulary words are mapped to a spacial token "UNK". Only the sentences of length up to 50 words are used in training, with 80 sentences in a batch. The dimension of word embedding is 620. The dimensions of both feed forward NN and RNN hidden layer are 1000. The beam size for decoding is 12, and the cost function is optimized by Adadelta with hyper-parameters suggested by Zeiler (2012). Particularly for TFA-NMT, the foresight embedding is also 620, and the hyper-parameter λ is 1.

4.2 Impact of Components

We conduct analyses on Chinese-to-English translation task, to investigate the impact of the added components and to figure out their best configuration for further testing in the next subsection.

4.2.1 Model Architectures

Table 1 lists the speeds and performances of the proposed models. Clearly the proposed approach improves the translation quality in all cases, although there are still considerable differences among the proposed variants.

Model Complexity The proposed models introduce a few parameters to the NMT baseline system NEMATUS, which has 105M parameters. The most complex model (i.e., MODEL3) introduces 27M new parameters, which are small compared with the baseline model. As seen, the proposed models significantly slows down the training speed, which we attribute to the new softmax operation over the foresight tags and more gradient operations associated with the new training objective, i.e., Eq.(12). For decoding, the most complex model reduces speed by around 30%, which is the cost of the proposed approach for improving translation quality.

Performance We measure the performance with BLEU and the result is shown in Table 1. MODEL1 marginally improves performance by guiding the decoder states to embed information for predicting foresight tags. MODEL2 achieves further improvement by introducing a new specific hidden layer to explicitly separate the predict function from the decoder states. MODEL3 achieves the best performance by adopting an independent attention model to attend corresponding source parts for foresight prediction, which may not be the same as the attended source parts for translation. We conduct the significant test using Kevin Gimpel's toolkit (Clark et al., 2011). We found that MODEL1 is not signif-

Type	Perc.	FPA	AER	
		OURS	BASE	OURS
Noun	30.13%	77.49	28.97	26.50
Verb	12.39%	71.94	37.06	33.93
Adj.	9.43%	55.99	34.67	31.86
Prep.	14.66%	79.40	**84.04**	**76.95**
Dete.	10.08%	72.06	80.15	76.51
Punc.	8.01%	74.89	**91.74**	**66.51**
Others	15.30%	81.22	53.64	39.11
All	100%	74.87	49.67	42.56

Table 2: Performances on syntactic categories. "BASE" denotes "NEMATUS", and OURS denotes the proposed model.

icantly better than baseline, but MODEL2 is significantly better with p<0.05 and MODEL3 is significantly better with p<0.01. Given that simply introducing an additional layer ("+2-LAYER") does not produce any improvement on this data, we believe the gain of our model is not only from the more introduced parameters. Besides, we augment the word embedding by concatenating the POS tag embedding, proposed by (Sennrich and Haddow, 2016), the BLEU is 38.96, which indicating the improvement of our model is not only from the POS tagging. In order to further validate the improvements of variant proposed models, we evaluate the foresight prediction accuracy (FPA) for three proposed prediction models. We found that the fine-grained MODEL3 achieves the best FPA, indicating a good estimated foresight is very important to obtain the gains in terms of BLEU.

4.2.2 Analysis on Syntactic Categories

In this experiment, we investigate which category of generated words benefit most from the proposed approach in terms of alignments measured by alignment error rate (AER) (Och, 2003). We carry out experiments on the evaluation dataset from (Liu and Sun, 2015), which contains 900 manually aligned Chinese-English sentence pairs. Following (Luong et al., 2015b), we force-decode both the bilingual sentences including source and reference sentences to obtain the attention matrices, and then we extract one-to-one alignments by picking up the source word with the highest alignment confidence as the hard alignment. As shown in Table 2, the AER improvements are modest for content words such as Noun, Verb, and adjective ("Adj.") words; but there are substantial improvements for function words such as preposition words ("Prep.") and punctuations ("Punc.").

Train (λ)	Decode	BLEU	$\bigtriangledown$
1	EXP	40.63	–
0	EXP	39.36	-1.27
1	MAP	40.34	-0.29

Table 3: Effect of foresight supervision signal in training (i.e., λ) and foresight representations in decoding: EXP for expectation and MAP for maximum a posteriori.

The reason can be explained as follows. The content words are easy to align with AER under 38 as shown in Table 2, and thus it is more difficult to gain over the BASE. On the other hand, as depicted in Table 2, function words are inherently more difficult than content words. These findings satisfy the linguistic intuition: content words tend to be less involved in multiple potential correspondences than function words, and function words tend to be attached to content words, as pointed out by Pianta and Bentivogli (2004). Fortunately, TFA-NMT can predict the POS tag for target foresight word with high confidence and thus it can improve the alignment quality by using of POS tags, which is useful for alignment task (Liu et al., 2005).

It is surprising that the AER for "Prep.", "Det." and "Punc." is relatively low especially for BASE. The main reason can be explained from the quantities y_{i-1}, s_i, and c_i in Eq.(2) as follows. These highly frequent function words are usually easy to be translated by using the history information from y_{i-1} and s_i even if c_i is not confident enough. For example, it is relatively easy to guess the "comma" by using the history words in language model task, where there are no bilingual information at all. Therefore, during the training, the model tries to adjust the parameters for highly frequent words from y_{i-1} and s_i while neglecting the attention model.

4.2.3 Foresight Strategies

Table 3 shows the performances of different foresight strategies in both training and de-

coding. Without an explicit objective to guide the training of foresight prediction model (i.e., $\lambda = 0$), the performance decreases by 1.27 BLEU points. When feeding the best foresight predicted result to the attention model (i.e., MAP), the performance decreases by 0.29 BLEU points. We attribute this to the propagation of prediction errors, which can be alleviated by using a weighted representation of all predicted results (i.e., EXP).

In the following experiments, we use "$\lambda = 1$ and EXP" as the default setting for the final system TFA-NMT.

4.3 Main Results

Chinese-to-English Task Table 4 shows the translation performances for the Chinese-to-English translation task. As seen, the proposed approach significantly outperforms the baseline system (i.e., NEMATUS) in all cases, demonstrating the effectiveness and university of our model.

Japanese-to-English Task Table 5 shows the translation quality of the NMT baseline and our TFA-NMT on Japanese-to-English task. From the table, we can see that our model still achieves a significant improvement of 1.22 and 1.31 BLEU points on the development and test set, respectively. This shows that the proposed approach works well across different language pairs.

5 Related Work

Attention model becomes a standard component for many applications due to its ability of dynamically selecting the informative context from sequential representations. For example, Xu et al. (2015) propose an attention based neural network for image caption task and advance the state-of-the-art results; Yin et al. (2015) put the attention structure between a pair of convolution networks for answer selection, paraphrase identification and textual entailment tasks. In the context of machine translation, the idea of attention based neural networks has been pioneered by Bahdanau et al. (2014); Luong et al. (2015b) and achieved impressive results over the traditional statistical machine translation. Since then many research works have been devoted to improve the neural machine translation by enhancing attention models.

Tu et al. (2016) design a coverage vector for the translation history and then integrates it into the attention model. Similarly, Meng et al. (2016) maintain a tag vector to keep track of the attention history and Sankaran et al. (2016) memorize historical alignments and accumulate them as temporal memory to improve the attention model. In addition, Zhang et al. (2017) improve the attention with a gated operator for encoding states and a decoding state, and and Dutil et al. (2017) enhance attention through a planning mechanism. Furthermore, Feng et al. (2016) adopt a recurrent structure for attention to take long-term dependencies into account, Zhou et al. (2017) propose a look-ahead attention by additionally modeling the translation history, and Cohn et al. (2016) incorporate structural biases into attention models. Recently Chen et al. (2017) introduce the syntactic knowledge into attention models. These works are essentially similar to the propose approach, since we introduce auxiliary information from a target foresight word into the attention model. However, there is a significant difference between our approach and their approaches. Our auxiliary information biases to the word to be translated at next timestep while theirs biases to the information available so far at the current timestep, and thereby our approach is orthogonal to theirs.

The works mentioned above improve the attention models by access auxiliary information, and thus they modify the structure of attention models in both inference and learning. In contrast, Mi et al. (2016); Liu et al. (2016b); Chen et al. (2016) maintain the structure of the attention models in inference but utilize some external signals to supervise the outputs of attention models during the learning. They improve the generalization abilities of attention models by use of the external aligners as the signals, which typically yield alignment results accurate enough to guide the learning of attention.

6 Conclusion

It has been argued that the traditional attention model in neural machine translation suf-

System	Model	Dev	MT05	MT06	MT08	Ave.
(Liu et al., 2016b)	MOSES	–	35.4	33.7	25.0	31.37
	NMT-J	–	36.8	36.9	28.5	34.07
(Liu et al., 2016a)	SA-NMT	40.0	**37.8**	37.6	29.9	35.10
This work	NEMATUS	38.65	36.32	36.10	28.24	33.55
	TFA-NMT	**40.63**	37.70	**38.01**	**30.12**	**35.28**

Table 4: Evaluation of translation performance on Chinese-to-English task.

System	Model	Dev	Test
(Liu et al., 2016b)	MOSES	28.6	30.2
	NMT-J	33.0	34.1
This work	NEMATUS	33.92	35.01
	TFA-NMT	35.14	36.32

Table 5: Evaluation of translation performance on Japanese-to-English task.

fers from model inadequacy due to the lack of information from the target foresight word (Peter et al., 2017; Liu et al., 2016a). To address this issue, this paper proposes a new attention model, which can serve for both alignment and translation tasks, by implicitly making use of the target foresight word. Empirical experiments on Chinese-to-English and Japanese-to-English tasks demonstrate that the proposed attention based NMT delivers substantial gains in terms of both BLEU and AER scores.

In future work, it is promising to exploit other target foresight information such as word cluster besides the POS tags in this paper, and it is also interesting to apply this idea on top of other attention models such as the local attention in Luong et al. (2015b).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Kehai Chen, Rui Wang, Masao Utiyama, Eiichiro Sumita, and Tiejun Zhao. 2017. Syntax-directed attention for neural machine translation. *arXiv preprint arXiv:1711.04231* .

Wenhu Chen, Evgeny Matusov, Shahram Khadivi, and Jan-Thorsten Peter. 2016. Guided alignment training for topic-aware neural machine translation. *arXiv preprint arXiv:1607.01628* .

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 263–270.

Jonathan H Clark, Chris Dyer, Alon Lavie, and Noah A Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*. Association for Computational Linguistics, pages 176–181.

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating structural alignment biases into an attentional neural translation model. *arXiv preprint arXiv:1601.01085* .

Francis Dutil, Caglar Gulcehre, Adam Trischler, and Yoshua Bengio. 2017. Plan, attend, generate: Planning for sequence-to-sequence models. *arXiv preprint arXiv:1711.10462* .

Theodoros Evgeniou and Massimiliano Pontil. 2004. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 109–117.

Shi Feng, Shujie Liu, Nan Yang, Mu Li, Ming Zhou, and Kenny Q Zhu. 2016. Improving attention modeling with implicit distortion and fertility for machine translation. In *COLING*. pages 3082–3092.

Isao Goto, Ka-Po Chow, Bin Lu, Eiichiro Sumita, and Benjamin K Tsou. 2013. Overview of the patent machine translation task at the ntcir-10 workshop. In *NTCIR*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting*

of the ACL on interactive poster and demonstration sessions. Association for Computational Linguistics, pages 177–180.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*. Association for Computational Linguistics, pages 48–54.

Lemao Liu, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. 2016a. Neural machine translation with supervised attention. *arXiv preprint arXiv:1609.04186* .

Lemao Liu, Masao Utiyama, Andrew M Finch, and Eiichiro Sumita. 2016b. Agreement on target-bidirectional neural machine translation. In *HLT-NAACL*. pages 411–416.

Yang Liu, Qun Liu, and Shouxun Lin. 2005. Log-linear models for word alignment. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 459–466.

Yang Liu and Maosong Sun. 2015. Contrastive unsupervised word alignment with non-local features. In *AAAI*. pages 2295–2301.

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

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015b. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* .

Fandong Meng, Zhengdong Lu, Hang Li, and Qun Liu. 2016. Interactive attention for neural machine translation. *arXiv preprint arXiv:1610.05011* .

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016. Supervised attentions for neural machine translation. *arXiv preprint arXiv:1608.00112* .

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*. Association for Computational Linguistics, pages 160–167.

Jan-Thorsten Peter, Arne Nix, and Hermann Ney. 2017. Generating alignments using target foresight in attention-based neural machine translation. *The Prague Bulletin of Mathematical Linguistics* 108(1):27–36.

Emanuele Pianta and Luisa Bentivogli. 2004. Knowledge intensive word alignment with knowa. In *Proceedings of the 20th international conference on Computational Linguistics*. Association for Computational Linguistics, page 1086.

Baskaran Sankaran, Haitao Mi, Yaser Al-Onaizan, and Abe Ittycheriah. 2016. Temporal attention model for neural machine translation. *arXiv preprint arXiv:1608.02927* .

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, et al. 2017. Nematus: a toolkit for neural machine translation. *arXiv preprint arXiv:1703.04357* .

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. *arXiv preprint arXiv:1606.02892* .

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*. Association for Computational Linguistics, pages 173–180.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. *arXiv preprint arXiv:1601.04811* .

Rongxiang Weng, Shujian Huang, Zaixiang Zheng, XIN-YU DAI, and Jiajun CHEN. 2017. Neural machine translation with word predictions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 136–145.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*. pages 2048–2057.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2015. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193* .

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

Biao Zhang, Deyi Xiong, and Jinsong Su. 2017. A gru-gated attention model for neural machine translation. *arXiv preprint arXiv:1704.08430* .

Zaixiang Zheng, Hao Zhou, Shujian Huang, Lili Mou, Xinyu Dai, Jiajun Chen, and Zhaopeng Tu. 2017. Modeling past and future for neural machine translation. *arXiv preprint arXiv:1711.09502* .

Long Zhou, Jiajun Zhang, and Chengqing Zong. 2017. Look-ahead attention for generation in neural machine translation. *arXiv preprint arXiv:1708.09217* .

Context Sensitive Neural Lemmatization with Lematus

Toms Bergmanis
School of Informatics
University of Edinburgh
T.Bergmanis@sms.ed.ac.uk

Sharon Goldwater
School of Informatics
University of Edinburgh
sgwater@inf.ed.ac.uk

Abstract

The main motivation for developing context-sensitive lemmatizers is to improve performance on unseen and ambiguous words. Yet previous systems have not carefully evaluated whether the use of context actually helps in these cases. We introduce Lematus, a lemmatizer based on a standard encoder-decoder architecture, which incorporates character-level sentence context. We evaluate its lemmatization accuracy across 20 languages in both a full data setting and a lower-resource setting with 10k training examples in each language. In both settings, we show that including context significantly improves results against a context-free version of the model. Context helps more for ambiguous words than for unseen words, though the latter has a greater effect on overall performance differences between languages. We also compare to three previous context-sensitive lemmatization systems, which all use pre-extracted edit trees as well as hand-selected features and/or additional sources of information such as tagged training data. Without using any of these, our context-sensitive model outperforms the best competitor system (Lemming) in the full-data setting, and performs on par in the lower-resource setting.

1 Introduction

Lemmatization is the process of determining the dictionary form of a word (e.g. *swim*) given one of its inflected variants (e.g. *swims*, *swimming*, *swam*, *swum*). Data-driven lemmatizers face two main challenges: first, to generalize beyond the training data in order to lemmatize unseen words; and second, to disambiguate ambiguous wordforms from their sentence context. In Latvian, for example, the wordform "*ceļu*" is ambiguous when considered in isolation: it could be an inflected variant of the verb "*celt*" (*to lift*) or the nouns "*celis*" (*knee*) or "*ceļš*" (*road*); without context, the lemmatizer can only guess.

By definition, sentence context (or latent information derived from it, such as the target word's morphosyntactic tags) is needed in order to correctly lemmatize ambiguous forms such as the example above. Previous researchers have also assumed that context should help in lemmatizing unseen words (Chrupała, 2006; Müller et al., 2015)—i.e., that the context contains useful features above and beyond those in the wordform itself. Nevertheless, we are not aware of any previous work that has attempted to quantify how much (or even whether) context actually helps in both of these cases. Several previous papers on context-sensitive lemmatization have reported results on unseen words (Chrupała, 2006; Chrupała et al., 2008; Müller et al., 2015; Chakrabarty et al., 2017), and some have compared versions of their systems that use context in different ways (Müller et al., 2015; Chakrabarty et al., 2017), but there are few if any direct comparisons between context-sensitive and context-free systems, nor have results been reported on ambiguous forms.

This paper presents *Lematus*—a system that adapts the neural machine translation framework of Sennrich et al. (2017) to learn context sensitive lemmatization using an encoder-decoder model. Context is represented simply using the character contexts of each form to be lemmatized, meaning that our system requires fewer training resources than previous systems: only a corpus with its lemmatized forms, without the need for POS tags (Chrupała et al., 2008; Müller et al., 2015) or word embeddings trained on a much larger corpus (Chakrabarty et al., 2017). We evaluate Lematus on data from 20 typologically varied languages, both using the full training data from the Universal Dependencies project (Nivre et al., 2017), as well as a lower-resource scenario with only 10k training tokens per language. We compare results to three previous systems and to a context-free version of our own system, including results on both unseen

Proceedings of NAACL-HLT 2018, pages 1391–1400
New Orleans, Louisiana, June 1 - 6, 2018.

and ambiguous words. We also examine the extent to which the rate of unseen and ambiguous words in a language can predict lemmatization performance.

On average across the 20 languages, the context-sensitive version of Lematus achieves significantly higher lemmatization accuracy than its context-free counterpart in both the low-resource and full-data settings. It also outperforms the best competitor system (Lemming; Müller et al. 2015) in the full-data setting, and does as well as Lemming in the low-resource setting. Thus, even without explicitly training on or predicting POS tags, Lematus seems able to implicitly learn similar information from the raw character context.

Analysis of our full-data results shows that including context in the model improves its accuracy more on ambiguous words (from 88.8% to 92.4% on average) than on unseen words (from 83.6% to 84.3% on average). This suggests that, to the extent that unseen words can be correctly lemmatized at all, the wordform itself provides much of the information needed to do so, and Lematus effectively exploits that information—indeed, Lematus without context outperforms all previous context-sensitive models on lemmatizing unseen words.

Finally, our cross-linguistic analysis indicates that the proportions of unseen words and ambiguous words in a language are anti-correlated. Altogether, then, our results suggest that context-free neural lemmatization is surprisingly effective, and may be a reasonable option if the language contains many unseen words but few ambiguous ones. Context is likely to help in most languages, but the main boost is for languages with higher ambiguity.

2 Background and Baseline Systems

Early work on context-sensitive lemmatization focused on disambiguation: given a set of analyses produced by a hand-built morphological analyzer (typically including both lemmas and morphosyntactic tags), choose the best one in context (Oflazer and Kuruöz, 1994; Ezeiza et al., 1998; Hakkani-Tür et al., 2002). Here, we focus on systems learning to generate the lemmas and tags without a pre-existing analyzer (Erjavec and Džeroski, 2004; Chrupała, 2006). The three systems we use as baselines follow Chrupała (2006) in treating the task as a classification problem, where the system learns to choose which of a set of *edit scripts* or *edit trees* (previously induced from the aligned wordform-lemma pairs) should be applied to transform each wordform into the correct lemma.

Two of our baselines, **Morfette**[1] (Chrupała et al., 2008) and **Lemming**[2] (Müller et al., 2015), learn from morphologically annotated corpora to jointly tag each word and lemmatize it by choosing an edit script. Morfette consists of two log-linear classifiers—one for lemmatization and one for tagging—which are combined using beam search to find the best sequence of lemma-tag pairs for all words in the input sentence. Lemming (which proves to be the strongest baseline) also consists of two log-linear components (a classifier for lemmatization and a sequence model for tagging), which are combined either using a pipeline (first tag, then lemmatize) or through joint inference. The lemmatization model uses a variety of features from the edit trees, alignments, orthography of the lemma, and morphosyntactic tags.

In experiments on six languages, Müller et al. (2015) showed that the joint Lemming model worked better than the pipelined model, and that adding morphosyntactic features helped. They also demonstrated improvements over an earlier context-free baseline model (Jiampojamarn et al., 2008). However, they did not evaluate on ambiguous forms, nor directly compare context-sensitive and context-free versions of their own model.

Our third baseline, **Ch-2017**[3] (Chakrabarty et al., 2017) uses a neural network rather than a log-linear model, but still treats lemmatization as a classification task to choose the correct edit tree. (Like our model, Ch-2017 does not perform morphological tagging.) The model composes syntactic and semantic information using two successive bidirectional GRU networks. The first bidirectional GRU network is similar to the character to word model by Ling et al. (2015) and learns syntactic information. The semantic information comes from word embeddings pre-trained on much larger corpora. The second GRU uses a composition of the semantic and syntactic embeddings for the edit tree classification task.

Rather than treating lemmatization as classification, our own model is inspired by recent work on morphological reinflection. As defined by two recent Shared Tasks (Cotterell et al., 2016, 2017), a morphological reinflection system gets as input

[1] https://sites.google.com/site/morfetteweb/

[2] http://cistern.cis.lmu.de/lemming

[3] https://github.com/onkarpandit00786/neural-lemmatizer

some inflected wordform (and possibly its morphosyntactic tags) along with a set of target tags. The system must produce the correct inflected form for the target tags. In the 2016 SIGMORPHON Shared Task, various neural sequence-to-sequence models gave the best results (Aharoni et al., 2016; Kann and Schütze, 2016; Östling, 2016). We base our work closely on one of these (Kann and Schütze, 2016), which also won one of the 2017 tasks (Bergmanis et al., 2017). Our lemmatization task can be viewed as a specific type of reinflection, but instead of assuming that tags are given in the input (or that the system simply has to guess the tags from the wordform itself, as in some of the Shared Tasks), we investigate whether the information available from the tags can instead be inferred from sentence context.

3 Model Description

Our model is based on the network architecture proposed by Sennrich et al. (2017), which implements an attentional encoder-decoder architecture similar to that of Bahdanau et al. (2015). Namely, our model is a deep attentional encoder-decoder with a 2-layer bidirectional encoder with a gated recurrent unit (GRU) (Cho et al., 2014) and a 2-layer decoder with a conditional GRU (Sennrich et al., 2017) in the first layer followed by a GRU in the second layer. For more architectural details see (Sennrich et al., 2017).

A default implementation of this architecture is available in the *Nematus* toolkit,[4] which we used as our starting point. However, Sennrich et al. (2017) used their model for machine translation, while we work on lemmatization. Since our task is closer to the problem of morphological reinflection described above, we changed some of the default model parameters to follow those used in systems that performed well in the 2016 and 2017 SIGMORPHON Shared Tasks (Kann and Schütze, 2016; Bergmanis et al., 2017). Specifically, we reduced the number of hidden units to 100 and the encoder and decoder embedding size to 300.

The input sequence is a space-separated character representation of a word in its N-character left and right sentence context. For example, with $N = 15$, the Latvian word *ceļu* (the genitive plural of the noun *ceļš*, meaning *road*) could be input as:

```
s a k a <s> p a š v a l d ī b u
<lc> c e ļ u <rc>
u n <s> i e l u <s> r e ģ i s t r
```

where <s>, <lc>, <rc> stand for word boundary, left and right context markers respectively. The target output is a sequence of characters forming the lemma of the word: `c e ļ š`

4 Datasets

We contend that the difficulty of the lemmatization task largely depends on three factors: morphological productivity, lexical ambiguity and morphological regularity. One aim of our work is to investigate the extent to which it is possible to predict lemmatization performance for a particular language by operationalizing and measuring these properties. Therefore in this section we provide statistics and some analysis of the datasets used in our experiments. We use the standard splits of the Universal Dependency Treebank (UDT) v2.0[5] (Nivre et al., 2017) datasets for 20 languages: Arabic, Basque, Croatian, Dutch[6], Estonian, Finnish, German, Greek, Hindi, Hungarian, Italian, Latvian, Polish, Portuguese, Romanian, Russian, Slovak, Slovene, Turkish and Urdu. See Figure 1 for training and development data sizes.

Because the amount of training data varies widely between languages, we perform some of our language analysis (and later, system evaluation) on a subset of the data, where we use only the first 10k tokens in each language for training. The 10k setting provides a clearer comparison between languages in terms of their productivity, ambiguity, and regularity, and also gives a sense of how much training data is needed to achieve good performance.

One of the main purposes of data-driven lemmatization is to handle unseen words at test time, yet languages with differing morphological productivity will have very different proportions of unseen words. Figure 2 shows the percentage of tokens in the development sets of each language that are not seen in training. Two conditions are given: the full training/development sets, and train/dev sets that are controlled in size across languages. For

[4] https://github.com/EdinburghNLP/nematus

[5] UTD v2.0 datasets are archived at http://hdl.handle.net/11234/1-1983

[6] We use UDT v2.1 dataset for Dutch due to inconsistencies in v2.0.

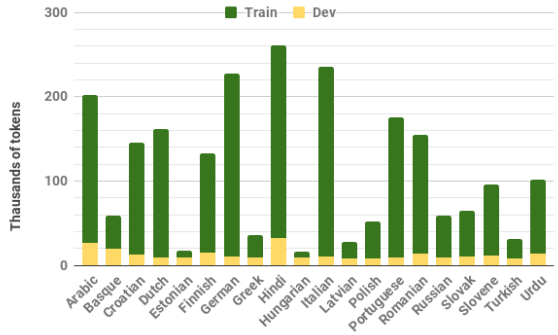


Figure 1: Training and development set sizes for each language, in thousands.

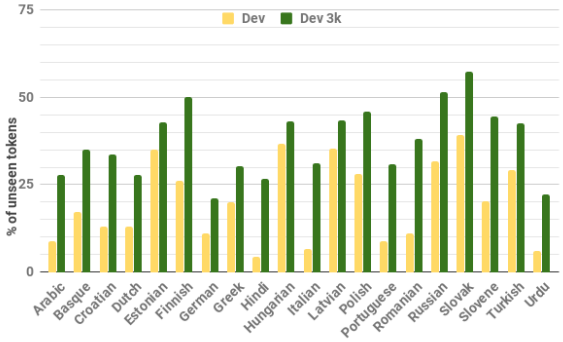


Figure 2: Percent of tokens unseen in training. Dev (yellow): within full development sets with respect to the full training sets. Dev 3k (green): within the first 3k tokens of development sets with respect to the first 10k tokens of training sets.

the languages with large data sets, the percentage of unseen words is (unsurprisingly) higher when training data is reduced to 10k. However, these differences are often small compared to the differences between languages, suggesting that productivity is likely to affect lemmatization performance as much as training data size.

Lexical ambiguity is the other major motivation for context-sensitive lemmatization. To quantify how frequently lemmatizers have to rely on context, Figure 3 shows the percentage of ambiguous tokens in each language, in either the full or reduced training sets. We define ambiguity empirically: ambiguous tokens are wordforms occurring with more than one lemma within the training set.

Overall, the level of measured ambiguity tends to be lower than the proportion of unseen tokens. Many of the languages with high productivity (e.g., Russian, Slovak, Slovene, Turkish) have low levels of ambiguity, while others (Arabic, Urdu) trend the opposite way. Indeed, across all 20 languages,

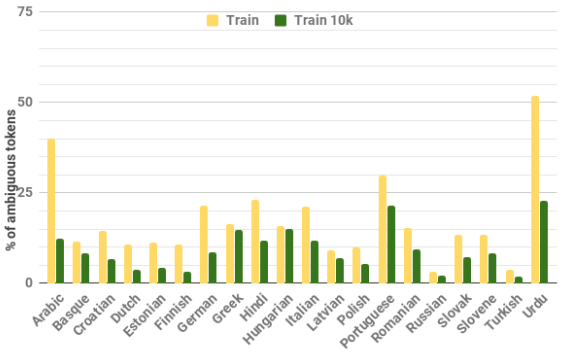


Figure 3: Percent of ambiguous tokens within the first 10k tokens of training sets and full training sets. Ambiguous tokens are word forms occurring with more than one lemma in the training set.

the levels of productivity and ambiguity are negatively correlated, with a rank correlation of -0.57 after controlling for training data size.[7] This is not surprising, since given a set of morphosyntactic functions, they must either be expressed using distinct forms (leading to higher productivity) or non-distinct forms (leading to higher ambiguity).

The final characteristic that we would expect to make some languages easier than others is morphological regularity, but it is unclear how to measure this property directly without an in-depth understanding of the morphophonological rules of a language. Nevertheless, the presence of many irregular forms, or other phenomena such as vowel harmony or spelling changes, complicates lemmatization and will likely affect accuracy.

5 Experimental Setup

Training Parameters[8] We use a mini batch size of 60 and a maximum sequence length of 75. For training we use stochastic gradient descent, Adadelta (Zeiler, 2012), with a gradient clipping threshold of 1.0, recurrent Bayesian dropout probability 0.2 (Gal and Ghahramani, 2016) and weight normalization (Salimans and Kingma, 2016). We use early stopping with patience 10 (Prechelt, 1998). We use the first 10 epochs as a burn-in period, after which at the end of every second epoch

[7] That is, the correlation is computed between the values in Figure 2 Dev 3k (unseen words wrt the first 10k training tokens for each language) and Figure 3 Train 10k (ambiguous words in the first 10k training tokens for each language). The correlation is significantly different from zero with $p < 0.01$.

[8] Training parameters were tunned/verified on the standard splits of UDT training and development sets for Spanish and Catalan, therefore the results on these languages are not included in our evaluation.

we evaluate the current model's lemmatization exact match accuracy on the development set and keep this model if it performs better than the previous best model. When making predictions we use beam-search decoding with a beam of size 12.

Baselines To train models we use the default settings for Morfette and Lemming. Ch-2017 requires word embeddings, for which we use *fastText*[9] (Bojanowski et al., 2017). For Ch-2017 we set the number of training epochs to 100 and implement early stopping with patience 10.[10] We leave the remaining model parameters as suggested by Chakrabarty et al. (2017).

We also use a lookup-based baseline (**Baseline**). For words that have been observed in training, it outputs the most frequent lemma (or the first observed lemma, if the options are equally frequent). For unseen words it outputs the wordform itself as the hypothesized lemma.

Context Representation We aim to use a context representation that works well across multiple languages, rather than to tune the context individually to each language. In preliminary experiments, we explored several different context representations: words, sub-word units, and N surrounding characters, for different values of N. These experiments were carried out on only six languages. Three of these (Latvian, Polish and Turkish) were also used in our main experiments, while three (Bulgarian, Hebrew, and Persian) were not, due to problems getting all the baseline systems to run on those languages.

For the word level context representation (**Words**), we use all words in the left and the right sentence contexts. For the character level context representations (**N-Ch**) we experiment with $N = 0$, 5, 10, 15, 20, or 25 characters of left and right contexts. For the sub-word unit context representation, we use byte pair encoding (**BPE**) (Gage, 1994), which has shown good results for neural machine translation (Sennrich et al., 2016). BPE is a data compression algorithm that iteratively replaces the most frequent pair of symbols (here, characters) in a sequence with a single new symbol. BPE has a single parameter—the number of merge operations. Suitable values for this parameter depend on the application and vary from 10k in language modeling (Vania and Lopez, 2017) to 50k in machine translation (Sennrich et al., 2016). We aim to use BPE to extract a few salient and frequently occurring strings, such as affixes, therefore we set the number of BPE merge operations to 500. We use BPE-encoded left and right sentence contexts that amount up to 20 characters of the original text.

Since we hoped to use context to help with ambiguous words, we looked specifically at ambiguous word performance in choosing the best context representation.[11] Table 1 summarizes Lematus' performance on ambiguous tokens using different sentence context representations. There is no context representation that works best for all six languages, but the 20-Ch system seems to work reasonably well in all cases, and the best on average. We therefore use the 20-Ch context in our main experiments.

Note that this choice was based on a relatively small number of experiments and it is quite possible that further tuning the BPE parameter, or the number of BPE units or words of context (or tuning separately for each language) could lead to better overall results.

Evaluation To evaluate models, we use test and development set lemmatization exact match accuracy. When calculating lemmatization accuracy we ignore casing of the tokens and ommit punctuation tokens and those tokens that contain digits or any of the following characters: `@+._/`.

6 Results and Discussion

Results on Complete Datasets Development set accuracies for all languages and systems in the full data setting are provided in Figure 4a, with results on unseen and ambiguous words in Figures 4b and 4c. Overall, Lematus 20-Ch outperforms the previous systems, Morfette, Lemming and Ch-2017, on 20, 15 and 20 languages respectively. In addition, Figure 4 makes it clear that the major benefit of all the systems over the baseline is for unseen words: in fact, for ambiguous words, the baseline even outperforms some of the systems in a few languages. Comparing the two versions of Lematus, we can see that Lematus 20-Ch does consistently better

[9] https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md

[10] We do so because it is unclear what stopping criterion was used by Chakrabarty et al. (2017) Their suggested default for the number of training epochs is 6, yet the values used in their experiments vary from 15 for Hindi to 80 for Bengali.

[11] The percentage of ambiguous tokens in the training sets of Bulgarian, Hebrew and Persian are 8.4%, 16.6% and 7.6% respectively; for the other languages, see Figure 3.

	Baseline	0-Ch	5-Ch	10-Ch	15-Ch	20-Ch	25-Ch	BPE	Words
Bulgarian	81.1	83.0	79.7	88.9	88.2	**89.2**	88.5	89.2	84.2
Hebrew	**95.3**	95.0	82.5	84.9	86.0	86.3	85.4	84.4	75.5
Latvian	73.8	76.6	70.1	73.2	73.9	**74.8**	71.1	71.6	66.2
Persian	**94.4**	92.5	90.5	91.5	91.0	92.5	92.5	93.0	88.0
Polish	90.6	91.7	84.0	84.0	93.0	**93.6**	83.5	85.6	83.0
Turkish	78.6	80.9	75.9	77.9	85.5	**85.9**	79.3	75.9	73.8
Average	85.6	86.6	80.5	83.4	86.3	**87.1**	83.4	83.3	78.5

Table 1: Lemmatization exact match accuracy on ambiguous tokens of dev sets, for baseline and for Lematus using various context representations: N characters, Byte Pair Encoding units, or words.

	Dev				Test	10k:Dev	10k:Test
	All	**Unseen**	**Ambig**	**SeenUA**	**All**	**All**	**All**
Baseline	85.8	39.6	88.0	**99.2***†‡	86.1	74.4	74.4
Morfette	92.9	75.7	91.4	98.9	93.1	86.8	86.5
Lemming	94.1	81.4	**92.4**†	98.8	94.1	87.5	87.3
Ch-2017	90.8	75.0	90.7	96.2	89.8	80.2	79.0
Lematus 0-Ch	94.3	83.6*	88.8	98.9	94.2	87.1	86.6
Lematus 20-Ch	**95.0***†	**84.3***†	**92.4**†	98.8	**94.9***†	**88.4**†	**87.8**†

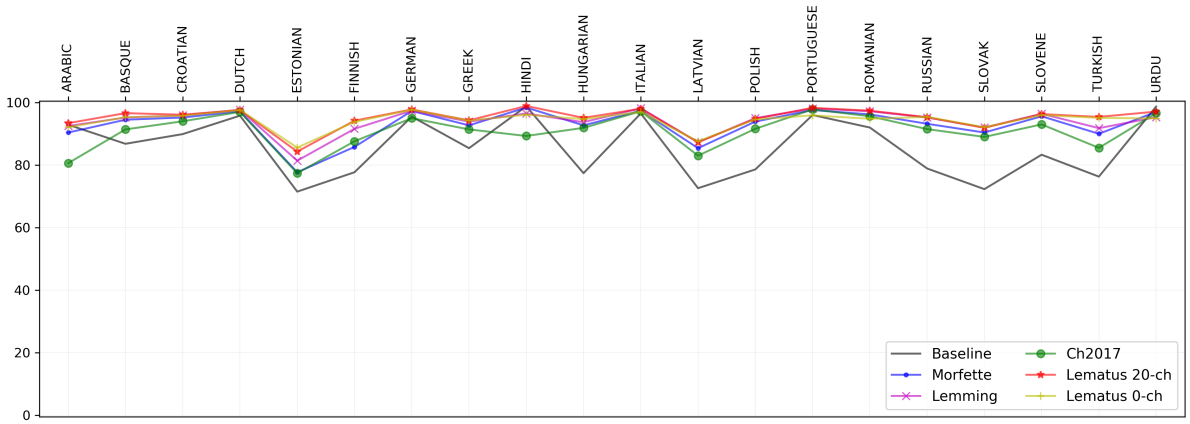
Table 2: Lemmatization exact match accuracy, averaged across all 20 languages. In the full training scenario (first five columns) results are given for All, Unseen, Ambiguous, and Seen Unambiguous tokens. (Note that ambiguity is empirical: is a type seen with more than one lemma in training?) We compare Lematus with/without context (20-Ch/0-Ch), the most frequent lemma baseline, and three previous systems. The numerically highest score in each column is bold; *, †, and ‡ indicate statistically significant improvements over Lemming, Lematus 0-Ch and 20-Ch, respectively (all $p < 0.05$; see text for details).

on ambiguous tokens than Lematus 0-Ch, whereas their performance on unseen tokens (and thus, overall) is much more similar. In fact, on unseen words, Lematus 0-Ch outperforms the context-sensitive baselines Morfette, Lemming and Ch-2017 on 18, 12 and 17 languages respectively. These results suggest that a good context-free model can do surprisingly well on unseen words, and the added model complexity and annotation requirements of earlier context-sensitive models are not always justified.

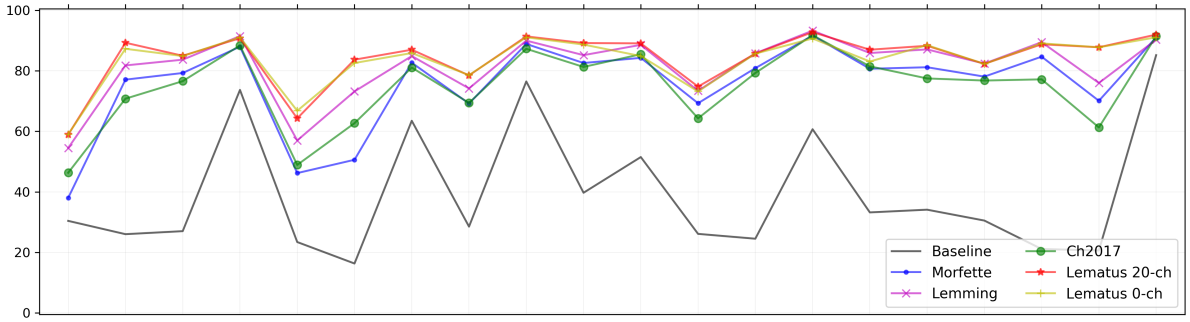
As further evidence of these claims, we summarize in Table 2 each system's average performance over all languages for both the development and test sets. In addition to performance breakdown into unseen and ambiguous words we also report each system's performance on tokens that were both seen and unambiguous in training. No system achieves 100% accuracy on seen unambiguous tokens—even the lookup baseline achieves only 99%, indicating that about 1% of tokens that appeared unambiguous in training occur with a previously unseen lemma in the development set. In principle, context-based systems could outperform the baseline on these words, but in practice none of them do. Indeed, switching to a dictionary lookup baseline for seen unambiguous words would slightly improve the performance of all models (though it would not change the overall ranking of the systems).

We tested for statistically significant differences between the results of Lemming (the numerically best-performing competitor system) and our two systems (Lematus 0-Ch and Lematus 20-Ch) using a Monte Carlo method: for each comparison (say, between 0-Ch and 20-Ch on unseen words), we generated 10000 random samples, where each sample randomly swapped the two systems' results for each language with probability .5. We then obtained a p-value by computing the proportion of samples for which the difference in average results was at least as large as the difference observed in our experiments.

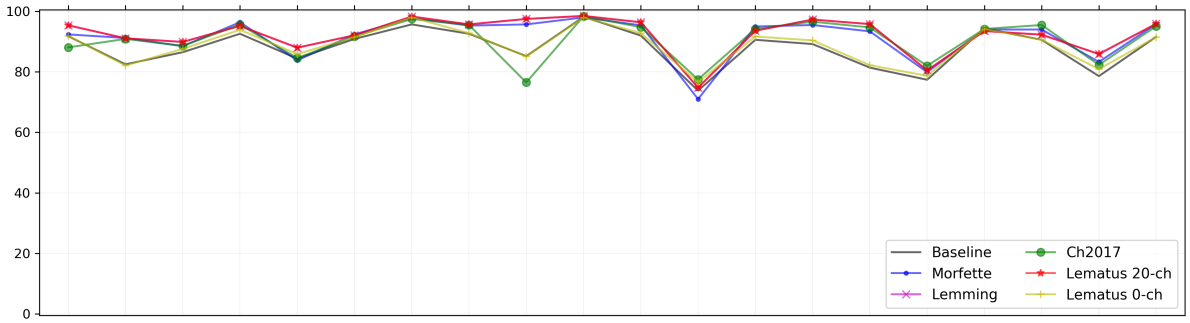
Because the results of 0-Ch and 20-Ch are highly correlated across languages, all differences between these systems, except for results on seen unambiguous tokens, are significant ($p < 0.01$ for dev set All, $p < 0.05$ for Unseen, $p < 0.001$ for Ambig, and $p < 0.01$ for test set All; $p > 0.1$ for



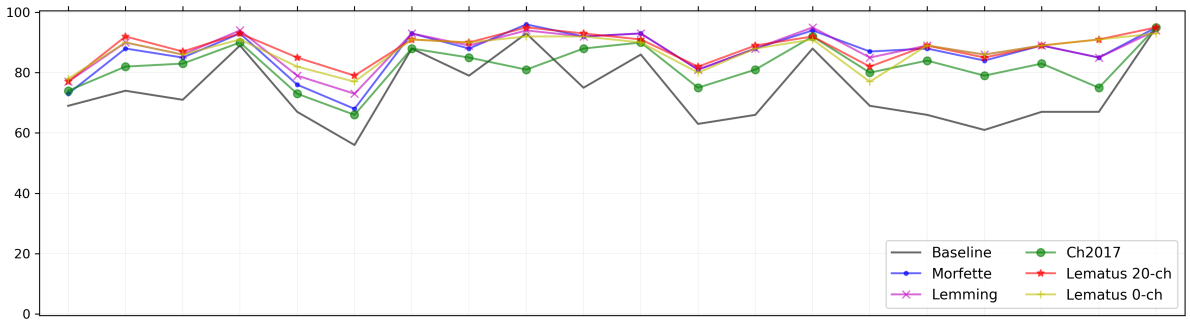
(a) **All tokens.** Models were trained on full training sets.



(b) **Unseen tokens.** Models were trained on full training sets.



(c) **Ambiguous tokens.** Models were trained on full training sets.



(d) **All tokens**. Models were trained on the first **10K** of the training sets.

Figure 4: Lemmatization exact match accuracy on development sets for each language.

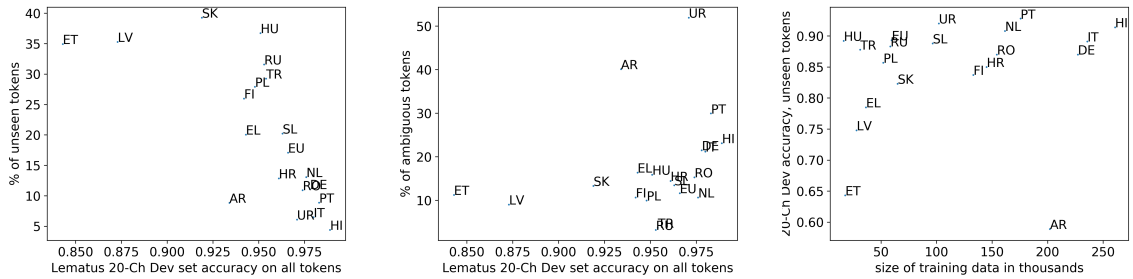


Figure 5: Lemmatization accuracy of Lematus 20-Ch on *all* dev set tokens vs percent of unseen tokens (left) or percent of ambiguous tokens (middle); accuracy on *unseen* tokens vs training set size (right).

dev set SeenUA). Lemming does as well as Lematus 20-Ch on ambiguous and SeenUA words, but its accuracy on unseen words is lower ($p < 0.001$), leading to worse performance overall ($p < 0.01$ on both dev and test). Interestingly, even Lematus 0-Ch does better than Lemming on unseen words ($p < 0.02$), and performs on par overall ($p = 0.28$). So, although including context clearly can help (compare Lematus 20-Ch vs 0-Ch), and Lemming exploits this advantage for ambiguous words, a good context-free model can still do very well. Overall, our models do as well or better than the earlier ones, without the added model complexity and annotation requirements. On the other hand, although our context-sensitive model does improve somewhat over its context-free counterpart, there is still some way to go, since average performance on unseen and ambiguous words is still 84% and 92% respectively.

Results on 10k Datasets Figure 4d shows the results on all tokens for each language in the 10k training setting, with averages in Table 2. On average, limiting training data to the first 10k examples resulted in an 82% reduction of training sets, and we see an average drop in test set performance of 5.6-6.8 percentage points for all systems except Ch-2017, which drops by about 10 percent. When comparing the 0-Ch and 20-Ch versions of Lematus we found the same pattern of significances as in the full data setting ($p < 0.01$), however the two best systems (Lematus 20-Ch and Lemming) are statistically equivalent on the test sets, as are Lemming and Lematus 0-Ch.

Patterns Across Languages In Section 4, we hypothesized that the success of data-driven lemmatization depends on a language's productivity, ambiguity, and regularity. We now explore the extent to which our results support this hypothesis. First, we examine the correlation between the overall performance of our best system on each language and the percentage of unseen (Figure 5, left) or ambiguous words (Figure 5, middle) in that language. As expected, there is a strong negative correlation between the percentage of unseen words and the accuracy of Lematus 20-Ch: the rank correlation is $R = -0.73$ ($p < 0.001$; we use rank correlation because it is less sensitive to outliers than is linear correlation, and the plot clearly shows several outliers.) In contrast to our original prediction, however, Lematus 20-Ch is actually *more* accurate for languages with greater ambiguity ($R = 0.44$, $p = 0.05$). The most likely explanation is that ambiguity is negatively correlated with productivity. Since there tend to be more unseen than ambiguous words, and since accuracy is typically lower for unseen than ambiguous words, higher ambiguity (which implies fewer unseen words) can actually lead to higher overall accuracy.

Our earlier results also suggested that the main benefit of Lematus 20-Ch over Lematus 0-Ch is for ambiguous words. To confirm this, we looked at the extent to which the *difference* in performance between the two systems correlates with the percentage of unseen or ambiguous words in a language. As expected, this analysis suggests that including context in the model helps more for languages with more ambiguity ($R = 0.67, p < 0.001$). In contrast, Lematus 20-Ch provides *less* benefit over Lematus 0-Ch for the languages with more unseen words ($R = -0.75, p < 0.0001$). Again, we assume the latter result is due to the negative correlation between ambiguity and productivity.

So far, our results and analysis show a clear relationship between productivity and ambiguity, and also suggest that using context for lemmatization may be unnecessary (or at least less beneficial) for languages with many unseen words but low am-

biguity. However, there are remaining differences between languages that are more difficult to explain. For example, one might expect that for languages with more training data, the system would learn better generalizations and lemmatization accuracy on unseen words would be higher. However, Figure 5 (right), which plots accuracy on *unseen* words in each language as a function of training data size, illustrates that there is no significant correlation between the two variables ($R = 0.32$, $p = 0.16$). In some languages (e.g., Hungarian, in the top left) Lematus performs very well on unseen words even with little training data, while in others (e.g., Arabic, along the bottom) it performs poorly despite relatively large training data. We assume that regularity (and perhaps the nonconcatenative nature of Arabic) must be playing an important role here, but we leave for future work the question of how to operationalize and measure regularity in order to further test this hypothesis.

7 Conclusion

We presented Lematus, a simple sequence-to-sequence neural model for lemmatization that uses character-level context. On average across 20 languages, we showed that even without using context, this model performs as well or better than three previous systems that treated lemmatization as an edit tree classification problem and required POS tags (Chrupała et al., 2008; Müller et al., 2015) or word embeddings trained on a much larger corpus (Chakrabarty et al., 2017). We also showed that with both larger and smaller training datasets, including context boosts performance further by improving accuracy on both unseen and (especially) ambiguous words.

Finally, our analysis suggests that lemmatization accuracy tends to be higher for languages with *low* productivity (as measured by the proportion of unseen words at test time), but more surprisingly also for languages with *high* ambiguity—perhaps because high ambiguity is also associated with low productivity. We also found that the amount of training data available for each language is not a good predictor of performance on unseen words, suggesting that morphological regularity or other language-specific characteristics are playing an important role. Understanding the causes of these differences is likely to be important for further improving neural lemmatization.

8 Acknowledgements

This work was supported in part by the James S McDonnell Foundation (Scholar Award #220020374).

References

Roee Aharoni, Yoav Goldberg, and Yonatan Belinkov. 2016. Improving sequence to sequence learning for morphological inflection generation: The BIU-MIT systems for the SIGMORPHON 2016 shared task for morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. page 41.

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

Toms Bergmanis, Katharina Kann, Hinrich Schütze, and Sharon Goldwater. 2017. Training data augmentation for low-resource morphological inflection. In *Proceedings of the CoNLL-SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*. Association for Computational Linguistics, Vancouver, Canada.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5:135–146.

Abhisek Chakrabarty, Onkar Arun Pandit, and Utpal Garain. 2017. Context sensitive lemmatization using two successive bidirectional gated recurrent networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1481–1491.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *The Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Association for Computational Linguistics, Doha, Qatar.

Grzegorz Chrupała. 2006. Simple data-driven context-sensitive lemmatization. *Procesamiento del Lenguaje Natural* 37.

Grzegorz Chrupała, Georgiana Dinu, and Josef van Genabith. 2008. Learning morphology with morfette. In Bente Maegaard Joseph Mariani Jan Odijk Stelios Piperidis Daniel Tapias Nicoletta Calzolari (Conference Chair), Khalid Choukri, editor, *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*. European Language Resources Association (ELRA), Marrakech, Morocco.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2017. Conll-sigmorphon 2017 shared task: Universal morphological reinflection in 52 languages. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*. Association for Computational Linguistics, Vancouver, pages 1–30.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 shared taskmorphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. pages 10–22.

Tomaž Erjavec and Sasčo Džeroski. 2004. Machine learning of morphosyntactic structure: Lemmatizing unknown slovene words. *Applied Artificial Intelligence* 18(1):17–41.

Nerea Ezeiza, Iñaki Alegria, José María Arriola, Rubén Urizar, and Itziar Aduriz. 1998. Combining stochastic and rule-based methods for disambiguation in agglutinative languages. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, pages 380–384.

Philip Gage. 1994. A new algorithm for data compression. *C Users J.* 12(2):23–38.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*. pages 1019–1027.

Dilek Z. Hakkani-Tür, Kemal Oflazer, and Gökhan Tür. 2002. Statistical morphological disambiguation for agglutinative languages. *Computers and the Humanities* 36(4):381–410.

Sittichai Jiampojamarn, Colin Cherry, and Grzegorz Kondrak. 2008. Joint processing and discriminative training for letter-to-phoneme conversion. In *Proceedings of ACL-08: HLT*. Association for Computational Linguistics, Columbus, Ohio, pages 905–913.

Katharina Kann and Hinrich Schütze. 2016. MED: The LMU system for the SIGMORPHON 2016 shared task on morphological reinflection. In *Proceedings of ACL*. Association for Computational Linguistics.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1520–1530.

Thomas Müller, Ryan Cotterell, Alexander Fraser, and Hinrich Schütze. 2015. Joint lemmatization and morphological tagging with lemming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 2268–2274.

Joakim Nivre et al. 2017. Universal dependencies 2.0 CoNLL 2017 shared task development and test data. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University.

Kemal Oflazer and Ìlker Kuruöz. 1994. Tagging and morphological disambiguation of turkish text. In *Proceedings of the fourth conference on Applied natural language processing*. Association for Computational Linguistics, pages 144–149.

Robert Östling. 2016. Morphological reinflection with convolutional neural networks. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. pages 23–26.

Lutz Prechelt. 1998. Early stopping-but when? *Neural Networks: Tricks of the trade* pages 553–553.

Tim Salimans and Diederik P. Kingma. 2016. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *CoRR* abs/1602.07868.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. 2017. Nematus: a toolkit for neural machine translation. *CoRR* abs/1703.04357.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1715–1725.

Clara Vania and Adam Lopez. 2017. From characters to words to in between: Do we capture morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 2016–2027.

Matthew D Zeiler. 2012. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

Modeling Noisiness to Recognize Named Entities using Multitask Neural Networks on Social Media

Gustavo Aguilar[⋆] **A. Pastor López-Monroy**[⋆] **Fabio A. González**[†] **Thamar Solorio**[⋆]
[⋆]Department of Computer Science, University of Houston
[†]Systems and Computer Engineering Department, Universidad Nacional de Colombia
{gaguilaralas, alopezmonroy}@uh.edu
fagonzalezo@unal.edu.co, solorio@cs.uh.edu

Abstract

Recognizing named entities in a document is a key task in many NLP applications. Although current state-of-the-art approaches to this task reach a high performance on clean text (e.g. newswire genres), those algorithms dramatically degrade when they are moved to noisy environments such as social media domains. We present two systems that address the challenges of processing social media data using character-level phonetics and phonology, word embeddings, and Part-of-Speech tags as features. The first model is a multitask end-to-end Bidirectional Long Short-Term Memory (BLSTM)-Conditional Random Field (CRF) network whose output layer contains two CRF classifiers. The second model uses a multitask BLSTM network as feature extractor that transfers the learning to a CRF classifier for the final prediction. Our systems outperform the current F1 scores of the state of the art on the Workshop on Noisy User-generated Text 2017 dataset by 2.45% and 3.69%, establishing a more suitable approach for social media environments.

1 Introduction

One of the core tasks in Natural Language Processing (NLP) is Named Entity Recognition (NER). NER is a sequence tagging task that consists in selecting the words that describe entities and recognizing their types (e.g., a person, location, company, etc.). Figure 1 shows examples of sentences from different domains that contain named entities. Recognizing entities in running text is typically one of the first tasks in the pipeline of many NLP applications, including machine translation, summarization, sentiment analysis, and question answering.

Traditional machine learning systems have proven to be effective in formal text, where grammatical errors are minimal and writers stick to the rules of the written language (Florian et al., 2003a; Chieu and Ng, 2003a). However, those traditional systems dramatically fail on informal text, where improper grammatical structures, spelling inconsistencies, and slang vocabulary prevail (Ritter et al., 2011). For instance, Table 1 shows a snapshot of NER systems' performance during the last years, where the results drop from 96.49% to 41.86% on the F1 metric as we move from formal to informal text. Although the results are not directly comparable because they consider different conditions and challenges, they serve as strong evidence that the NER task in social media is far from being solved.

CoNLL 2003
[***Spanish***]$_{\text{MISC}}$ Farm Minister [***Loyola de Palacio***]$_{\text{PER}}$ had earlier accused [***Fischler***]$_{\text{PER}}$ at an [***EU***]$_{\text{ORG}}$ farm ministers ' meeting of causing unjustified alarm through " dangerous generalisation . "

WNUT 2017, Twitter domain
been listenin to [***trey***]$_{\text{PER}}$ allllll week ... can u luv someone u never met ?? bcuz i think im in luv yeeuuuuppp !!!

Figure 1: Examples from the CoNLL 2003 and the WNUT 2017 datasets. The noise from the WNUT dataset makes a clear difference from one text to the other, establishing new challenges to the current state-of-the-art systems on formal text. The words in bold are grouped to described the entities.

Recently, researchers have approached NER using different neural network architectures. For instance, Chiu and Nichols (2016) proposed a neural model using Convolutional Neural Networks (CNN) for characters and a bidirectional Long Short Term Memory (LSTM) for words. Their model learned from word embeddings, capitalization, and lexicon features. On a slightly different approach, Lample et al. (2016) used a BLSTM with a CRF at the output layer, re-

Proceedings of NAACL-HLT 2018, pages 1401–1412
New Orleans, Louisiana, June 1 - 6, 2018.

Organizer	Competition	Domain	F1	Classes
Grishman and Sundheim (1996a)	MUC-6	Newswire	96.49%	2
Tjong Kim Sang and De Meulder (2003)	CoNLL	Newswire	88.76%	4
Strauss et al. (2016)	WNUT	Twitter	52.41%	10
Derczynski et al. (2017)	WNUT	SM domains	41.86%	6

Table 1: Results on different NER shared tasks. The performance degrades as the systems are moved to social media (SM) environments. The last row considers multiple SM domains, such as Twitter, YouTube, Reddit, and StackExchange.

moving the dependencies on external resources. Moreover, Ma and Hovy (2016) proposed an end-to-end BLSTM-CNN-CRF network, whose loss function is based on the maximum log-likelihood estimation of the CRF. These architectures were benchmarked on the standard CoNLL 2003 dataset (Tjong Kim Sang and De Meulder, 2003). Although most of the work has focused on formal datasets, similar approaches have been evaluated on SM domains (Strauss et al., 2016; Derczynski et al., 2017). In the Workshop on Noisy User-generated Text (WNUT) 2016, Limsopatham and Collier (2016), the winners of the NER shared task, used a BLSTM-CRF model that induced features from an orthographic representation of the text. Later, in the WNUT 2017 shared task, the best performing system used a multitask network that transferred the learning to a CRF classifier for the final prediction (Aguilar et al., 2017).

In this work we focus on addressing the challenges of the NER task found in social media environments. We propose that what is traditionally categorized as noise (i.e., misspellings, inconsistent orthography, emerging abbreviations, and slang) should be modeled *as is* since it is an inherent characteristic of SM text. Specifically, the proposed models attempt to address i) misspellings using subword level representations, ii) grammatical mistakes with SM-oriented Part-of-Speech tags (Owoputi et al., 2013), iii) sound-driven text with phonetic and phonological features (Bharadwaj et al., 2016), and iv) the intrinsic skewness of NER datasets by applying class weights. It is worth noting that our models do not rely on capitalization or any external resources such as gazetteers. The reasons are that capitalization is arbitrarily used on SM environments, and gazetteers are expensive resources to develop for a scenario where novel entities constantly and rapidly emerge (Derczynski et al., 2017; Augenstein et al., 2017).

Based on our experiments, we have seen that a multitask variation of the proposed networks improves the results over a single-task network. Additionally, this multitask version, paired with phonetic and phonological features, outperforms previous state-of-the-art results on the WNUT 2017 dataset, and the same models obtain reasonable results with respect to the state of the art on the CoNLL 2003 dataset (Tjong Kim Sang and De Meulder, 2003).

The rest of the paper is organized as follows: §2 presents the proposed features, the formal description of the models, and the implementation details. §3 describes the datasets and their challenges. On §4, we show the evaluation process of our models and the results. We explain the performance of the models on §5. §6 describes related work and, finally, we draw conclusions on §7.

2 Methods

Our methods are based on two main strategies: i) a representation of the input text using complementary features that are more suitable to social media environments, and ii) a fusion of these features by using a multitask neural network model whose main goal is to learn how entities are contextualized with and without the entity type information.

2.1 Feature representation

Semantic features. Semantic features play a crucial role in our pipeline as they provide contextual information to the model. This information allows the model to infer the presence of entities as well as the entity types. We use the pretrained word embedding model provided by Godin et al. (2015). This model has been trained on 1 million tweets (roughly 1% of the tweets in a year) with the skipgram algorithm. We take advantage of this resource as it easily adapts to other SM environments besides Twitter (Aguilar et al., 2017).

Syntactic features. Syntactic features help the models deal with word disambiguation based on

Sentence	IPA
u hav to b KIDDDDING me	/ju hæv tə bi kɪdɪŋ mi/
you have to be kidding me	/ju hæv tə bi kɪdɪŋ mi/

Table 2: Examples of both noisy and normalized text. In both cases, the mappings to the International Phonetic Alphabet (IPA) are the same.

the grammatical role that the words play on a sentence. That is, a word that can be a verb or a noun in different scenarios may conflict with the interpretations of the models; however, by providing syntactical information the models can improve their decisions. We capture grammatical patterns using the Part-of-Speech (POS) tagger provided by Owoputi et al. (2013). This POS tagger has custom labels that are suitable to SM data (i.e., the tagger considers emojis, hashtags, URLs and others).

Phonetic and phonological features. We also consider the phonetic and phonological aspects of the data at the character level. In Table 2 we show an example of two phrases: the first sentence is taken from SM, and the second one is its normalized representation. Even though the spellings of both phrases are significantly different, by using the phonological (articulatory) aspects of those phrases it is possible to map them to the same phonetic representation. In other words, our assumption is that social media writers heavily rely on the way that words sound while they write. We use the Epitran[1] library (Bharadwaj et al., 2016), which transliterates graphemes to phonemes with the International Phonetic Alphabet (IPA). In addition to the IPA phonemes, we also use the phonological (articulatory) features generated by the PanPhon[2] library (Mortensen et al., 2016). These features provide articulatory information such as the way the mouth and nasal areas are involved in the elaboration of sounds while people speak.

2.2 Models

We have experimented with two models. In the first one, we use an end-to-end BLSTM-CRF network with a multitask output layer comprised of one CRF per task, similar to Yang et al. (2016). In the second one, we define a stacked model that is based on two phases: i) a multitask neural network and ii) a CRF classifier. In the first phase, the network acts as a feature extractor, and then, for the second phase, it transfers the learning to a CRF classifier for the final predictions (see Figure 3). In both cases, the multitask layer is defined with the following two tasks:

- **Segmentation**. This task focuses on the Begin-Inside-Outside (BIO scheme) level of the tokens. That is, for a given NE, the model has to predict whether a word is B, I, or O regardless of the entity type. The idea is to let the models learn how entities are treated in general, rather than associating the types to certain contexts. This task acts as a regularizer of the primary task to prevent overfitting.

- **Categorization**. In this case, the models have to predict the types of the entities along with the BIO scheme (e.g., B-person, I-person, etc.), which represent the final labels.

We formalize the definitions of our models as follows: let $X = [x_1, x_2, ..., x_n]$ be a sample sentence where x_i is the i^{th} word in the sequence. Then, let $\alpha : V_x \rightarrow \mathbb{R}^{dim_x}$ be a word embedding, and let $\mathbf{x} = [\alpha(x_1), \ldots, \alpha(x_n)]$ be the word embedding matrix for the sample sentence such that V_x is the vocabulary and dim_x is the dimension of the embedding space. Similarly, let $\beta : V_p \rightarrow \mathbb{R}^{dim_p}$ be the POS tag embedding, and let $\mathbf{p} = [\beta(p_1), \ldots, \beta(p_n)]$ be the POS tag embedding matrix for the sample sentence such that V_p is the set of Part-of-Speech tags and dim_p is the dimension of the embedding space. Notice that the POS tag embedding matrix $\mathbf{p}$ is learned during training. Also, let $Q = [q_1, q_2, ..., q_m]$ be the phonetic letters of a word; let $\gamma : V_q \rightarrow \mathbb{R}^{|V_q|+dim_{PanPhon}}$ be an embedding that maps each phonetic character to a one-hot vector of the International Phonetic Alphabet (V_q) concatenated with the 21 ($dim_{PanPhon}$) phonological features of the PanPhon library (tongue position, movement of lips, etc.) (Bharadwaj et al., 2016); and let $\mathbf{q} = [\gamma(q_1), ..., \gamma(q_m)]$ be the matrix representation of the word-level phonetics and phonology.

We first apply an LSTM (Hochreiter and Schmidhuber, 1997) to the $\mathbf{q}$ matrix on forward and backward directions. Then we concatenate the output from both directions:

$$\overrightarrow{\mathbf{h}} = \text{LSTM}(\{\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_m\})$$
$$\overleftarrow{\mathbf{h}} = \text{LSTM}(\{\mathbf{q}_m, \mathbf{q}_{m-1}, ..., \mathbf{q}_1\})$$
$$\mathbf{h} = [\overrightarrow{\mathbf{h}}; \overleftarrow{\mathbf{h}}]$$

[1] https://github.com/dmort27/epitran
[2] https://github.com/dmort27/panphon

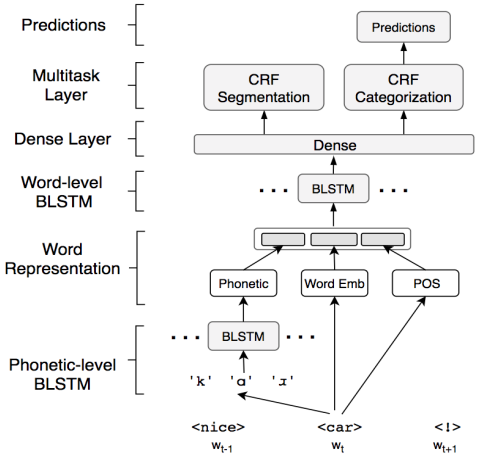


Figure 2: This is an end-to-end system that uses the CRF loss function as the objective function of the network. It also uses multitask learning on the output layer.

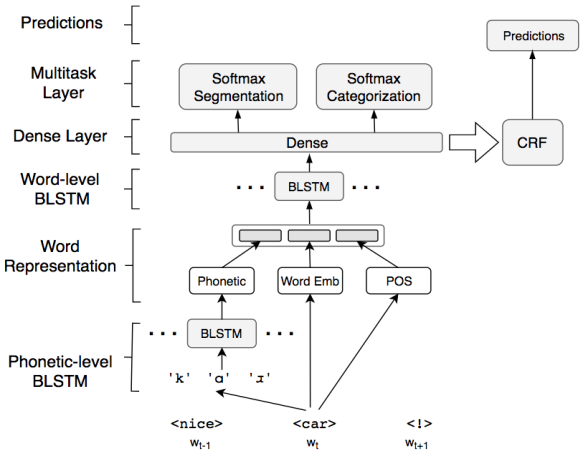


Figure 3: This is a stacked model that uses a network as feature extractor, and then it transfers the learning to a CRF classifier. The network uses multitask learning to capture the features.

This vector not only encodes the phonetic and phonological features, but it also captures some morphological patterns at the character level based on the IPA representations. Then, we concatenate this vector with the word and POS tag representations: $\mathbf{a} = [\mathbf{x}_t; \mathbf{p}_t; \mathbf{h}_t]$. We feed this representation to another bidirectional LSTM network (Dyer et al., 2015), similar to the BLSTM described for the character level. The bidirectional LSTM generates a word-level representation that accounts for the context in the sentence using semantics, syntax, phonetics and phonological aspects. We feed this representation to a fully-connected layer:

$$\mathbf{r}_i = \text{BLSTM}(\{\mathbf{a}_1, \mathbf{a}_2, ...\mathbf{a}_n\}) \quad (1)$$

$$\mathbf{z}_i = \text{ReLU}(\mathbf{W}_a \mathbf{r}_i + \mathbf{b}) \quad (2)$$

At this point, both models share the same definition. From here, we describe the multitask learning characteristics for each model separately.

End-to-end model. For the end-to-end network (see Figure 2), we define an output layer based on two Conditional Random Fields (Lafferty et al., 2001), each assigned to one of the tasks. The idea of adding a CRF to the model is to capture the relation of the output probabilities of the network with respect to the whole sequence. This means that the CRFs will maximize the log-likelihood of the entire sequence, which allows the model to learn very specific constraints from the data (e.g., a label *I-location* cannot be followed by *I-person*). Following Ma and Hovy (2016), we formalize the definition of the CRF as follows: let $\mathbf{y} = [y_1, y_2, ..., y_n]$ be the labels for a sequence $\mathbf{x}$, where y_i represents the i^{th} label of the $\mathbf{x}_i$ token in the sentence. Next, we calculate the conditional probability of seeing $\mathbf{y}$ given the extracted features $\mathbf{z}$ from the network and the weights $\mathbf{W}$ associated to the labels:

$$p(\mathbf{y}|\mathbf{z}; \mathbf{W}) = \frac{\exp(\mathbf{W}_y \Phi(\mathbf{z}, \mathbf{y}))}{\sum_{\mathbf{y}' \in \mathbf{y}} \exp(\mathbf{W}_y \Phi(\mathbf{z}, \mathbf{y}'))}$$

Where Φ is a feature function that codifies the interactions between consecutive labels, y_t and y_{t+1}, as well as the interactions between labels and words, represented by z_t. Then, the objective function for one CRF is defined by the maximum log-likelihood of this probability. However, we are running two CRFs as the objective function:

$$\mathcal{L}_1(\mathbf{z}, \mathbf{W}) = \log p(\mathbf{y}_{seg}|\mathbf{z}; \mathbf{W})$$
$$\mathcal{L}_2(\mathbf{z}, \mathbf{W}) = \log p(\mathbf{y}_{cat}|\mathbf{z}; \mathbf{W})$$
$$L(\mathbf{z}, \mathbf{W}) = \alpha \mathcal{L}_1(\mathbf{z}, \mathbf{W}) + \mathcal{L}_2(\mathbf{z}, \mathbf{W})$$

Where $\mathcal{L}_1$ is the loss function of the segmentation task with labels $\mathbf{y}_{seg}$. Similarly, $\mathcal{L}_2$ is the loss function of the categorization task with labels $\mathbf{y}_{cat}$. L is the loss function that accounts for both tasks, where the segmentation task is weighted by an α scalar.

Stacked model. For this model, we use a multitask network as a feature extractor whose loss function is defined as a categorical cross entropy (see Figure 3). We apply a softmax activation function to produce the probability distribution over the labels, and then we calculate the loss as

follows:

$$H_1(\mathbf{y}, \mathbf{z}) = -\sum_{\mathbf{z}_i} \mathbf{y} \log(softmax(\mathbf{W}_{seg}\mathbf{z}_i + \mathbf{b}))$$

$$H_2(\mathbf{y}, \mathbf{z}) = -\sum_{\mathbf{z}_i} \mathbf{y} \log(softmax(\mathbf{W}_{cat}\mathbf{z}_i + \mathbf{b}))$$

$$L(\mathbf{y}, \mathbf{z}) = \alpha H_1(\mathbf{z}, \mathbf{W}_{seg}) + H_2(\mathbf{z}, \mathbf{W}_{cat})$$

After training the multitask network, we take the activation outputs from Equation 2. These vectors are used as features to train a Conditional Random Fields classifier. The definition of the CRF is the same as the one described for the end-to-end network.

2.3 Implementation details

We have performed a very simple preprocessing on the data, which consists in replacing URLs, emojis, tags, and numbers with predefined tokens. Additionally, the vocabulary of the pretrained word embeddings was not sufficient to cover all the words in the WNUT dataset (i.e., training, validation, and testing sets have OOV words). We handled this situation using the Facebook library FastText (Bojanowski et al., 2016). This library can produce an embedding vector from the subword level of the word (i.e., ngrams). The advantage of FastText over other embedding learning algorithms is that we can still extract useful embeddings for OOV words from their subword embeddings. For instance, if there is a missing letter in one word, the subword-level vector will be reasonably close to the vector of the correct spelling.

The models have been trained using weighted classes, which forces the models to pay more attention to the labels that are less frequent. This is a very important step since the NE datasets usually show a skewed distribution, where the NE tokens represent approximately 10% of the entire corpus. Although weighting classes improves the recall of the model, we tried to be sensitive to this aspect as the model can be forced to predict entities even in cases where there are none. The weights were experimentally defined, keeping the same distribution but decreasing the loss on non-entity tokens.

Additionally, we defined our models using the following hyperparameters: the phonetic and phonological BLSTM at the character level uses 64 units per direction, which adds up to 128 units. Similarly, the word level BLSTM uses 100 units per direction, which accounts for a total of 200 units. The fully-connected layer has 100 neurons, and it uses a Rectified Linear Unit (ReLU) activation function. We also use a dropout operation before and after each BLSTM component. This forces the networks to find different paths to predict the data, which ultimately improves the generalization capabilities (i.e., they do not rely on a single path for certain inputs). The dropout value is 0.5. For the stacked model we use the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.001.

Corpus	Dataset	Classes	% Unique
CoNLL 2003	Train	4	26%
	Dev	4	40%
	Test	4	41%
WNUT 2017	Train	6	75%
	Dev	6	85%
	Test	6	80%

Table 3: Percentage of unique NEs in two benchmark datasets, the one from CoNLL 2003 and the one used in the 2017 shared task held by the WNUT workshop.

3 Datasets

Social media (SM) captures the fast evolving behavior of the language, and, as its influence in society grows, SM platforms play an important role in language understanding. We focus this work on the WNUT 2017 dataset for NER (Derczynski et al., 2017). This dataset covers multiple SM platforms and suits perfectly the purpose of this work. Table 5 shows the distribution of the dataset and its classes. The training set uses tweets, whereas the development set is based on YouTube comments. The testing set combines content from Reddit and StackExchange. The cross domain nature of the dataset establishes an additional challenge to the task. For instance, besides the particularities of the domains (e.g., length of the sentences, domain-specific expressions such as hashtags, emojis and others), the users tend to address different topics on each of the SM domains with different levels of relaxed language and style (Ritter et al., 2011; Strauss et al., 2016; Derczynski et al., 2017). Moreover, the predominant factors in those SM environments are the emerging and rare entities. As stated by Derczynski et al. (2017), *emerging* describes the entity instances that started to appear in context recently (e.g., a movie title released a

Statistics	Train	Dev	Test
Posts	3,395	1,009	1,287
Tokens	62,729	15,733	23,394
NE tokens	3,160	1,250	1,589
NE tokens (%)	5.04	7.95	6.79

Table 4: General statistics of the WNUT 2017 dataset. It is worth noting that the NE tokens account for less than 10% on any dataset, which shows the inherent skewness of the task.

Classes	Train	Dev	Test
person	995	46	532
location	793	238	188
group	414	64	202
creative-work	346	107	331
product	345	586	250
corporation	267	209	86
TOTAL	3,160	1,250	1,589

Table 5: Classes and their frequency distribution on the WNUT 2017 dataset.

year ago), whereas *rare* depicts the entities that appear less than certain number of times. It is worth noting that this dataset presents a great challenge to systems that rely on external resources due to the rare and emerging properties.

We also consider the CoNLL 2003 dataset (Tjong Kim Sang and De Meulder, 2003) as it has been used as the standard dataset for NER benchmarks. However, we emphasize that both datasets present significantly different challenges and, thus, some relevant aspects in CoNLL 2003 may not be that relevant in the WNUT 2017 dataset. For example, capitalization is a crucial feature in newswire text, but it is less important in SM data since users tend to arbitrarily alter the character casing. Moreover, the target classes on the WNUT 2017 dataset cover the CoNLL 2003 classes plus fine-grained classes such as *creative-work* (e.g., movie titles, T.V. shows, etc.), *group* (e.g., sports teams, music bands, etc.), and *product*. The additional classes are more heterogeneous, and thus, it makes the task more difficult to generalize. Furthermore, Table 3 shows the percentage of unique tokens of the WNUT 2017 dataset, which certainly shows a great diversity compared to the CoNLL 2003 dataset.

4 Experiments and results

We mainly focus our experiments on the WNUT 2017 dataset. However, we consider relevant to compare our approach to the standard CoNLL 2003 dataset where current state-of-the-art systems are benchmarked. This section addresses the experiments and results of both datasets.

4.1 WNUT 2017 experiments

In this section we discuss the experiments of the proposed approaches. We compare our models and describe the contribution of each component of the stacked system. Additionally, we compare our results against the state of the art in the WNUT 2017 dataset.

Stacked vs. end-to-end model. Table 6 shows that the stacked system has a lower precision than the end-to-end model, but its recall is the highest. This means that the stacked model is slightly better at generalizing than the other models since it can detect a more diverse set of entities. The surface form F1 metric (Derczynski et al., 2017) supports that intuition as well. It assigns a better F1 score to the stacked system (43.90%) than to the end-to-end model (42.79%) because the former finds more *rare* and *emerging* entities than the latter. Moreover, Table 6 also shows that the precision of the end-to-end model is higher than the rest of the systems. This tends to capture the most frequent entities and leave behind the *rare* ones, which explains the different behaviors between the precision and recall of both models.

Stacked model. The feature extractor contains a category task that can produce predictions of the test set. We explored predicting the final labels with the feature extractor and compared the results against the predictions of the CRF classifier. We noticed that the CRF always outperformed the network. For the best scores the feature extractor achieved 40.64% whereas the CRF reached 45.55%. This is consistent with previous research (Lample et al., 2016; Aguilar et al., 2017) in that the individual output probabilities of the network do not consider the whole sequence, and thus, a sequential algorithm such as a CRF can improve the results by learning global constraints (i.e., the *B-person* cannot be followed by *I-corporation*).

Ablation experiment. We explored the contribution of the features and different aspects of our models. For instance, we tried a BLSTM network using pretrained word embeddings only. The re-

Classes	Precision (%)			Recall (%)			F1 (%)		
	Stacked	E2E	WNUT	Stacked	E2E	WNUT	Stacked	E2E	WNUT
corporation	33.33	30.77	31.91	19.70	12.12	22.73	24.76	17.39	26.55
creative-work	50.00	55.56	36.67	14.79	10.56	7.75	22.83	17.75	12.79
group	47.76	63.16	41.79	19.39	14.55	16.97	27.59	23.65	24.14
location	62.20	78.12	56.92	52.67	50.00	49.33	57.04	60.98	52.86
person	73.49	71.15	70.72	51.05	51.75	50.12	60.25	59.92	58.66
product	40.58	34.29	30.77	22.05	9.45	9.45	28.57	14.81	14.46
Overall	61.06	**66.67**	57.54	**36.33**	32.99	32.90	**45.55**	44.14	41.86

Table 6: The class-level and overall results of our systems on the WNUT 2017 dataset. WNUT represents the winning system of the shared task (UH-RiTUAL), E2E is the end-to-end model, and Stacked shows the results of the stacked model. Both systems considerably outperform the state-of-the-art results. Between the end-to-end and the stacked models, the former gets better overall precision while the latter stands out on recall.

Model	F1	Delta
Stacked Model	**45.55**	
- Multitask Learning	44.76	-0.79
- Character phonetics	43.83	-0.93
- Weighted classes	41.25	-2.58
- POS tag vectors	40.15	-1.10
- FastText OOV vectors	39.78	-0.37
- Pretrained embeddings	12.72	-27.06

Table 7: We performed an ablation experiment on the stacked model. The results in the table are the average of the scores of three iterations.

sults of this model set our baseline on a 39.78% F1-score (see Table 7). This score is considerably close to the state-of-the-art performance, but improvements beyond that are small. For instance, Table 7 shows an ablation experiment using the stacked model. The ablation reveals that weighting the classes is the most influential factor, which accounts for a 2.58% of F1 score improvement. This aligns with the fact that the data is highly skewed, and thus, the model should pay more attention to the less frequent classes. The second most important aspect is the POS tags, which enhance the results by 1.10%. This improvement suggests that POS tags are important whether the dataset is from a noisy environment or not since other researchers have found positive effects by using this feature on formal text (Huang et al., 2015). Almost equally influential are the phonetic and phonological features that push the F1 score by 0.93%. According to the ablation experiment, using phonetic and phonology along with the pretrained word embeddings and POS tags can reach an F1 measure of 41.81%, which is a very similar result to the state-of-the-art score, but with a simpler and more suitable model for SM environments (i.e., without gazetteers or capitalization).

We explored the multitask learning aspect by empirically trying multiple combinations of auxiliary tasks. The best combination is the standard NER categorization along with the segmentation task. The segmentation slightly improves the binary task proposed by Aguilar et al. (2017) by around 0.3%. Additionally, trying the binarization, segmentation, and categorization tasks together drops the results by around 0.2% with respect to the categorization paired with the binary task. Moreover, the ablation experiment shows that the multitask layer boosts the performance of the stacked model with 0.79% of F1 score.

For the OOV problem, we use FastText to provide vectors to 2,333 words (around 13% of the vocabulary). However, the ablation experiment shows a small improvement, which suggests that those words did not substantially contribute to the meaning of the context. Another aspect that we explored was adding all the letters of the dataset to the character level of the stacked model without modifying the casing. Surprisingly, the models produced a slightly worse result (around -0.5%). Our intuition is that the character aspects are already captured by the model with the phonetic (IPA) representation, and the arbitrary use of capitalization renders this information useless. It is also worth noting that having phonetics instead of a language-dependent alphabet allows the adaptability of this approach to other languages.

State of the art comparison. Table 6 shows that our end-to-end and stacked models significantly outperform the state-of-the-art score by 2.28% and 3.69% F1 points, respectively. In the case of the stacked system, the precision and recall outper-

No	**Predictions**
1	Road and airport closure isolate **Srinagar** as avalanche risk remains high
2	The **Defence Research Development Organisation** (**DRDO**) is working on four projects to develop new technologies for more accurate ...
3	Her name is **Scout** .

Table 8: Examples of the predictions of our stacked model in the Reddit domain of the WNUT 2017 dataset. The bold words are the gold labels, and the underlined words are the predictions of our model. The model matches the entity types of the labeled data.

form the winning system of the shared task (UH-RiTUAL) across all the classes. Moreover, even though the UH-RiTUAL system uses gazetteers, it only outperforms the recall of the end-to-end model on the *corporation* class. These results can be explained by the entity diversity of the dataset, where the *emerging* and *rare* properties are difficult to capture with external resources.

4.2 CoNLL 2003 evaluation

We also benchmarked our approach on a standard CoNLL 2003 dataset for the NER task. The stacked model reached 89.01% while the end-to-end model achieved 88.98% on the F1 metric. Although the state-of-the-art performance is 91.21% (Ma and Hovy, 2016), our approach targets SM domains and, consequently, our models disregard some of the important aspects on formal text while still getting reasonable results. For instance, Ma and Hovy (2016) input the text to their model *as is*, which indirectly introduce capitalization to the morphological analysis at the character level. This aspect becomes relevant in this dataset because entities are usually capitalized on formal text. As explained before, our models do not rely on capitalization because the characters are represented by the International Phonetic Alphabet, which does not differentiate between lower and upper cases.

5 Analysis

Table 8 shows some predictions of our stacked model on the WNUT 2017 test set. In example number 1, the model is able to correctly label *Srinagar* as *person*, even though the model does not rely on gazetteers or capitalization. It is also important to mention that the word was not in the training or development set, which means that the network had to infer the entity purely from the context. Moreover, the second example shows that the model has problems to determine whether the article *the* belongs to an NE or not. This is an ambiguous problem that even humans struggle with. This example also has a variation on spelling for the words *Defence* and *Organisation*. We suspect that the mitigation of OOV words using the FastText library helped in this case. Also, from the phonetic perspective, the model treated the word *Defence* as if it was the word *Defense* because both words map to the same IPA sequence, /dɪfɛns/. In the third case, the model is not able to identify the NE *Scout*, even though the context makes it fairly easy.

6 Related work

In its former years, NER systems focused on newswire text, where the goal was to identify mainly three types of entities: *person*, *corporation*, and *location*. These entity types were originally proposed in the 6th Message Understanding Conference (MUC-6) (Grishman and Sundheim, 1996b). In MUC-7, the majority of the systems were based on heavily hand-crafted features and manually elaborated rules (Borthwick et al., 1998). Some years later, many researchers incorporated machine learning algorithms to their systems, but there was still a strong dependency on external resources and domain-specific features and rules (Tjong Kim Sang and De Meulder, 2003). In addition, the majority of the systems used Maximum Entropy (Bender et al., 2003; Chieu and Ng, 2003b; Curran and Clark, 2003; Florian et al., 2003b; Klein et al., 2003) and Hidden Markov Models (Florian et al., 2003b; Klein et al., 2003; Mayfield et al., 2003; Whitelaw and Patrick, 2003). Furthermore, McCallum and Li (2003) used a CRF combined with web-augmented lexicons. The features were selected by hand-crafted rules and refined based on their relevance to the domain of the entities. Moreover, Nothman et al. (2013) used Wikipedia resources to take advantage of structured data and reduce the human-annotated labels. In general, the results of the systems were reasonable for formal text, yet the scalability and the expensive detailed rules were not; their systems were difficult to maintain and adapt to other domains where different rules were needed.

Recently, NER has been focused on noisy data as a result of the growth in social media users. However, the limits of the previous systems dramatically affected the results on noisy domains. For instance, Derczynski et al. (2014) evaluated multiple NER tools in noisy environments: Stanford NER (Finkel et al., 2005), ANNIE (Cunningham et al., 2002), among others. They reported that the majority of the tools were not capable of adapting to the noisy conditions showing a drop in performance of around 40% on a F1-score metric. This motivated many researchers to solve the problem using different techniques. In 2015, Baldwin et al. (2015) organized a NER shared task at the 1st Workshop on Noisy User-generated Text (WNUT), where three of the participants used word embedding as features to train their traditional machine learning algorithms (Godin et al., 2015; Toh et al., 2015; Cherry et al., 2015). The shared task introduced noisy data as well as more difficult entity types to identify (e.g., tv show, product, sports team, movie, music artist, etc.). Notably, the WNUT 2016 and 2017 were predominated by neural network systems (Limsopatham and Collier, 2016; Aguilar et al., 2017).

Deep neural networks have proven to be effective for NER. The state-of-the-art and the most competitive architectures can be characterized by the use of recurrent neural networks (Chiu and Nichols, 2016) combined with CRF (Lample et al., 2016; Ma and Hovy, 2016; Peng and Dredze, 2016; Bharadwaj et al., 2016; Aguilar et al., 2017). Our work primarily focuses on social media data and explores more suitable variations and combinations of those models. The most important differences of our approach and previous works are i) the use of phonetics and phonology (articulatory) features at the character level to model SM noise, ii) consistent BLSTMs for character and word levels, iii) the segmentation and categorization tasks, iv) a multitask neural network that transfers the learning without using lexicons or gazetteers, and v) weighted classes to handle the inherent skewness of the datasets.

7 Conclusions

This paper proposed two models for NER on social media environments. The first one is a stacked model that uses a multitask BLSTM network as a feature extractor to transfer the learning to a CRF classifier. The second one is an end-to-end multitask BLSTM-CRF model whose output layer has a CRF per task. Both models improve the state-of-the-art results on the WNUT 2017 dataset, where the data comes from multiple SM domains (i.e., Twitter, YouTube, Reddit, and StackExchange). Instead of working on normalizing text, we designed representations that are robust to inherent properties of SM data: inconsistent spellings, diverse vocabulary, and flexible grammar. Considering that SM is a prevalent communication channel that constantly generates massive amounts of data, it is practical to design NLP tools to process this domain *as is*. In this sense, we showed that the phonetic and phonological features are useful to capture sound-driven writing. This approach avoids the standard normalization process and boosts prediction performance. Furthermore, the use of multitask learning with segmentation and categorization is important to improve the results of the models. Finally, the weighted classes force the model to pay more attention on skewed datasets. We showed that these components can point to more suitable approaches for NER on social media data.

References

Gustavo Aguilar, Suraj Maharjan, Adrian Pastor López Monroy, and Thamar Solorio. 2017. A multi-task approach for named entity recognition in social media data. In *Proceedings of the EMNLP 3rd Workshop on Noisy User-generated Text*. Association for Computational Linguistics, Copenhagen, Denmark, pages 148–153. http://www.aclweb.org/anthology/W17-4419.

Isabelle Augenstein, Leon Derczynski, and Kalina Bontcheva. 2017. Generalisation in named entity recognition: A quantitative analysis. *CoRR* abs/1701.02877. http://arxiv.org/abs/1701.02877.

Timothy Baldwin, Marie-Catherine de Marneffe, Bo Han, Young-Bum Kim, Alan Ritter, and Wei Xu. 2015. Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In *Proceedings of the Workshop on Noisy User-generated Text*. Association for Computational Linguistics, Beijing, China, pages 126–135. http://www.aclweb.org/anthology/W15-4319.

Oliver Bender, Franz Josef Och, and Hermann Ney. 2003. Maximum entropy models for named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL*

2003. pages 148–151. http://www.aclweb.org/anthology/W03-0420.pdf.

Akash Bharadwaj, David Mortensen, Chris Dyer, and Jaime Carbonell. 2016. Phonologically aware neural model for named entity recognition in low resource transfer settings. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1462–1472. https://aclweb.org/anthology/D16-1153.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606* .

A. Borthwick, J. Sterling, E. Agichtein, and R. Grishman. 1998. Nyu: Description of the mene named entity system as used in muc-7. In *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29 - May 1, 1998*. http://www.aclweb.org/anthology/M98-1018.

Colin Cherry, Hongyu Guo, and Chengbi Dai. 2015. Nrc: Infused phrase vectors for named entity recognition in twitter. In *Proceedings of the Workshop on Noisy User-generated Text*. Association for Computational Linguistics, Beijing, China, pages 54–60. http://www.aclweb.org/anthology/W15-4307.

Hai Leong Chieu and Hwee Tou Ng. 2003a. Named entity recognition with a maximum entropy approach. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. Association for Computational Linguistics, Stroudsburg, PA, USA, CONLL '03, pages 160–163. https://doi.org/10.3115/1119176.1119199.

Hai Leong Chieu and Hwee Tou Ng. 2003b. Named entity recognition with a maximum entropy approach. In Walter Daelemans and Miles Osborne, editors, *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*. pages 160–163. http://www.aclweb.org/anthology/W03-0423.pdf.

Jason Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics* 4:357–370. https://transacl.org/ojs/index.php/tacl/article/view/792.

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. 2002. Gate: an architecture for development of robust hlt applications. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, pages 168–175. https://doi.org/10.3115/1073083.1073112.

James R. Curran and Stephen Clark. 2003. Language independent ner using a maximum entropy tagger. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. Association for Computational Linguistics, Stroudsburg, PA, USA, CONLL '03, pages 164–167. https://doi.org/10.3115/1119176.1119200.

Leon Derczynski, Diana Maynard, Giuseppe Rizzo, Marieke van Erp, Genevieve Gorrell, Raphaël Troncy, Johann Petrak, and Kalina Bontcheva. 2014. Analysis of named entity recognition and linking for tweets. *CoRR* abs/1410.7182. http://arxiv.org/abs/1410.7182.

Leon Derczynski, Eric Nichols, Marieke van Erp, and Nut Limsopatham. 2017. Results of the WNUT2017 Shared Task on Novel and Emerging Entity Recognition. In *Proceedings of the 3rd Workshop on Noisy, User-generated Text (W-NUT) at EMNLP*. ACL. http://aclweb.org/anthology/W/W17/W17-4418.pdf.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependeny parsing with stack long short-term memory. In *Proceedings of the* 53^{rd} *Annual Meeting of the Association of Computational Linguistics and the* 7^{th} *International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP 2015)*. ACL.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '05, pages 363–370. https://doi.org/10.3115/1219840.1219885.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003a. Named entity recognition through classifier combination. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. Association for Computational Linguistics, Stroudsburg, PA, USA, CONLL '03, pages 168–171. https://doi.org/10.3115/1119176.1119201.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003b. Named entity recognition through classifier combination. In Walter Daelemans and Miles Osborne, editors, *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*. pages 168–171. http://www.aclweb.org/anthology/W03-0425.pdf.

Fréderic Godin, Baptist Vandersmissen, Wesley De Neve, and Rik Van de Walle. 2015. Multimedia Lab @ ACL WNUT NER Shared Task: Named Entity Recognition for Twitter Microposts using Distributed Word Representations. In *Proceedings of*

the Workshop on Noisy User-generated Text. Association for Computational Linguistics, Beijing, China, pages 146–153. http://www.aclweb.org/anthology/W15-4322.

Ralph Grishman and Beth Sundheim. 1996a. Message understanding conference - 6: A brief history. In *COLING-1996 Vol. 1*. pages 466–471.

Ralph Grishman and Beth Sundheim. 1996b. Message understanding conference-6: A brief history. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, COLING '96, pages 466–471. https://doi.org/10.3115/992628.992709.

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

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

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

Dan Klein, Joseph Smarr, Huy Nguyen, and Christopher D. Manning. 2003. Named entity recognition with character-level models. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. Association for Computational Linguistics, Stroudsburg, PA, USA, CONLL '03, pages 180–183. https://doi.org/10.3115/1119176.1119204.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, ICML '01, pages 282–289. http://dl.acm.org/citation.cfm?id=645530.655813.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *CoRR* abs/1603.01360. http://arxiv.org/abs/1603.01360.

Nut Limsopatham and Nigel Collier. 2016. Bidirectional lstm for named entity recognition in twitter messages. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 145–152. http://aclweb.org/anthology/W16-3920.

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

James Mayfield, Paul McNamee, and Christine Piatko. 2003. Named entity recognition using hundreds of thousands of features. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. Association for Computational Linguistics, Stroudsburg, PA, USA, CONLL '03, pages 184–187. https://doi.org/10.3115/1119176.1119205.

Andrew McCallum and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. Association for Computational Linguistics, Stroudsburg, PA, USA, CONLL '03, pages 188–191. https://doi.org/10.3115/1119176.1119206.

David R. Mortensen, Patrick Littell, Akash Bharadwaj, Kartik Goyal, Chris Dyer, and Lori Levin. 2016. Panphon: A resource for mapping ipa segments to articulatory feature vectors. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 3475–3484. http://aclweb.org/anthology/C16-1328.

Joel Nothman, Nicky Ringland, Will Radford, Tara Murphy, and James R. Curran. 2013. Learning multilingual named entity recognition from wikipedia. *Artif. Intell.* 194:151–175. https://doi.org/10.1016/j.artint.2012.03.006.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved Part-of-Speech Tagging for Online Conversational Text with Word Clusters. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Atlanta, Georgia, pages 380–390. http://www.aclweb.org/anthology/N13-1039.

Nanyun Peng and Mark Dredze. 2016. Improving named entity recognition for chinese social media with word segmentation representation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 149–155. http://anthology.aclweb.org/P16-2025.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: An experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP '11, pages 1524–1534. http://dl.acm.org/citation.cfm?id=2145432.2145595.

Benjamin Strauss, Bethany Toma, Alan Ritter, Marie-Catherine de Marneffe, and Wei Xu. 2016. Results of the wnut16 named entity recognition shared task. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 138–144. http://aclweb.org/anthology/W16-3919.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*. pages 142–147. http://www.aclweb.org/anthology/W03-0419.pdf.

Zhiqiang Toh, Bin Chen, and Jian Su. 2015. Improving twitter named entity recognition using word representations. In *Proceedings of the Workshop on Noisy User-generated Text*. Association for Computational Linguistics, Beijing, China, pages 141–145. http://www.aclweb.org/anthology/W15-4321.

Casey Whitelaw and Jon Patrick. 2003. Named entity recognition using a character-based probabilistic approach. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. Association for Computational Linguistics, Stroudsburg, PA, USA, CONLL '03, pages 196–199. https://doi.org/10.3115/1119176.1119208.

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

Reusing Weights in Subword-aware Neural Language Models

Zhenisbek Assylbekov
School of Science and Technology
Nazarbayev University
zhassylbekov@nu.edu.kz

Rustem Takhanov
School of Science and Technology
Nazarbayev University
rustem.takhanov@nu.edu.kz

Abstract

We propose several ways of reusing subword embeddings and other weights in subword-aware neural language models. The proposed techniques do not benefit a competitive character-aware model, but some of them improve the performance of syllable- and morpheme-aware models while showing significant reductions in model sizes. We discover a simple hands-on principle: in a multi-layer input embedding model, layers should be tied consecutively bottom-up if reused at output. Our best morpheme-aware model with properly reused weights beats the competitive word-level model by a large margin across multiple languages and has 20%–87% fewer parameters.

1 Introduction

A statistical language model (LM) is a model which assigns a probability to a sequence of words. It is used in speech recognition, machine translation, part-of-speech tagging, information retrieval and other applications. Data sparsity is a major problem in building traditional n-gram language models, which assume that the probability of a word only depends on the previous n words. To deal with potentially severe problems when confronted with any n-grams that have not explicitly been seen before, some form of smoothing is necessary.

Recent progress in statistical language modeling is connected with neural language models (NLM), which tackle the data sparsity problem by representing words as vectors. Typically this is done twice: at input (to embed the current word of a sequence into a vector space) and at output (to embed candidates for the next word of a sequence). Especially successful are the models in which the architecture of the neural network between input and output is recurrent (Mikolov et al., 2010), which we refer to as recurrent neural network language models (RNNLM).

Tying input and output word embeddings in word-level RNNLM is a regularization technique, which was introduced earlier (Bengio et al., 2001; Mnih and Hinton, 2007) but has been widely used relatively recently, and there is empirical evidence (Press and Wolf, 2017) as well as theoretical justification (Inan et al., 2017) that such a simple trick improves language modeling quality while decreasing the total number of trainable parameters almost two-fold, since most of the parameters are due to embedding matrices. Unfortunately, this regularization technique is not directly applicable to subword-aware neural language models as they receive subwords at input and return words at output. This raises the following questions: Is it possible to reuse embeddings and other parameters in subword-aware neural language models? Would it benefit language modeling quality? We experimented with different subword units, embedding models, and ways of reusing parameters, and our answer to both questions is as follows: There are several ways to reuse weights in subword-aware neural language models, and none of them improve a competitive character-aware model, but some of them do benefit syllable- and morpheme-aware models, while giving significant reductions in model sizes. A simple morpheme-aware model that sums morpheme embeddings of a word benefits most from appropriate weight tying, showing a significant gain over the competitive word-level baseline across different languages and data set sizes. Another contribution of this paper is the discovery of a hands-on principle that in a multi-layer input embedding model, layers should be tied consecutively bottom-up if reused at output.

The source code for the morpheme-aware model is available at `https://github.com/zh3nis/morph-sum`.

Proceedings of NAACL-HLT 2018, pages 1413–1423
New Orleans, Louisiana, June 1 - 6, 2018.

2 Related Work

Subword-aware NLM: There has been a large number of publications in the last 2–3 years on subword-level and subword-aware NLMs,[1] especially for the cases when subwords are characters (Ling et al., 2015; Kim et al., 2016; Verwimp et al., 2017) or morphemes (Botha and Blunsom, 2014; Qiu et al., 2014; Cotterell and Schütze, 2015). Less work has been done on syllable-level or syllable-aware NLMs (Mikolov et al., 2012; Assylbekov et al., 2017; Yu et al., 2017). For a thorough and up-to-date review of the previous work on subword-aware neural language modeling we refer the reader to the paper by Vania and Lopez (2017), where the authors systematically compare different subword units (characters, character trigrams, BPE, morphs/morphemes) and different representation models (CNN, Bi-LSTM, summation) on languages with various morphological typology.

Tying weights in NLM: Reusing embeddings in word-level neural language models is a technique which was used earlier (Bengio et al., 2001; Mnih and Hinton, 2007) and studied in more details recently (Inan et al., 2017; Press and Wolf, 2017). However, not much work has been done on reusing parameters in subword-aware or subword-level language models. Jozefowicz et al. (2016) reused the *CharCNN* architecture of Kim et al. (2016) to dynamically generate softmax word embeddings without sharing parameters with an input word-embedding sub-network. They managed to significantly reduce the total number of parameters for large models trained on a huge dataset in English (1B tokens) with a large vocabulary (800K tokens) at the expense of deteriorated performance. Labeau and Allauzen (2017) used similar approach to augment the output word representations with subword-based embeddings. They experimented with characters and morphological decompositions, and tried different compositional models (CNN, Bi-LSTM, concatenation) on Czech dataset consisting of 4.7M tokens. They were not tying weights between input and output representations, since their preliminary experiments with tied weights gave worse results.

Our approach differs in the following aspects: we focus on the ways to *reuse* weights at output, seek both model size reduction *and* performance improvement in subword-aware language models, try different subword units (characters, syllables, and morphemes), and make evaluation on small (1M–2M tokens) and medium (17M–51M tokens) data sets across multiple languages.

[1] *Subword-level* LMs rely on subword-level inputs and make predictions at the level of subwords; *subword-aware* LMs also rely on subword-level inputs but make predictions at the level of words.

3 Recurrent Neural Language Model

Let $\mathcal{W}$ be a finite vocabulary of words. We assume that words have already been converted into indices. Let $\mathbf{E}^{\text{in}}_{\mathcal{W}} \in \mathbb{R}^{|\mathcal{W}| \times d_{\mathcal{W}}}$ be an input embedding matrix for words — i.e., it is a matrix in which the wth row (denoted as $\mathbf{w}$) corresponds to an embedding of the word $w \in \mathcal{W}$.

Based on word embeddings $\mathbf{w_{1:k}} = \mathbf{w_1}, \ldots, \mathbf{w_k}$ for a sequence of words $w_{1:k}$, a typical word-level RNN language model produces a sequence of states $\mathbf{h_{1:k}}$ according to

$$\mathbf{h_t} = \text{RNNCell}(\mathbf{w_t}, \mathbf{h_{t-1}}), \quad \mathbf{h_0} = \mathbf{0}. \tag{1}$$

The last state $\mathbf{h_k}$ is assumed to contain information on the whole sequence $w_{1:k}$ and is further used for predicting the next word w_{k+1} of a sequence according to the probability distribution

$$\Pr(w_{k+1}|w_{1:k}) = \text{softmax}(\mathbf{h_k}\mathbf{E}^{\text{out}}_{\mathcal{W}} + \mathbf{b}), \tag{2}$$

where $\mathbf{E}^{\text{out}}_{\mathcal{W}} \in \mathbb{R}^{d_{\text{LM}} \times |\mathcal{W}|}$ is an output embedding matrix, $\mathbf{b} \in \mathbb{R}^{|\mathcal{W}|}$ is a bias term, and d_{LM} is a state size of the RNN.

Subword-based word embeddings: One of the more recent advancements in neural language modeling has to do with segmenting words at input into subword units (such as characters, syllables, morphemes, etc.) and composing each word's embedding from the embeddings of its subwords. Formally, let $\mathcal{S}$ be a finite vocabulary of subwords,[2] and let $\mathbf{E}^{\text{in}}_{\mathcal{S}} \in \mathbb{R}^{|\mathcal{S}| \times d_{\mathcal{S}}}$ be an input embedding matrix for subwords. Any word $w \in \mathcal{W}$ is a sequence of its subwords $(s_1, s_2, \ldots, s_{n_w}) = \sigma(w)$, and hence can be represented as a sequence of the corresponding subword vectors:

$$[\mathbf{s_1}, \mathbf{s_2}, \ldots, \mathbf{s_{n_w}}]. \tag{3}$$

A subword-based word embedding model $E(\cdot; \mathbf{E}^{\text{in}}_{\mathcal{S}}, \mathbf{\Theta}^{\text{in}})$ with parameters $(\mathbf{E}^{\text{in}}_{\mathcal{S}}, \mathbf{\Theta}^{\text{in}})$ constructs a word vector $\mathbf{x}$ from the sequence of subword vectors (3), i.e.

$$\mathbf{x} = E(\sigma(w); \mathbf{E}^{\text{in}}_{\mathcal{S}}, \mathbf{\Theta}^{\text{in}}), \tag{4}$$

[2] As in the case of words, we assume that subwords have already been converted into indices.

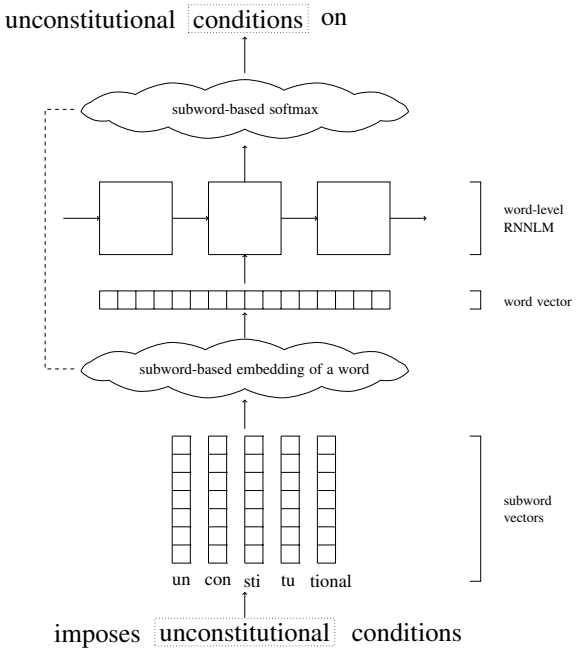


Figure 1: Subword-aware RNNLM with subword-based softmax.

which is then fed into a RNNLM (1) instead of a plain embedding $\mathbf{w}$. The additional parameters $\mathbf{\Theta}^{\text{in}}$ correspond to the way the embedding model constructs the word vector: for instance, in the *CharCNN* model of Kim et al. (2016), $\mathbf{\Theta}^{\text{in}}$ are the weights of the convolutional and highway layers.

Reusing word embeddings: Another recent technique in word-level neural language modeling is tying input and output word embeddings:

$$\mathbf{E}^{\text{in}}_{\mathcal{W}} = \left(\mathbf{E}^{\text{out}}_{\mathcal{W}}\right)^T,$$

under the assumption that $d_{\mathcal{W}} = d_{\text{LM}}$. Although being useful for word-level language modeling (Press and Wolf, 2017; Inan et al., 2017), this regularization technique is not directly applicable to subword-aware language models, as they receive subword embeddings at input and return word embeddings at output. In the next section we describe a simple technique to allow reusing subword embeddings $\mathbf{E}^{\text{in}}_{\mathcal{S}}$ as well as other parameters $\mathbf{\Theta}^{\text{in}}$ in a subword-aware RNNLM.

4 Reusing Weights

Let $\mathbf{E}^{\text{out}}_{\mathcal{S}}$ be an output embedding matrix for subwords and let us modify the softmax layer (2) so that it utilizes $\mathbf{E}^{\text{out}}_{\mathcal{S}}$ instead of the word embedding matrix $\mathbf{E}^{\text{out}}_{\mathcal{W}}$. The idea is fairly straightforward: we reuse an embedding model (4) to construct a new word embedding matrix:

$$\hat{\mathbf{E}}^{\text{out}}_{\mathcal{W}} = [E(\sigma(w); \mathbf{E}^{\text{out}}_{\mathcal{S}}, \mathbf{\Theta}^{\text{out}}) \text{ for } w \in \mathcal{W}], \quad (5)$$

and use $\hat{\mathbf{E}}^{\text{out}}_{\mathcal{W}}$ instead of $\mathbf{E}^{\text{out}}_{\mathcal{W}}$ in the softmax layer (2). Such modification of the softmax layer will be referred to as *subword-based softmax*. The overall architecture of a subword-aware RNNLM with subword-based softmax is given in Figure 1. Such a model allows several options for reusing embeddings and weights, which are discussed below.

- Reusing neither subword embeddings nor embedding model weights: As was shown by Jozefowicz et al. (2015), this can significantly reduce the total number of parameters for large models trained on huge datasets (1B tokens) with large vocabularies (800K tokens). However, we do not expect significant reductions on smaller data sets (1-2M tokens) with smaller vocabularies (10-30K tokens), which we use in our main experiments.
- Reusing subword embeddings (*RE*) can be done by setting $\mathbf{E}^{\text{out}}_{\mathcal{S}} = \mathbf{E}^{\text{in}}_{\mathcal{S}}$ in (5). This will give a significant reduction in model size for models with $|\mathbf{E}^{\text{in}}_{\mathcal{S}}| \gg |\mathbf{\Theta}^{\text{in}}|$,[3] such as the morpheme-aware model of Botha and Blunsom (2014).
- Reusing weights of the embedding model (*RW*) can be done by setting $\mathbf{\Theta}^{\text{out}} = \mathbf{\Theta}^{\text{in}}$. Unlike the previous option, this should significantly reduce sizes of models with $|\mathbf{E}^{\text{in}}_{\mathcal{S}}| \ll |\mathbf{\Theta}^{\text{in}}|$, such as the character-aware model of Kim et al. (2016).
- Reusing both subword embeddings and weights of the embedding model (*RE+RW*) can be done by setting $\mathbf{E}^{\text{out}}_{\mathcal{S}} = \mathbf{E}^{\text{in}}_{\mathcal{S}}$ and $\mathbf{\Theta}^{\text{out}} = \mathbf{\Theta}^{\text{in}}$ simultaneously in (5). This should significantly reduce the number of trainable parameters in any subword-aware model. Here we use exactly the same word representations both at input and at output, so this option corresponds to the reusing of plain word embeddings in pure word-level language models.

5 Experimental Setup

Data sets: All models are trained and evaluated on the PTB (Marcus et al., 1993) and the WikiText-2 (Merity et al., 2017) data sets. For the PTB we utilize the standard training (0-20), validation (21-22), and test (23-24) splits along with preprocessing per Mikolov et al. (2010). WikiText-2 is an alternative to PTB, which is approximately two times as large in size and three times as large

[3] $|\mathbf{A}|$ denotes number of elements in $\mathbf{A}$.

in vocabulary (Table 1).

Data set	T	$\lvert\mathcal{W}\rvert$	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{M}\rvert$
PTB	0.9M	10K	5.9K	3.4K
WikiText-2	2.1M	33K	19.5K	8.8K

Table 1: Corpus statistics. T = number of tokens in training set; $|\mathcal{W}|$ = word vocabulary size; $|\mathcal{S}|$ = syllable vocabulary size; $|\mathcal{M}|$ = morph vocabulary size.

Subword-based embedding models: We experiment with existing representational models which have previously proven effective for language modeling.

- *CharCNN* (Kim et al., 2016) is a character-aware convolutional model, which performs on par with the 2014–2015 state-of-the-art word-level LSTM model (Zaremba et al., 2014) despite having 60% fewer parameters.
- *SylConcat* is a simple concatenation of syllable embeddings suggested by Assylbekov et al. (2017), which underperforms *CharCNN* but has fewer parameters and is trained faster.
- *MorphSum* is a summation of morpheme embeddings, which is similar to the approach of Botha and Blunsom (2014) with one important difference: the embedding of the word itself is not included into the sum. We do this since other models do not utilize word embeddings.

In all subword-aware language models we inject a stack of two highway layers (Srivastava et al., 2015) right before the word-level RNNLM as done by Kim et al. (2016), and the non-linear activation in any of these highway layers is a ReLU. The highway layer size is denoted by d_{HW}.

Word-level RNNLM: There is a large variety of RNN cells to choose from in (1). To make our results directly comparable to the previous work of Inan et al. (2017), Press and Wolf (2017) on reusing word embeddings we select a rather conventional architecture – a stack of two LSTM cells (Hochreiter and Schmidhuber, 1997).

Hyperparameters: We experiment with two configurations for the state size d_{LM} of the word-level RNNLM: 200 (small models) and 650 (medium-sized models). In what follows values outside brackets correspond to small models, and values within brackets correspond to medium models.

- *CharCNN:* We use the same hyperparameters as in the work of Kim et al. (2016), where "large model" stands for what we call "medium-sized model".
- *SylConcat:* $d_{\mathcal{S}} = 50$ (200), $d_{\text{HW}} = 200$ (800). These choices are guided by the work of Assylbekov et al. (2017).
- *MorphSum*: $d_{\mathcal{S}} = d_{\text{HW}} = 200$ (650). These choices are guided by Kim et al. (2016).

Optimizaton method is guided by the previous works (Zaremba et al., 2014; Gal and Ghahramani, 2016) on word-level language modeling with LSTMs. See Appendix A for details.

Syllabification and morphological segmentation: True syllabification of a word requires its grapheme-to-phoneme conversion and then its splitting up into syllables based on some rules. True morphological segmentation requires rather expensive morphological analysis and disambiguation tools. Since these are not always available for under-resourced languages, we decided to utilize Liang's widely-used hyphenation algorithm (Liang, 1983) and an unsupervised morphological segmentation tool, Morfessor 2.0 (Virpioja et al., 2013), as approximations to syllabification and morphological segmentation respectively. We use the default configuration of Morfessor 2.0. Syllable and morpheme vocabulary sizes for both PTB and WikiText-2 are reported in Table 1.

6 Results

In order to investigate the extent to which each of our proposed options benefits the language modeling task, we evaluate all four modifications (no reusing, RE, RW, RE+RW) for each subword-aware model against their original versions and word-level baselines. The results of evaluation are given in Table 2. We have both negative and positive findings which are summarized below.

Negative results:

- The 'no reusing' and RW options should never be applied in subword-aware language models as they deteriorate the performance.
- Neither of the reusing options benefits *CharCNN* when compared to the original model with a plain softmax layer.

Positive results:

- The *RE+RW* option puts *CharCNN*'s performance close to that of the original version, while reducing the model size by 30–75%.
- The *RE* and *RE+RW* are the best reusing options for *SylConcat*, which make it on par with the original *CharCNN* model, despite having 35–75% fewer parameters.

Model	PTB				Wikitext-2			
	Small		Medium		Small		Medium	
	Size	PPL	Size	PPL	Size	PPL	Size	PPL
Word	4.7M	88.1	19.8M	79.8	14M	111.9	50.1M	95.7
Word + reusing word emb's	2.7M	86.6	13.3M	74.5	7.3M	104.1	28.4M	89.9
CharCNN (original)	4.1M	**87.3**	19.4M	**77.1**	8.7M	**101.6**	34.5M	88.7
CharCNN	3.3M	97.5	18.5M	89.2	3.3M	110.6	18.5M	—
CharCNN + RE	3.3M	99.1	18.5M	82.9	3.3M	110.2	18.5M	—
CharCNN + RW	2.2M	93.5	13.6M	103.2	2.2M	111.5	13.6M	—
CharCNN + RE + RW	2.2M	91.0	13.6M	79.9	2.2M	101.8	13.6M	—
SylConcat (original)	3.2M	89.0	18.7M	77.9	8.5M	105.7	36.6M	91.4
SylConcat	1.7M	96.9	17.7M	90.5	3.1M	118.1	23.2M	114.8
SylConcat + RE	1.4M	**87.4**	16.6M	**75.7**	2.1M	**101.0**	19.3M	94.2
SylConcat + RW	1.6M	99.9	15.2M	96.2	2.9M	118.9	19.4M	112.1
SylConcat + RE + RW	1.2M	88.4	12.7M	76.2	1.9M	**101.0**	15.5M	**86.7**
MorphSum (original)	3.5M	87.5	17.2M	78.5	9.3M	101.9	35.8M	90.1
MorphSum	2.4M	89.0	14.5M	82.4	4.5M	100.3	21.7M	86.7
MorphSum + RE	1.6M	85.5	12.3M	74.1	2.8M	97.6	15.9M	81.2
MorphSum + RW	2.2M	89.6	12.8M	81.0	4.4M	101.4	20.0M	86.6
MorphSum + RE + RW	1.5M	**85.1**	10.7M	**72.2**	2.6M	**96.5**	14.2M	**77.5**

Table 2: Results. The pure word-level models and original versions of subword-aware models (with regular softmax) serve as baselines. Reusing the input embedding architecture at output in *CharCNN* leads to prohibitively slow models when trained on WikiText-2 (≈800 tokens/sec on NVIDIA Titan X Pascal); we therefore abandoned evaluation of these configurations.

- The *RE* and *RE+RW* configurations benefit *MorphSum* making it not only better than its original version but also better than all other models and significantly smaller than the word-level model with reused embeddings.

In what follows we proceed to analyze the obtained results.

6.1 CharCNN is biased towards surface form

We hypothesize that the reason *CharCNN* does not benefit from tied weights is that CNN over character embeddings is an excessively flexible model which learns to adapt to a surface form more than to semantics. To validate this hypothesis we pick several words[4] from the English PTB vocabulary and consider their nearest neighbors under cosine similarity as produced by the medium-sized models (with the regular softmax layer) at input (Table 3). As we can see from the examples, the *CharCNN* model is somewhat more biased towards surface forms at input than *SylConcat* and *MorphSum*.[5] When *CharCNN* is reused to generate a softmax embedding matrix this bias is propagated to output embeddings as well (Table 3).

6.2 Tying weights bottom-up

From Table 2 one can notice that tying weights without tying subword embeddings (*RW*) *always* results in worse performance than the tying both weights and embeddings (*RE+RW*). Recall that subword embedding lookup is done before the weights of subword-aware embedding model are used (see Figure 1). This leads us to the following

Conjecture. *Let* $\mathbf{E}^{in}_{\mathcal{S}} = \mathbf{\Theta}^{in}_0, \mathbf{\Theta}^{in}_1, \mathbf{\Theta}^{in}_2, \dots, \mathbf{\Theta}^{in}_n$ *be the parameters of the consecutive layers of a subword-aware input embedding model (4), i.e.* $\mathbf{x} = \mathbf{x}^{(n)} = f_n\left(\mathbf{x}^{(n-1)}; \mathbf{\Theta}^{in}_n\right)$, $\dots$, $\mathbf{x}^{(1)} = f_1\left(\mathbf{x}^{(0)}; \mathbf{\Theta}^{in}_1\right)$, $\mathbf{x}^{(0)} = f_0\left(\sigma(w); \mathbf{E}^{in}_{\mathcal{S}}\right)$ *and let* $\mathbf{E}^{out}_{\mathcal{S}} = \mathbf{\Theta}^{out}_0, \mathbf{\Theta}^{out}_1, \mathbf{\Theta}^{out}_2, \dots, \mathbf{\Theta}^{out}_n$ *be the parameters of the consecutive layers of a subword-aware embedding model used to generate the output projection matrix (5). Let* A *be a subword-aware neu-*

[4] We pick the same words as Kim et al. (2016).

[5] A similar observation for character-aware NLMs was made by Vania and Lopez (2017).

	Model	In Vocabulary					Out-of-Vocabulary		
		while	*his*	*you*	*richard*	*trading*	*computer-aided*	*misinformed*	*looooook*
INPUT EMBEDDINGS	CharCNN (original)	chile	hhs	god	graham	traded	computer-guided	informed	look
		whole	its	we	harold	tradition	computerized	performed	looks
		meanwhile	her	your	edward	heading	computer-driven	formed	looking
		although	this	i	ronald	eroding	black-and-white	confirmed	looked
	SylConcat (original)	although	my	kemp	thomas	printing	computer-guided	reinforced	—
		though	historic	welch	robert	working	computer-driven	surprised	—
		when	your	i	stephen	lending	computerized	succeeding	—
		mean	irish	shere	alan	recording	computer	succeed	—
	MorphSum (original)	although	mystery	i	stephen	program-trading	cross-border	informed	nato
		whenever	my	ghandi	leonard	insider-trading	bank-backed	injured	lesko
		when	whiskey	we	william	relations	pro-choice	confined	imo
		1980s	sour	cadillac	robert	insurance	government-owned	formed	swapo
I/O EMB'S	CharCNN + RE + RW	thi	her	we	gerard	trades	computer-guided	informed	look
		when	its	your	gerald	trader	large-scale	performed	outlook
		after	the	young	william	traders	high-quality	outperformed	looks
		above	heir	why	edward	trade	futures-related	confirmed	looked

Table 3: Nearest neighbors based on cosine similarity. We underline character ngrams in words which are close to the given word orthographically rather than semantically. The `pyphen` syllabifier, which is used in *SylConcat*, failed to segment the word 'looooook' into syllables, and therefore its neighbors are not available.

ral language model in which the first $(j+1)$ layers of input and output embedding sub-networks have tied weights:

$$\forall i = \overline{0, j} : \quad \mathbf{\Theta}_i^{in} = \mathbf{\Theta}_i^{out},$$

and let B be a model in which at least one layer below the $(j+1)^{th}$ layer has untied weights:

$$\exists i = \overline{0, j-1} : \quad \mathbf{\Theta}_i^{in} \neq \mathbf{\Theta}_i^{out},\ \mathbf{\Theta}_j^{in} = \mathbf{\Theta}_j^{out}.$$

Then model B performs at most as well as model A, i.e. $PPL_A \leq PPL_B$.

To test this conjecture empirically, we conduct the following experiments: in all three embedding models (*CharCNN*, *SylConcat*, and *MorphSum*), we reuse different combinations of layers. If an embedding model has n layers, there are 2^n ways to reuse them, as each layer can either be tied or untied at input and output. However, there are two particular configurations for each of the embedding models that do not interest us: (i) when neither of the layers is reused, or (ii) when only the very first embedding layer is reused. Hence, for each model we need to check $2^n - 2$ configurations. For faster experimentation we evaluate only small-sized models on PTB. The results are reported in Table 4. As we can see, the experiments in general reject our conjecture: in *SylConcat* leaving an untied first highway layer between tied embedding and second highway layers (denote this as HW_2+Emb) turned out to be slightly better than tying all three layers (HW_2+HW_1+Emb). Recall, that a highway is a weighted average between nonlinear and identity transformations of the incoming vector:

$$\mathbf{x} \mapsto \mathbf{t} \odot \text{ReLU}(\mathbf{x}\mathbf{A} + \mathbf{b}) + (\mathbf{1} - \mathbf{t}) \odot \mathbf{x},$$

where $\mathbf{t} = \sigma(\mathbf{x}\mathbf{W} + \mathbf{c})$ is a transform gate, $\mathbf{A}$, $\mathbf{W}$, $\mathbf{b}$ and $\mathbf{c}$ are trainable parameters, and $\odot$ is the element-wise multiplication operator. To find out why leaving an untied highway below a tied one is beneficial in *SylConcat*, we compare the distributions of the transform gate values $\mathbf{t}$ from the first highway layers of both configurations, HW_2+Emb and HW_2+HW_1+Emb, in *SylConcat* and *MorphSum* (Figure 2).

We can see that *SylConcat* heavily relies on nonlinearity in the first highway layer, while *MorphSum* does not utilize much of it. This means that in MorphSum, the highway is close to an identity operator ($\mathbf{t} \approx \mathbf{0}$), and does not transform the sum of morpheme vectors much, either at input or at output. Therefore, tying the first highway layer is natural to *Morh-Sum*. *SylConcat*, on the other hand, applies non-linear transformations to the concatenation of syllable vectors, and hence makes additional preparations of the word vector for the needs of the RNNLM at input and for Softmax prediction at output. These needs differ from each other (as shown in the next subsection). This is why *SylConcat* benefits from an additional degree of freedom when the first highway is left untied.

Despite not being true in all cases, and due to being true in many cases, we believe that the above-mentioned conjecture is still useful. In short it can be summarized as a practical hands-

HW_2	HW_1	CNN	Emb	PPL
		✓		94.1
		✓	✓	**92.8**
	✓			94.6
	✓		✓	94.5
	✓	✓		93.1
	✓	✓	✓	**90.1**
✓				94.9
✓			✓	99.2
✓		✓		94.1
✓		✓	✓	92.5
✓	✓			94.3
✓	✓		✓	97.8
✓	✓	✓		96.3
✓	✓	✓	✓	**91.0**

HW_2	HW_1	Emb	PPL
	✓		95.4
	✓	✓	**87.4**
✓			99.0
✓		✓	**87.9**
✓	✓		96.2
✓	✓	✓	88.4

HW_2	HW_1	Emb	PPL
	✓		90.0
	✓	✓	**84.7**
✓			89.9
✓		✓	85.7
✓	✓		89.4
✓	✓	✓	**85.1**

Table 4: Reusing different combinations of layers in small *CharCNN* (left), small *SylConcat* (top right) and small *MorphSum* on PTB data. "✓" means that the layer is reused at output.

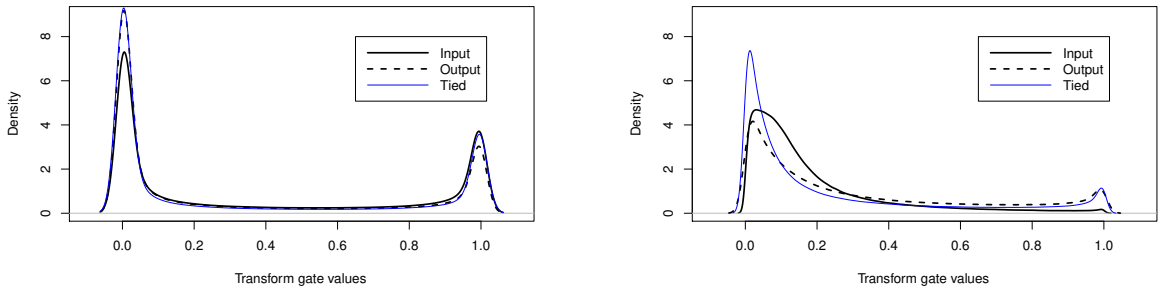


Figure 2: Kernel density estimations of the transform gate values of the first highway layers in *SylConcat* (left) and *MorphSum*. Values corresponding to 'Input' and 'Output' curves come from the HW_2+Emb configurations, while those corresponding to 'Tied' curves come from the HW_2+HW_1+Emb configurations.

on rule:

Layers should be tied consecutively bottom-up,

i.e. one should not leave untied layer(s) below a tied one. Keep in mind that this rule does not guarantee a performance increase as more and more layers are tied. It only says that leaving untied weights below the tied ones is likely to be worse than not doing so.

6.3 Difference between input and output embeddings

One can notice from the results of our experiments (Table 4) that having an untied second highway layer above the first one always leads to better performance than when it is tied. This means that there is a benefit in letting word embeddings slightly differ at input and output, i.e. by specializing them for the needs of RNNLM at input and of Softmax at output. This specialization is quite natural, as input and output representations of words have two different purposes: input representations send a signal to the RNNLM about the current word in a sequence, while output representations are needed to predict the next word given *all* the preceding words. The difference between input and output word representations is discussed in greater detail by Garten et al. (2015) and Press and Wolf (2017). Here we decided to verify the difference indirectly: we test whether intrinsic dimensionality of word embeddings significantly differs at input and output. For this, we apply principal component analysis to word embeddings produced by all models in "no reusing" mode. The results are given in Figure 3, where we can see that dimensionalities of input and output embeddings differ in the word-level model, *CharCNN*, and *SylConcat* models, but the difference is less significant in *MorphSum* model. Interestingly, in word-level and *MorphSum* models the output embeddings have more principal components than the input ones. In *CharCNN* and *SylConcat*, however, results are to other way around. We defer the study of this phenomenon to the future work.

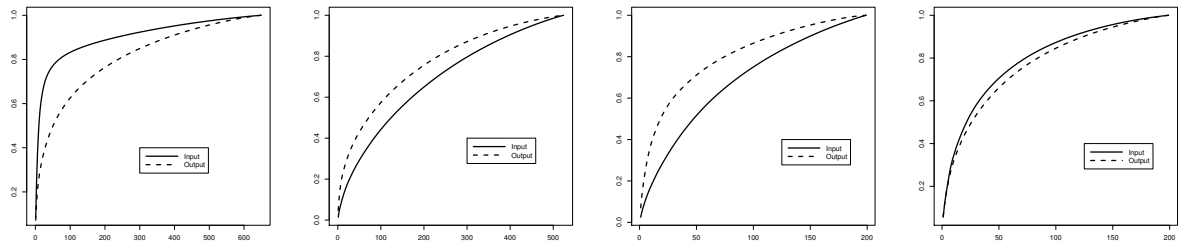


Figure 3: PCA applied to input and word embeddings produced by different models. Horizontal axis corresponds to number of principal components, vertical axis corresponds to percentage of total variance to retain. From left to right: word-level model, *CharCNN*, *SylConcat*, *MorphSum*.

6.4 CharCNN generalizes better than MorphSum

One may expect larger units to work better than smaller units, but smaller units to generalize better than larger units. This certainly depends on how one defines generalizability of a language model. If it is an ability to model unseen text with *unseen* words, then, indeed, character-aware models may perform better than syllable- or morpheme-aware ones. This can be partially seen from Table 3, where the OOV words are better handled by *CharCNN* in terms of in-vocabulary nearest neighbors. However, to fully validate the above-mentioned expectation we conduct additional experiments: we train two models, *CharCNN* and *MorphSum*, on PTB and then we evaluate them on the test set of Wikitext-2 (245K words, 10K word-types). Some words in Wikitext-2 contain characters or morphemes that are not present in PTB, and therefore such words cannot be embedded by *CharCNN* or *MorphSum* correspondingly. Such words were replaced by the `<unk>` token, and we call them new OOVs[6]. The results of our experiments are reported in Table 5. Indeed, *CharCNN* faces less OOVs on unseen text, and thus generalizes better than *MorphSum*.

Model	# new OOVs	PPL
CharCNN + RE + RW	3659	306.8
MorphSum + RE + RW	4195	316.2

Table 5: Training on PTB and testing on Wikitext-2.

6.5 Performance on non-English Data

According to Table 2, *MorphSum+RE+RW* comfortably outperforms the strong baseline *Word+RE* (Inan et al., 2017). It is interesting to see whether this advantage extends to non-English languages which have richer morphology. For this purpose we conduct evaluation of both models on small (1M tokens) and medium (17M–51M tokens) data in five languages (see corpora statistics in Appendix B). Due to hardware constraints we only train the small models on medium-sized data. We used the same architectures for all languages and did not perform any language-specific tuning of hyperparameters, which are specified in Appendix A. The results are provided in Table 6. As one can see, the advantage of the morpheme-aware model over the word-level one is even more pronounced for non-English data. Also, we can notice that the gain is larger for small data sets. We hypothesize that the advantage of *MorphSum+RE+RW* over *Word+RE* diminishes with the decrease of type-token ratio (TTR). A scatterplot of PPL change versus TTR (Figure 4) supports this hypothesis. Moreover, there is a strong correlation between these two quantities: $\hat{\rho}(\Delta\text{PPL}, \text{TTR}) = 0.84$, i.e. one can predict the mean decrease in PPL from the TTR of a text with a simple linear regression:

$$\Delta\text{PPL} \approx 2,109 \times \text{TTR}.$$

	Model	FR	ES	DE	CS	RU	
S	Word+RE	218	205	305	514	364	D-S
S	MorphSum+RE+RW	**188**	**171**	**246**	**371**	**237**	D-S
M	Word+RE	205	193	277	488	351	D-S
M	MorphSum+RE+RW	**172**	**157**	**222**	**338**	**210**	D-S
S	Word+RE	167	149	285	520	267	D-M
S	MorphSum+RE+RW	**159**	**143**	**242**	**463**	**229**	D-M

Table 6: Evaluation on non-English data. *MorphSum+RE+RW* has significantly less parameters than *Word+RE* (Appendix B). S — small model, M — medium model, D-S — small data, D-M — medium data; FR — French, ES — Spanish, DE — German, CS — Czech, RU — Russian.

[6]These are "new" OOVs, since the original test set of Wikitext-2 already contains "old" OOVs marked as `<unk>`.

Model	PTB	WT-2	CS	DE	ES	FR	RU
AWD-LSTM-Word w/o emb. dropout	61.38	68.50	410	241	145	151	232
AWD-LSTM-MorphSum + RE + RW	61.17	66 .92	253	177	126	140	162

Table 7: Replacing LSTM with AWD-LSTM.

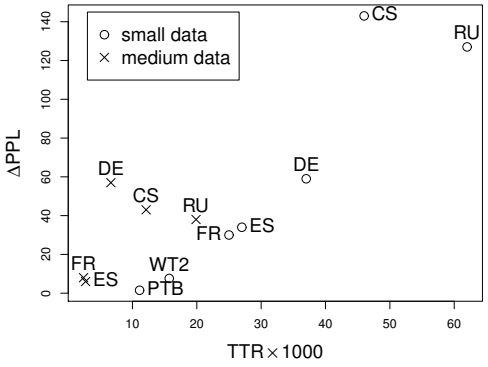


Figure 4: PPL improvement vs TTR. $\Delta\text{PPL} = \text{PPL}_{\text{Word+RE}} - \text{PPL}_{\text{MorphSum+RE+RW}}$.

6.6 Replacing LSTM with AWD-LSTM

The empirical perplexities in Table 2 are way above the current state-of-the-art on the same datasets (Melis et al., 2018). However, the approach of Melis et al. (2018) requires thousands of evaluations and is feasible for researchers who have access to hundreds of GPUs. Unfortunately, we do not have such access. Also, the authors do not disclose the optimal hyperparameters they found, and thus we could not reproduce their models. There is another state-of-the-art language model, AWD-LSTM (Merity et al., 2018), which has open-source code. We replaced this model's word embedding layer with the *MorphSum* subnetwork and fully reused morpheme embeddings and other weights of *MorphSum* at output. We refer to such modification as *AWD-LSTM-MorphSum + RE + RW*. We trained both models without fine-tuning (due to time constraints) and we did not use embedding dropout (section 4.3 of Merity et al. (2018)) in either model, as it is not obvious how embeddings should be dropped in the case of *AWD-LSTM-MorphSum*. The results of evaluation on the PTB, Wikitext-2, and non-English datasets are given in Table 7.

Although *AWD-LSTM-MorphSum* is on par with *AWD-LSTM-Word* on PTB and is slightly better on Wikitext-2, replacing plain word embeddings with the subword-aware model with appropriately reused parameters is crucial for non-English data. Notice that AWD-LSTM underperforms LSTM (used by us) on Czech dataset (cf. Table 6). We think that the hyperparameters of AWD-LSTM in Merity et al. (2018) are thoroughly tuned for PTB and Wikitext-2 and may poorly generalize to other datasets.

7 Conclusion

There is no single best way to reuse parameters in all subword-aware neural language models: the reusing method should be tailored to each type of subword unit and embedding model. However, instead of testing an exponential (w.r.t. sub-network depth) number of configurations, it is sufficient to check only those where weights are tied consecutively bottom-up.

Despite being similar, input and output embeddings solve different tasks. Thus, fully tying input and output embedding sub-networks in subword-aware neural language models is worse than letting them be slightly different. This raises the question whether the same is true for pure word-level models, and we defer its study to our future work.

One of our best configurations, a simple morpheme-aware model which sums morpheme embeddings and fully reuses the embedding sub-network, outperforms the competitive word-level language model while significantly reducing the number of trainable parameters. However, the performance gain diminishes with the increase of training set size.

Acknowledgements

We gratefully acknowledge the NVIDIA Corporation for their donation of the Titan X Pascal GPU used for this research. The work of Zhenisbek Assylbekov has been funded by the Committee of Science of the Ministry of Education and Science of the Republic of Kazakhstan, contract # 346/018-2018/33-28, IRN AP05133700. The authors would like to thank anonymous reviewers for their valuable feedback, and Dr. J. N. Washington for proofreading an early version of the paper.

References

Zhenisbek Assylbekov, Rustem Takhanov, Bagdat Myrzakhmetov, and Jonathan N. Washington. 2017. Syllable-aware neural language models: A failure to beat character-aware ones. In *Proc. of EMNLP*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2001. A neural probabilistic language model `http://www.iro.umontreal.ca/œlisa/pointeurs/nips00_lm.ps`.

Jan Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *Proc. of ICML*.

Wenlin Chen, David Grangier, and Michael Auli. 2016. Strategies for training large vocabulary neural language models. In *Proc. of ACL*.

Ryan Cotterell and Hinrich Schütze. 2015. Morphological word-embeddings. In *Proc. of HLT-NAACL*.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Proc. of NIPS*.

Justin Garten, Kenji Sagae, Volkan Ustun, and Morteza Dehghani. 2015. Combining distributed vector representations for words. In *In Proc. of VS@HLT-NAACL*.

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

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. In *Proc. of ICLR*.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proc. of ACL-IJCNLP*.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410* .

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proc. of ICML*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Proc. of AAAI*.

Matthieu Labeau and Alexandre Allauzen. 2017. Character and subword-based word representation for neural language modeling prediction. In *Proc. of SCLeM@EMNLP*.

Franklin Mark Liang. 1983. *Word Hy-phen-a-tion by Com-put-er*. Citeseer.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proc. of EMNLP*.

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

Gábor Melis, Chris Dyer, and Phil Blunsom. 2018. On the state of the art of evaluation in neural language models. In *Proc. of ICLR*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing lstm language models .

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *Proc. of ICLR*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proc. of INTERSPEECH*.

Tomáš Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and Jan Cernocky. 2012. Subword language modeling with neural networks. *preprint (http://www. fit. vutbr. cz/imikolov/rnnlm/char. pdf)* .

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proc. of ICML*.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proc. of EACL*.

Siyu Qiu, Qing Cui, Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Co-learning of word representations and morpheme representations. In *Proc. of COLING*.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Proc. of NIPS*.

Clara Vania and Adam Lopez. 2017. From characters to words to in between: Do we capture morphology? In *Proc. of ACL*.

Lyan Verwimp, Joris Pelemans, Patrick Wambacq, et al. 2017. Character-word lstm language models. In *Proc. of EACL*.

Sami Virpioja, Peter Smit, Stig-Arne Grönroos, Mikko Kurimo, et al. 2013. Morfessor 2.0: Python implementation and extensions for morfessor baseline .

Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proc. of the IEEE* 78(10):1550–1560.

Seunghak Yu, Nilesh Kulkarni, Haejun Lee, and Jihie Kim. 2017. Syllable-level neural language model for agglutinative language. In *Proc. of SCLeM@EMNLP*.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329* .

A Optimization

Training the models involves minimizing the negative log-likelihood over the corpus $w_{1:K}$:

$$-\sum_{k=1}^{T} \log \Pr(w_k|w_{1:k-1}) \longrightarrow \min,$$

by truncated BPTT (Werbos, 1990). We backpropagate for 35 time steps using stochastic gradient descent where the learning rate is initially set to

- 1.0 in small word-level models,
- 0.5 in small and medium *CharCNN*, medium *SylConcat* (SS, SS+RW) models,
- 0.7 in all other models,

and start decaying it with a constant rate after a certain epoch. This is 5 and 10 for the small word-level and all other networks respectively except *CharCNN*, for which it is 12. The decay rate is 0.9. The initial values for learning rates were tuned as follows: for each model we start with 1.0 and decrease it by 0.1 until there is convergence at the very first epoch. We use a batch size of 20. We train for 70 epochs. Parameters of the models are randomly initialized uniformly in $[-0.1, 0.1]$ and in $[-0.05, 0.05]$ for the small and medium networks, except the forget bias of the word-level LSTM, which is initialized to 1, and the transform bias of the highway layer, which is initialized to values around -2. For regularization we use a variant of variational dropout (Gal and Ghahramani, 2016) proposed by Inan et al. (2017). For PTB, the dropout rates are 0.3 and 0.5 for the small and medium models. For Wikitext-2, the dropout rates are 0.2 and 0.4 for the small and medium models. We clip the norm of the gradients (normalized by minibatch size) at 5.

For non-English small-sized data sets (Data-S) we use the same hyperparameters as for PTB. To speed up training on non-English medium-sized data (Data-M) we use a batch size of 100 and sampled softmax (Jean et al., 2015) with the number of samples equal to 20% of the vocabulary size (Chen et al., 2016).

	Data set	T	$\lvert\mathcal{W}\rvert$	$\lvert\mathcal{M}\rvert$
Small	French (FR)	1M	25K	6K
	Spanish (ES)	1M	27K	7K
	German (DE)	1M	37K	8K
	Czech (CS)	1M	46K	10K
	Russian (RU)	1M	62K	12K
Medium	French (FR)	57M	137K	26K
	Spanish (ES)	56M	152K	26K
	German (DE)	51M	339K	39K
	Czech (CS)	17M	206K	34K
	Russian (RU)	25M	497K	56K

Table 8: Non-English corpora statistics. T = number of tokens in training set; $|\mathcal{W}|$ = word vocabulary size; $|\mathcal{M}|$ = morph vocabulary size.

	Model	FR	ES	DE	CS	RU	
Small	Word + RE	5.6	6.1	8.0	10.0	13.4	Data-S
	MorphSum + RE + RW	2.1	2.3	2.5	2.8	3.2	
Medium	Word + RE	23.0	24.3	30.6	36.9	48.0	
	MorphSum + RE + RW	12.6	13.2	13.8	15.0	16.1	
Small	Word + RE	28.2	31.2	68.8	42.0	100.6	Data-M
	MorphSum + RE + RW	6.2	6.1	8.9	7.7	12.6	

Table 9: Model sizes in millions of trainable parameters.

B Non-English corpora statistics and model sizes

The non-English small and medium data comes from the 2013 ACL Workshop on Machine Translation[7] with pre-processing per Botha and Blunsom (2014). Corpora statistics is provided in Table 8.

Model sizes for *Word+RE* and *MorphSum+RE+RW*, which were evaluated on non-English data sets, are given in Table 9. *MorphSum+RE+RW* requires 45%–87% less parameters than *Word+RE*.

[7] http://www.statmt.org/wmt13/translation-task.html

Simple Models for Word Formation in English Slang

Vivek Kulkarni
Department of Computer Science
University of California, Santa Barbara
vvkulkarni@cs.ucsb.edu

William Yang Wang
Department of Computer Science
University of California, Santa Barbara
william@cs.ucsb.edu

Abstract

We propose generative models for three types of extra-grammatical word formation phenomena abounding in English slang: *Blends, Clippings, and Reduplicatives*. Adopting a data-driven approach coupled with linguistic knowledge, we propose simple models with state of the art performance on human annotated gold standard datasets. Overall, our models reveal insights into the generative processes of word formation in slang – insights which are increasingly relevant in the context of the rising prevalence of slang and non-standard varieties on the Internet.

1 Introduction

Linguistic analysis of slang has traditionally received little attention with some arguing that research on slang be assigned to an *"extra-linguistic darkness"* (Labov, 1972). However, Eble (2012) argues that the emergence of social media has predominantly increased the usage of slang and non standard forms and *"slang is now worldwide the vocabulary of choice of young people"*[1]. This increasing pervasiveness has recently motivated research on slang and the linguistic phenomena it manifests. Most notable are the works of (Mattiello, 2005, 2008, 2013), who argues that slang exhibits extra-grammatical properties that distinguish it from the standard form.

Specifically, linguistic phenomena like *alphabetisms*, *blending*, *clippings*, and *reduplicatives* abound in slang (see Table 1)[2]. Note the rich and varied word formation patterns ranging from simple abbreviations like `dink` to more complex combinations like `lambortini`, a blend of `lamborghini` and `martini`. Note further, that even within a particular class like BLENDS there are variations in what portions of the components are retained.

Word	Derived From	Type
dink	double income no kids	alphabetism
lambortini	lamborghini + martini	blend
diamat	dialectical + materialism	blend
tude	attitude	clipping (fore)
brill	brilliant	clipping (back)
yik-yak	yik	reduplicative

Table 1: Sample words depicting word formation in slang. Note the rich variation across different types.

These word formation mechanisms are not only attractive from a linguistic standpoint in deepening our understanding of slang but also have applications spanning the development of rich conversational agents and tools like brand name generators (Özbal and Strapparava, 2012). While such phenomena have been qualitatively studied by Mattiello (2008, 2013), computational models for their generation have not been proposed.

In this paper, we propose the first simple models for generating blends, clippings, and reduplicatives[3]. Our models incorporate linguistic insights coupled with data-driven analysis to model the above phenomena. In line with "Occam's razor", we strive for simplicity. The simplicity of our models not only implies better generalization, more robust estimation of parameters in the wake of small dataset sizes, and better interpret-ability but also yields state of the art performance.

Specifically, we show that by exploiting structural constraints, blend formation can be modeled as a simple *sequence labeling* problem as opposed to prior work which models it as a *general sequence to sequence* problem. This view enables the use of

[1] While the definition of slang is a controversial issue, we adopt a broad definition including non-standard expressions.

[2] While such phenomena are likely present in several languages, in this work we restrict ourselves to slang in English.

[3] While there has been some prior work on computationally modeling blends, generative models for clippings and reduplicatives have not been outlined.

Proceedings of NAACL-HLT 2018, pages 1424–1434
New Orleans, Louisiana, June 1 - 6, 2018.

a simple LSTM model to yield competitive performance. Similarly, we propose the first probabilistic generative models for clippings and reduplicatives effectively incorporating phonetic constraints. In a nutshell, our contributions are:

1. **Generative models.** We propose simple models for generating blends, clippings and, reduplicatives with state of the art performance.

2. **Linguistic Insights.** We reveal linguistic insights into these phenomena which we incorporate into the generative models.

3. **Resources.** We release all our models and the compiled datasets to aid further research [4].

2 Datasets and Definitions

Here, we define the extra-grammatical morphological phenomena modeled and describe the datasets used for our experiments and analysis. Mattiello (2013) argues that slang exhibits extra-grammatical morphological properties that distinguish them from the standard variety and identified four broad word formation phenomena[5] described below:

1. **Alphabetisms** are shortenings of a multiword sequence. Examples include `lol` from `laugh out loud` or `YOLO` from `you only live once`. They can be further sub-categorized into two types based on their pronunciation although the distinction may not always be clear: (a) *Acronyms* are pronounced using the regular reading rules (for example. YOLO) (b) *Initialisms* are pronounced letter by letter (for example. BBC).

2. **Blends** or portmanteaus, are formed by merging parts of existing words. For example, `edutainment` is a blend of `education` and `entertainment`. Prior work notes that blend formation does not exhibit rigid rules but only demonstrates affinities towards certain patterns of formation (Mattiello, 2013) suggesting learning based approaches to modeling blends (Deri and Knight, 2015; Gangal et al., 2017).

3. **Clippings** are constructed by shortening words (lexemes). For example, `berg` is a clipping of `iceberg`, `gym` is a clipping of `gymnasium` and `ammo` is a clipping of `ammunition`. Based on the portion that is being clipped, clippings are sub-categorized into three types: (a) BACK clipping where the beginning of the word (lexeme) is retained (like `brill` from `brilliant`) (b) a FORE clipping, where the end of a word is retained (like `choke` from `artichoke`) and (c) A COMPOUND clipping (`adman`), a clipping of a compound word (`advertisment man`).

4. **Reduplicatives** are word pairs constructed by either repeating a word (`boo boo`) or by alternating certain vowels or consonants so that they are phonologically similar (`clickety-clackety`, `teenie-weenie`, `itsy-bitsy`).

In our work, we propose generative models using a data-driven approach towards generating blends, clippings ,and reduplicatives. We do not consider generative models for alphabetisms since a majority of them can be trivially generated by picking the first letter of each word making up the acronym (for example. `laugh out loud` → `lol`). We now outline the datasets considered:

1. **Blends.** We consider a gold standard dataset $\mathcal{D}_{knight}$ of 400 blends constructed by (Deri and Knight, 2015) from Wikipedia as well as a larger list of 1624 blends manually compiled by (Gangal et al., 2017) called $\mathcal{D}_{large}$, a superset of $\mathcal{D}_{knight}$. We define $\mathcal{D}_{blind} = \mathcal{D}_{large} - \mathcal{D}_{knight}$.

2. **Clippings.** We consider a list of 576 human curated clippings constructed by Mattiello (2013) for our analysis of clippings. These were manually collected from a variety of sources including prior work and dictionaries like the *Oxford English Dictionary* and the *Merriam-Webster online dictionary*.

3. **Reduplicatives.** We consider a dataset of a total of 337 reduplicatives manually constructed by Mattiello (2013) also collected from prior work and online dictionaries[6].

[4] Code and data with reproducible results is available at `https://github.com/viveksck/simplicity`

[5] These categories are not exhaustive and several slang words and non-standard expressions do not fall into any of the above categories.

[6] We note that the dataset released by (Mattiello, 2013) is the largest human curated dataset of slang word formation we are aware of. Furthermore, large crowd sourced data-sets like Urban Dictionary are a poor fit for such modeling since they do not explicitly contain annotations by linguistic experts.

3 Models and Methods

We now describe our proposed models to generate blends, clippings, and reduplicatives. We precisely formulate the problem, specify our models and comprehensively outline our evaluation.

3.1 Blends

Problem Formulation Given a string $C = C_1\#C_2$, consisting of components C_1 and C_2, we seek to combine them to yield the blend B. For example, given $C_1 =$ `brad` and $C_2 =$ `angelina` such that $C =$ `brad#angelina`, we seek to generate the blend `brangelina`.

Existing Models (Deri and Knight, 2015) proposed a model to generate blends using multi-tape finite state transducers. Most recently, (Gangal et al., 2017) (the current state of the art) model this as a general sequence to sequence learning problem and propose a neural encoder-decoder architecture with attention to outperform the model by (Deri and Knight, 2015). However, this model fails to effectively exploit the inherent structure and linguistic constraints of blending. One implication is an exploration of an overly complex hypothesis space with a small amount of training data making it harder to generalize. A second implication is that the decoding phase uses exhaustive generation. In fact, the best model *exhaustively* generates all candidate strings, where the first part is a prefix of C_1 and the second part is a suffix of C_2 and scores them to pick the best candidate while using a backward model learned to generate the components given the blend. In contrast, we propose a more straightforward model that explicitly incorporates inherent linguistic constraints entirely obviating the need for decoding using exhaustive candidate generation yet yielding competitive performance.

3.1.1 Linguistic Insights into Blend Formation

While one can model the problem of learning to blend as a variable length sequence to sequence learning problem (akin to machine translation), we argue that incorporating structural constraints yields a different view of modeling the problem that can enable better generalization given the small amount of training data. We motivate this by observing the following constraints:

1. **Blend length and vocabulary constraints.** First, we observe that a majority of blends (99.0% in D_{knight} and 92.4% in D_{blind}) are formed by using only characters present in the original components. Second, the length of the blends in these cases is at-most the length of the components.

2. **Fixed length input output representation.** The blend B can thus be encoded as a string of the exact same length as C by noting that B only contains characters copied from C or deleted from C. Specifically, we represent B by $E(B)$ denoting the sequence of *copy* (**C**) and *delete* (**D**) operations needed to transform C to B. $E(B)$ can easily be computed by the edit distance function between C and B. For example, `brangelina` is encoded as `CCCDDDCCCCCCC`. Since $E(B)$ has the same length of C, we can now model this as a *fixed length sequence labeling* problem rather than a variable length sequence to sequence learning problem completely obviating the need for the *"encoder-decoder"* architecture for this large class of blends.

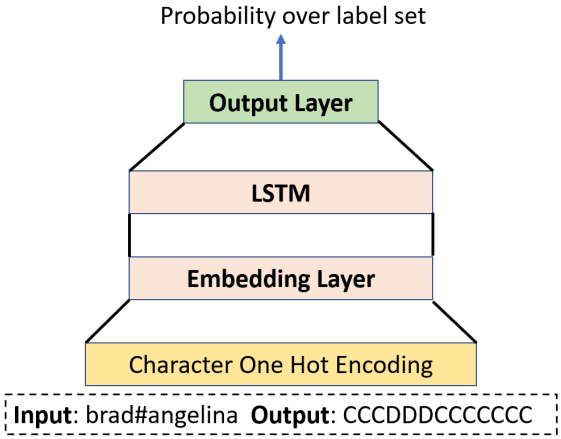


Figure 1: Our simple LSTM sequence labeling model for blends. Our model is considerably simpler both in complexity of the parameter space and in terms of implementation compared to the encoder-decoder model proposed by (Gangal et al., 2017) which modeled the problem as a machine translation problem.

Equivalent Problem Definition Given a string $C = C_1\#C_2$, consisting of components C_1 and C_2, learn $E(B)$, a labeling of each character in C from the label set $\{\mathbf{C}, \mathbf{D}\}$.

3.1.2 Proposed model: CopyCat

Model Architecture Inline with work on neural sequence labeling (Wang et al., 2015; Plank et al., 2016), our model uses a single layer long

short term memory (LSTM) (Graves et al., 2012) and an embedding layer as depicted in Figure 1. Since the size of the training set is relatively small, we reduce the number of learn-able parameters by using pre-trained character embeddings frozen during training. In particular, we use 50 dimensional character embeddings from (Gangal et al., 2017) obtained by training an LSTM language model on an English Dictionary. The implementation of the LSTM layer is described by (Graves et al., 2012; Wang et al., 2015) and therefore omitted here. The output layer is a soft-max over the label set $\{\mathbf{C}, \mathbf{D}\}$.

Candidate Generation Our model outputs probability scores over the label set for each element in the sequence. As in previous works (Gangal et al., 2017), we use the output to generate an ordered candidate set $\mathcal{T}$. To construct $\mathcal{T}$, we use a simple top-K decoder which selects the k most probable label sequences. Finding the k most probable tag sequences from the soft-max outputs can be cast as finding the top k shortest simple paths in a directed acyclic graph which can be efficiently solved using (Yen, 1971; Eppstein, 1998). Note that the greedy decoder which just picks the most likely label at each position is a special case with $k = 1$. While the number of candidates generated by (Gangal et al., 2017) depends on the size of C, our model generates a constant number of candidates ($k = 5$) regardless of the input C.

Candidate Ranking and Selection While the list of candidates in $\mathcal{T}$, can be used to make a prediction we note that re-ranking these candidates can result in better performance and thus consider multiple ranking strategies:

1. **LSTM.** We consider only the scores as obtained by the LSTM with no re-ranking.

2. **LSTM + LM.** We augment the score of each candidate to include both the score of the LSTM as well as its score according to a character level language model where the language model is trained on a large amount of unsupervised text [7].

3. **LSTM + LM + LEN.** Figure 2 shows a least squares fit to the length of the blends versus the length of its component, suggesting a strong correlation between these two variables. We capture this notion through a probabilistic model. Specifically, we model $\mathbf{Pr}(Blend_{len}|Component_{len})$ by fitting a Bayesian Ridge Regression model to the training data and score each candidate on this model as well. Finally, we combine the scores obtained for the LSTM, the language model and the length model uniformly to yield the final score for each candidate.

In each of the above cases, we pick the topmost candidate as our prediction.

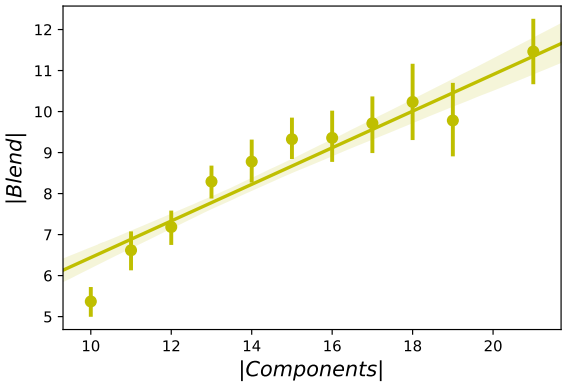


Figure 2: Best fit between blend length and the length of the input string, implying that one can infer the length of the blend from the length of the components using a regression model.

Table 2 illustrates the effects of each of these ranking strategies on an exemplary set of strings. While ranking using the raw scores of the LSTM yields blends which are close, observe that the LSTM alone does not capture ease of pronunciation effectively. For example, the top ranking candidate `brngelina` is relatively unlikely under both a character language model and a phoneme language model. Incorporating scores from the language model results in much more natural blends like `brangelina`. To observe the effect of the length model, note that for `kentucky#indiana` the blends obtained by LSTM and LSTM + LM are too short (`keiana` and `keana`). Incorporating the length model boosts scores for candidates closer to the target length yielding `keniana` which is closer to the target `kentuckiana`.

3.1.3 Evaluation

We compare our model to previous methods namely (Deri and Knight, 2015) and (Gangal et al., 2017). Inline with (Gangal et al., 2017), we evaluate our model on $\mathcal{D}_{knight}$ and $\mathcal{D}_{blind}$ (consisting of 1078

[7] We use words from the CMU pronouncing dictionary.

Input	LSTM	LSTM + LM	LSTM + LM + LEN	Gold standard
brad#angelina	brngelina	brangelina	brangelina	brangelina
animated#matrix	animtrix	animatrix	animatrix	animatrix
merkel#sarkozy	merkrkozy	merkkozy	merkkozy	merkozy
dramatic#drastic	draastic	drastic	dramastic	dramastic
kentucky#indiana	keiana	keana	keniana	kentuckiana
employability#agility	emplglity	emplgility	employgility	employagility

Table 2: Set of examples demonstrating the effect of incorporating the language model and length model when ranking candidates revealing insights into blend formation. Incorporating the language model generally improves the fluency of the blend while the length model helps generate blends with lengths close to the target.

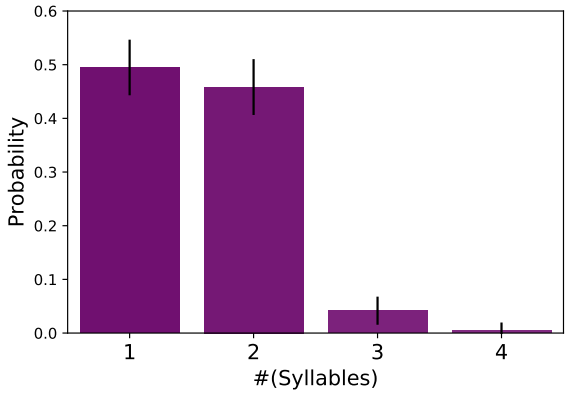


Figure 3: Distribution of the number of syllables in clippings. Most clippings have at-most 2 syllables but it is a challenge to infer whether a given word has a one or two syllable clipping.

instances)[8]. For evaluating on $\mathcal{D}_{knight}$, we use 10-fold cross-validation. For evaluating on $\mathcal{D}_{blind}$ we train our model on $\mathcal{D}_{knight}$ and report the mean score on the test-set obtained using 10 random splits of the training data. As in previous work, our metric is the edit distance between the predicted blend and the true blend.

3.2 Clippings

Problem Formulation Given a word w, learn a model that can generate its clipping c. For example, given the word `administration` we would like the model to output the clipping `admin`.

Proposed Models We motivate our models by presenting two insights into linguistic properties of clippings first noted by (Mattiello, 2013). First, most clippings are back clippings while fore clippings are relatively rare. Second, most clippings tend to have at most two syllables (see Figure 3). The first insight guides our model in determining whether to retain a prefix or a suffix of the original word. The second insight guides our model in determining how much to retain (or clip). In particular, we can capture this in two ways: (a) Working in the phoneme space and (b) Learn a function to predict the length of the clipping (which encodes the number of syllables implicitly).

3.2.1 CLIPPHONE

CLIPPHONE operates by mapping the word w to a sequence of phonemes, explicitly clipping in the phoneme space and mapping the phoneme sequence back to the grapheme space. In particular, the model can be described as follows: **(1)** Let θ and π represent multinomial distributions over clipping types and the number of syllables respectively. **(2)** Represent the word w as a sequence of phonemes $\mathcal{P}$ and identify each syllable. **(3)** Draw a sample l from π to represent the number of syllables in the clipping and the type t by drawing a sample from θ. **(4)** If $t \in \{\text{BACK}, \text{FORE}\}$, clip $\mathcal{P}$ to have exactly l syllables by selecting the appropriate length prefix or suffix depending on clipping type t to make exactly l syllables represented by $\mathcal{P}_{clip}$. If t is a COMPOUND, clip each word recursively and concatenate the outputs. **(5)** Map P_{clip} back to grapheme space to yield the clipping. **(6)** The parameters of θ and π, both multinomial distributions can be estimated from observed data via maximum likelihood estimators.

Phoneme-based representation Given w, we obtain its phoneme representation using the pretrained state-of-art neural model G2PSEQ2SEQ (Yao and Zweig, 2015)[9]. For example, the word `captain` is mapped to the following sequence of phonemes $\mathcal{P}$ given by `K AE P T AH N`.

Clipping in the phoneme space We identify the syllable boundaries in $\mathcal{P}$ by looking for vowel phonemes and clip $\mathcal{P}$ until it contains the desired number of syllables. For example, a one syllable clipping of `K AE P T AH N` is `K AE P T`

[8] We discard relatively rare instances with insertions.

[9] G2PSeq2Seq has a PER of 5.45% on CMUDict.

since AH is the beginning of the second syllable ($\mathcal{P}_{clip}$ = K AE P T).

Mapping clipping back to graphemes Finally, given the clipped phoneme sequence $\mathcal{P}_{clip}$, we map it back to the grapheme space by learning a sequence to sequence model to map a phoneme sequence to its grapheme sequence. We follow the same model architecture as used in G2PSEQ2SEQ (Yao and Zweig, 2015) but just flip the input and output sequences, thus learning a model to map phonemes back to graphemes. Finally, we quantify the effectiveness of this model explicitly in our evaluation by establishing an upper-bound on the expected performance using this model.

3.2.2 CLIPGRAPH

As we will demonstrate empirically, CLIPPHONE poses the following challenges: Since a given phoneme sequence can map to multiple grapheme sequences, explicitly mapping to phoneme space and back introduces errors and loss of fidelity [10]. This is compounded by noting that whether w's clipping should have one or two syllables is hard to predict (since both occur in almost equal proportions empirically, see Figure 3). Furthermore, deciding whether the clipping should end in a vowel or a consonant and determining the length is yet another challenge. As an alternative, we propose CLIPGRAPH where we seek to directly learn a model to predict the length of the clipping directly given w, obviating the need to work in phoneme space. CLIPGRAPH works as follows: **(1)** Let θ represent multinomial distributions over clipping types and let π be a model that predicts the length l of the clipping w_{clip} given w. **(2)** Draw a sample from θ to get the type of clipping t. **(3)** Retain the appropriate l-length prefix or suffix of w based on the type t handling compound words recursively. **(4)** We use Ridge regression to learn π from the training data and estimate θ using its maximum likelihood estimator.

Evaluation To evaluate our models, we consider a naive baseline NAIVE which clips w to one of its prefixes or suffixes randomly. For CLIPPHONE, we also consider one (CLIPPHONE-1SYL) and two (CLIPPHONE-2SYL) syllable clippings. Since (Jamet, 2009) notes that determining whether a word has a one or two syllable clipping is extremely challenging, we also consider a model with an oracle (CLIPPHONE(O)) on the number of syllables in the clipping to quantify this. Finally, we establish an upper bound (G2P-GOLD) on the performance of any method using our learned P2G model. G2P-GOLD maps the gold standard G to phoneme space and maps the resulting phoneme sequence back to yield $\hat{G}$ as the predicted clipping. We use the edit distance between the predicted clipping and the gold standard as the evaluation metric.

[10] K AE P T can be mapped to capped or capt.

3.3 Reduplicatives

Problem Formulation Given a word v, we seek to generate a word w such that $v.w$ is a reduplicative. For example, given the word flip, we would like to generate flop or flap to yield the reduplicatives flip-flop or flip-flap.

Proposed Model We motivate our model from a linguistic standpoint by referring to observations by (Mattiello, 2013) that a large fraction of reduplicatives can be formed by either (a) Duplicating the word DUPLICATE (boo-boo) (b) Exchanging the initial vowel VOWELEX (bing-bong) and (c) Exchanging the initial consonant CONEX (teenie-weenie). Other patterns include adding a consonant (artsy-fartsy) or adding schm/shm (moodle-schmoodle). In our work, we propose a generative model for the three dominant forms of reduplication mentioned above. Broadly, our model captures the notion that vowels and consonants display strong replacement preferences. For example, the vowel i is much more likely to be replaced with a than u (instances like clip-clap, wishy-washy). Similarly the consonant t is much more likely to be replaced by w (as in teenie-weenie, tinky-winky). We incorporate these insights into our generative model as follows: **(1)** Let θ be a distribution over the three different types of reduplicatives. Let ϕ_v, ψ_c be distributions over letters that replace vowel v and consonant c respectively. **(2)** Sample the type of reduplicative t generated from θ. **(3)** If t is DUPLICATE, set w to v and return w as the reduplicative component. **(4)** If t is VOWELEX, find the first vowel x with non-zero replacement probability. Sample the replacement z from ϕ_x and replace x with z. If t is CONEX, find the first consonant c with non-zero replacement probability and sample the replacement z from ψ_c. Replace c with z. **(5)** Return the edited string as the second component of the reduplicative. **(6)** The parame-

ters of multinomial distributions θ, ϕ_v, and ψ_c are estimated via MLE estimates from the data.

Evaluation We consider two baseline models (a) **LET** Uniformly replace a letter with another letter in v to return w. (b) **LET(COND)** Uniformly replace a letter (vowel or consonant) with a letter from its class. Since reduplicatives are characterized by phonologically similar sounds, merely using edit distance as a metric for evaluation would be ineffective. For example, even the ill-rhyming `flip-flsp` has the same edit distance as the correct reduplicative `flip-flop`. Thus, we use a distance measure (MIR) defined over the phoneme space (Hixon et al., 2011) which effectively captures the similarity of two phonetic sequences by modeling the affinities between pairs of phonemes using a point access mutation matrix and is superior over metrics like PER (phoneme error rate).

4 Experiments

Having described our models and evaluation metrics in the previous section, we proceed to evaluate our models empirically and describe our results.

Blends We train our model on $\mathcal{D}_{knight}$ and evaluate on the $\mathcal{D}_{blind}$ dataset as in (Gangal et al., 2017) comparing against previous methods (Deri and Knight, 2015; Gangal et al., 2017). We set the number of hidden units to 50 with a dropout probability of 0.5[11]. We use ADAM (Kingma and Ba, 2014) optimizer with an initial learning rate of 0.001 to train the model for 500 epochs with early stopping over a validation set.

Tables 3 and 4 show the mean edit distance of our predictions from the target blend. First, the evaluation on the $\mathcal{D}_{knight}$ dataset compares the performance of our models against previous baselines. Note that just using our basic model COPYCAT -(LSTM + LM) outperforms the baseline proposed by (Deri and Knight, 2015) (1.59 vs 1.49). Furthermore, observe that even our vanilla model (LSTM) significantly outperforms the equivalent "FORWARD" models by (Gangal et al., 2017)[12] that use greedy and beam-search decoding (1.90 and 2.37 vs 1.75). Moreover, observe that our model even achieves almost equivalent performance to the "FORWARD" state-of-art model which uses exhaustive decoding (1.37 vs 1.33). Furthermore, our model achieves competitive performance with the more complex models proposed by GANGAL-BACKWARD which use exhaustive decoding. We emphasize that we can achieve competitive performance using a simpler model without using exhaustive decoding. Similar observations can also be made for the evaluation of the $\mathcal{D}_{blind}$ data set (our best model yields a score of 1.91 vs 1.77). Altogether these observations suggest that even simple models with effective modeling of linguistic structure can perform competitively and even outperform overly complex models (see Table 6 for a few example predictions).

Model	Distance
KNIGHT	1.59
GANGAL-FORWARD (GREEDY)	1.90
GANGAL-FORWARD (BEAM)	2.37
GANGAL-FORWARD (SOTA)†	1.33
GANGAL-BACKWARD (SOTA)†	1.12
COPYCAT-(LSTM)	1.75
COPYCAT-(LSTM + LM)	1.49
COPYCAT-(LSTM + LM + LEN)	**1.40**
COPYCAT-(LSTM + LM + LEN)†	**1.37**

Table 3: 10-fold cross validation performance of our blending model COPYCAT in terms of edit distance (lower is better) on $\mathcal{D}_{knight}$ dataset. † indicates ensemble approach using sub-samples of training data consistent with previous work. Our simpler model yields competitive performance (especially compared to the state of art forward model) without the need for exhaustive decoding (which the state of art uses), uses a smaller learn-able parameter set while effectively using linguistic insights into the blending process.

Clippings We consider the dataset of clippings introduced by (Mattiello, 2013) and report the mean edit distance (μ) on a set of 173 clippings in Figure 4 (see Table 5 as well). First, both CLIPPHONE ($\mu = 3.39$) and CLIPGRAPH ($\mu = 2.65$) outperform the naive baseline ($\mu = 4.6$). Second, based on the empirical distribution, it is almost impossible to predict whether a word has a one or two syllables (0.49 vs 0.46 significant only at $\alpha > 0.2$) clipping and incorrect guesses critically affect the downstream performance. In the absence of such information, just clipping to one syllable yields better performance. However, when this information is exactly known (CLIPPHONE(O)), we note an improvement as expected ($\mu = 2.79$). Finally, CLIPGRAPH shows a small but not signif-

[11] All hyper-parameters were chosen using a validation set.

[12] We obtain the numbers for these baselines from the latest released data and predictions at https://github.com/vgtomahawk/Charmanteau-CamReady.

Model	Distance
KNIGHT	2.10
GANGAL-FORWARD (SOTA)†	1.76
GANGAL-BACKWARD (SOTA)†	1.77
COPYCAT-(LSTM)	2.27
COPYCAT-(LSTM + LM)	2.13
COPYCAT-(LSTM + LM + LEN)	**1.98**
COPYCAT-(LSTM + LM + LEN)†	**1.91**

Table 4: Performance of our blending model COPYCAT in terms of edit distance (lower is better) on $\mathcal{D}_{blind}$ dataset. † indicates ensemble approach using sub-samples of training data consistent with previous work. Our simpler model yields competitive performance without the need for exhaustive decoding, uses a smaller learn-able parameter set while effectively using linguistic insights into the blending process. To ensure fair comparison, numbers for the baselines were obtained by filtering the released predictions for these models to the same set of words our models were evaluated on.

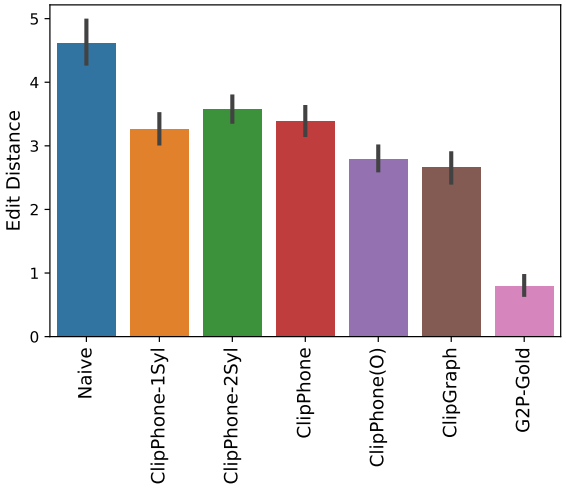


Figure 4: **Evaluation of clipping models** (lower is better). The NAIVE model is the performance lower bound. Both CLIPPHONE and CLIPGRAPH substantially outperform the baseline. Estimating whether a given word has one or two syllable clipping is a major challenge hindering CLIPPHONE since both cases are equally likely from empirical estimation. Using an oracle on the number of syllables (CLIPPHONE(O)) improves performance. CLIPGRAPH which operates purely in grapheme space performs as well as CLIPPHONE(O). G2PGOLD denotes a upper bound when using our P2G model.

icant advantage over methods working explicitly in the phoneme space which are prone to errors by imperfect conversion. Finally, the upper bound on the performance is substantially better than our models ($\mu = 0.79$) suggesting scope for future improvements.

Input	Our models		Gold
	CLIPPHONE	CLIPGRAPH	Clipping
cocaine	coke	coca	coke
juvenile	juve	juve	juvey
amelia	umm	amel	mel
alfred	fred	fred	fred
kid video	kidvid	kivid	kidvid

Table 5: Exemplary predictions from clippings models. We can generate one/two syllable clippings, as well as compound clippings. Note the effect of incorrect P2G conversion for `amelia` as `umm` which is pronounced similar to `ame`. The current G2P model does not incorporate stress. It is possible that incorporating stress into the model can address this scenario.

Input	Prediction	Gold	Dist
jizz#disney	jizzney	jizzney	0
scum#fuzz	scuzz	scuzz	0
phone#neck	phoneck	phoneck	0
woman#romance	womance	womance	0
piss#mishap	pisshap	pisshap	0
pass#asshole	passhole	passhole	0
spike#angel	spangel	spangel	0
man#amazon	mamazon	manazon	1
awkward#sauce	awkwauce	awksauce	1
bed#orgasm	bergasm	bedgasm	1

Table 6: Exemplary predictions from our simple blends model which suggests our model effectively captures blending phenomena by incorporating linguistic constraints.

Reduplicatives We evaluate our model on a held-out test set of 50 reduplicatives obtained using the manually compiled dataset by (Mattiello, 2013). We evaluate two flavors of our model: (a) OUR(NODUP) where we disallow generating duplicates (which are trivial to generate) and (b) OUR, the full-fledged model where duplicate reduplicatives are allowed. We report the mean MIR for each model over 10 independent runs in Figure 5. Our model consistently outperforms the baselines (LET, and LET(COND)) by at-least 8 percentage points suggesting it adeptly captures patterns in reduplicative formation. Finally, we examine the inferred probability distributions to gain insights into the linguistic phenomena in reduplicative formation few of which we outline: (a) The most common reduplicative types are DUPLICATE, VOWELEX followed by CONEX. (b) Vowel `i` is more likely to be replaced by `a` and `o` and (c) Consonant `t` is much more likely to be replaced by `w` and `l` (like

in `teenie-weenie`). We leave a comprehensive analysis of these patterns to future work.

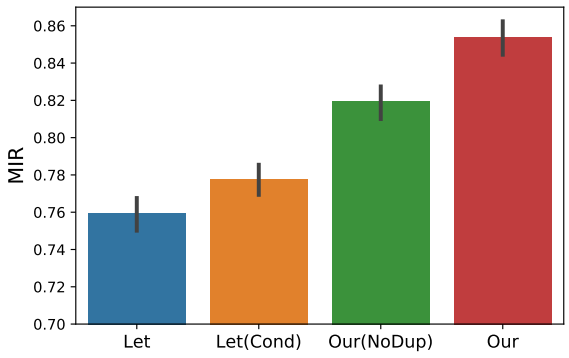


Figure 5: Performance of our reduplicative models based on the MIR metric (higher is better). Our models OUR(NODUP) and OUR consistently outperform baselines by at-least 8 percentage points (0.86 vs. 0.78).

5 Related Work

Blends, clippings and reduplicatives have been studied from a linguistic standpoint (Thun, 1963; Murata, 1990; Merlini Barbaresi, 2008; Hladky, 1998; Hamans, 1997; Fandrych, 2008; Moehkardi, 2016; Ungerer, 2007; Beal, 1991; Algeo, 1977; Smith et al., 2014; Shaw et al., 2014; Beliaeva, 2014; Broad et al., 2016; Renner et al., 2013; Gries, 2004a,b). Most relevant is the work of (Mattiello, 2005, 2008, 2013) who argues that slang is pervasive on the Internet, suggests its extra-grammatical nature and outlines some phonological and morphological properties. Specifically, these phenomena are discussed followed by a qualitative analysis on a manually compiled dataset of 1580 words from various sources. Recently (Deri and Knight, 2015) and (Gangal et al., 2017) study the problem of learning to blend and derive data-driven computational models for the task. (Deri and Knight, 2015) propose a model based on finite state transducers. (Gangal et al., 2017) outperform (Deri and Knight, 2015) by modeling the problem of generating blends as a variable length sequence to sequence learning problem and propose a neural encoder-decoder based model. We differ from all of these works in several ways. In contrast to the blending model proposed by Gangal et al. (2017), our model is simpler with lesser parameters, does not require using an encoder-decoder framework, or exhaustive decoding and yet yields competitive performance on a large class of blends. We also propose the first computational generative data-driven models for clippings and reduplicatives while capturing phonetic similarity as well and evaluate our models quantitatively.

6 Conclusion

We proposed generative models for blends, clippings, and reduplicatives, three dominant word-formation phenomena in slang. Our models are distinguished by their simplicity, adept use of linguistic and structural constraints, easy to implement and yield state of the art performance.

Our work suggests several directions for future research. First, our blending model can be extended to handle relatively rare insertions, incorporate the language model and the length model in a unified reinforcement learning framework optimizing a joint reward. Second, we do not investigate the complementary problem of de-blending. Moreover, we note that our evaluation of blends is based on an assumed gold standard. It would be useful to also characterize our blending model based on a human evaluation. Third, the gap between the performance of our clipping model and the upper bound (by the oracle) opens up the question of developing more nuanced models for clipping perhaps using deeper linguistic cues like stress patterns. Fourth, our models do not incorporate relatively rare types of reduplicative formation (like *schm* reduplicatives) suggesting yet another direction for research. Yet another open question is whether a global model can effectively model all the above phenomena. Finally, in our work, we focus on only developing models for word formation in English slang. However, such word formation patterns are also evident in other languages (Štekauer et al., 2012) and it is an open question as to whether similar models generalize to other languages as well.

Finally, our work potentially enables the development of several applications some of which include brand name generators, and rich conversational agents that are not only passive agents but can actively contribute to the evolution of language varieties.

Altogether our work has implications for the broader fields of Internet Linguistics and natural language understanding especially in the context of slang formation.

References

John Algeo. 1977. Blends, a structural and systemic view. *American speech* 52(1/2):47–64.

Joan Beal. 1991. Toy boys and lager louts: Motivation by linguistic form. In *Language usage and description: Studies presented to NE Osselton on the occasion of his retirement*. pages 139–148.

Natalia Beliaeva. 2014. Unpacking contemporary english blends: Morphological structure, meaning, processing .

Rachel Broad, Brandon Prickett, Elliott Moreton, Katya Pertsova, and Jennifer L Smith. 2016. Emergent faithfulness to proper nouns in novel english blends. In *Proceedings of WCCFL*. volume 33, pages 77–87.

Aliya Deri and Kevin Knight. 2015. How to make a frenemy: Multitape fsts for portmanteau generation.

Connie Eble. 2012. *Slang and sociability: In-group language among college students*. UNC Press Books.

David Eppstein. 1998. Finding the k shortest paths. *SIAM Journal on computing* 28(2):652–673.

Ingrid Fandrych. 2008. Submorphemic elements in the formation of acronyms, blends and clippings. *Lexis. Journal in English Lexicology* (2).

Varun Gangal, Harsh Jhamtani, Graham Neubig, Eduard Hovy, and Eric Nyberg. 2017. Charmanteau: Character embedding models for portmanteau creation. *arXiv preprint arXiv:1707.01176* .

Alex Graves et al. 2012. *Supervised sequence labelling with recurrent neural networks*, volume 385. Springer.

Stefan Th Gries. 2004a. Isnt that fantabulous? how similarity motivates intentional morphological blends in english. *Language, culture, and mind* pages 415–428.

Stefan Th Gries. 2004b. Shouldn't it be breakfunch? a quantitative analysis of blend structure in english. *Linguistics* pages 639–668.

Camiel Hamans. 1997. Clippings in modern french, english, german and dutch. *Trends in Linguistic Studies and Monographs* 101:1733–1742.

Ben Hixon, Eric Schneider, and Susan L Epstein. 2011. Phonemic similarity metrics to compare pronunciation methods. In *Twelfth Annual Conference of the International Speech Communication Association*.

Josef Hladky. 1998. Notes on reduplicative words in english. *Brno studies in English* 24:33–78.

Denis Jamet. 2009. A morphophonological approach to clipping in english. can the study of clipping be formalized? *Lexis. Journal in English Lexicology* (HS 1).

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

William Labov. 1972. Some principles of linguistic methodology. *Language in society* 1(1):97–120.

Elisa Mattiello. 2005. The pervasiveness of slang in standard and non-standard english .

Elisa Mattiello. 2008. *An introduction to English slang: A description of its morphology, semantics and sociology*, volume 2. Polimetrica sas.

Elisa Mattiello. 2013. *Extra-grammatical morphology in English: abbreviations, blends, reduplicatives, and related phenomena*, volume 82. Walter de Gruyter.

L Merlini Barbaresi. 2008. Extra-grammatical morphology: English reduplicatives. *Words in action. Diachronic and synchronic approaches to English discourse. Studies in honour of Ermanno Barisone* pages 228–241.

Rio Rini Diah Moehkardi. 2016. Patterns and meanings of english words through word formation processes of acronyms, clipping, compound and blending found in internet-based media. *Humaniora* 28(3):207–221.

Tadao Murata. 1990. Ab type onomatopes and reduplicatives in english and japanese. *Linguistic fiesta: festschrift for Professor Hisao Kakehis sixtieth birthday* pages 257–272.

Gözde Özbal and Carlo Strapparava. 2012. A computational approach to the automation of creative naming. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics, pages 703–711.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. *arXiv preprint arXiv:1604.05529* .

Vincent Renner, François Maniez, and Pierre Arnaud. 2013. *Cross-disciplinary perspectives on lexical blending*, volume 252. Walter De Gruyter.

Katherine E Shaw, Andrew M White, Elliott Moreton, and Fabian Monrose. 2014. Emergent faithfulness to morphological and semantic heads in lexical blends. In *Proceedings of the Annual Meetings on Phonology*. volume 1.

Michael R Smith, Ryan S Hintze, and Dan Ventura. 2014. Nehovah: A neologism creator nomen ipsum.

Pavol Štekauer, Salvador Valera, and Lívia Kőrtvélyessy. 2012. *Word-formation in the world's languages: A typological survey*. Cambridge University Press.

Nils Thun. 1963. *Reduplicative words in English: A study of formations of the types tick-tick, hurly-burly, and shilly-shally*. Uppsala.

Friedrich Ungerer. 2007. Word-formation. In *The Oxford handbook of cognitive linguistics*.

Peilu Wang, Yao Qian, Frank K Soong, Lei He, and Hai Zhao. 2015. Part-of-speech tagging with bidirectional long short-term memory recurrent neural network. *arXiv preprint arXiv:1510.06168* .

Kaisheng Yao and Geoffrey Zweig. 2015. Sequence-to-sequence neural net models for grapheme-to-phoneme conversion. *arXiv preprint arXiv:1506.00196* .

Jin Y Yen. 1971. Finding the k shortest loopless paths in a network. *management Science* 17(11):712–716.

Using Morphological Knowledge in Open-Vocabulary Neural Language Models

Austin Matthews and **Graham Neubig**
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA, USA
{austinma,gneubig}@cs.cmu.edu

Chris Dyer
DeepMind
London, UK
cdyer@google.com

Abstract

Languages with productive morphology pose problems for language models that generate words from a fixed vocabulary. Although character-based models allow any possible word type to be generated, they are linguistically naïve: they must discover that words exist and are delimited by spaces—basic linguistic facts that are built in to the structure of word-based models. We introduce an open-vocabulary language model that incorporates more sophisticated linguistic knowledge by predicting words using a mixture of three generative processes: (1) by generating words as a sequence of characters, (2) by directly generating full word forms, and (3) by generating words as a sequence of morphemes that are combined using a hand-written morphological analyzer. Experiments on Finnish, Turkish, and Russian show that our model outperforms character sequence models and other strong baselines on intrinsic and extrinsic measures. Furthermore, we show that our model learns to exploit morphological knowledge encoded in the analyzer, and, as a byproduct, it can perform effective unsupervised morphological disambiguation.

1 Introduction

Language modelling of morphologically rich languages is particularly challenging due to the vast set of potential word forms and the sparsity with which they appear in corpora. Traditional *closed vocabulary* models are unable to produce word forms unseen in training data and unable to generalize sub-word patterns found in data.

The most straightforward solution is to treat language as a sequence of characters (Sutskever et al., 2011). However, models that operate at two levels—a character level and a word level—have better performance (Chung et al., 2016). Another solution is to use morphological information, which has shown benefits in non-neural models (Chahuneau et al., 2013). In this paper, we present a model that combines these approaches in a fully neural framework.

Our model incorporates explicit morphological knowledge (e.g. from a finite-state morphological analyzer/generator) into a neural language model, combining it with existing word- and character-level modelling techniques, in order to create a model capable of successfully modelling morphologically complex languages. In particular, our model achieves three desirable properties.

First, it conditions on all available (intra-sentential) context, allowing it, in principle, to capture long-range dependencies, such as that the verb agreement between "students" and "are" in the sentence "The students who studied the hardest are getting the highest grades". While traditional n-gram based language models lack this property, RNN-based language models fulfill it.

Second, it explicitly captures morphological variation, allowing sharing of information between variants of the same word. This allows faster, smoother training as well as improved predictive generalization. For example, if the model sees the phrase "gorped the ball" in data, it is able to infer that "gorping the ball" is also likely to be valid. Similarly, the model is capable of understanding that morphological consistency within noun phrases is important. For example in Russian, one might say *malen'kaya chërniya koshka* ("small black cat", nominative), or *malen'kuyu chërniyu koshku* (accusative), but *malen'kiy chërnuyu koshke* (mixing nominative, accusative and dative) would have much lower probability.

Third, the language model seamlessly handles out of vocabulary items and their morphological variants. For example, even if the word *Obama* was never seen in a Russian corpus, we expect *Ya dal eto prezidentu Obame* ("I gave it to presi-

Proceedings of NAACL-HLT 2018, pages 1435–1445
New Orleans, Louisiana, June 1 - 6, 2018.

dent Obama") to have higher probability using the dative *Obame* than *Ya dal eto prezidentu Obama*, which uses the nominative. The model can also learn to decline proper nouns, including OOVs. Here it can recognize that *dal* ("gave") requires a dative, and that nouns ending with "a" generally do not meet that requirement.

In order to capture these properties, our model combines two pieces: an alternative embedding module that uses sub-word information such as character and morpheme-level information, and a generation module that allows us to output words at the word, morpheme, or character-level. The embedding module allows for the model to share information between morphological variants of surface forms, and produce sensible word embeddings for tokens never seen during training. The generation model allows us to emit tokens never seen during training, either by combining a lemma and a sequence of affixes to create a novel surface form, or by directly spelling out the desired word character by character. We then demonstrate the effectiveness both intrinsically, showing reduced perplexity on several morphologically rich languages, and extrinsically on machine translation and morphological disambiguation tasks.

2 Multi-level RNNLMs

Recurrent neural network language models are composed of three parts: (a) an encoder, which turns a context word into a vector, (b) a recurrent backbone that turns a sequence of word vectors that represent the ordered sequence of context vectors into a single vector, and (c) a generator, which assigns a probability to each word that could follow the given context. RNNLMs often use the same process for (a) and (c), but there is no reason why these processes cannot be decoupled. For example, Kim et al. (2016) and Ling et al. (2015) compose character-level representations for their word encoder, but generate words using a softmax whose probabilities rely on inner products between the current context vector and type-specific word embeddings.

In our model both the word generator (§2.1) and the word encoder (§2.2) compute representations that leverage three different views of words: frequent words have their own parameters, words that can be analyzed/generated by an analyzer are represented in terms of sequences of abstract morphemes, and all words are represented as a sequence of characters.

2.1 Word generation mixture model

In typical RNNLMs the probability of the ith word in a sentence, w_i given the preceding words is computed by using an RNN to encode the context followed by a softmax:

$$\begin{aligned} p(w_i \mid \boldsymbol{w}_{<i}) &= p(w_i \mid \mathbf{h}_i = \boldsymbol{\varphi}_{\text{RNN}}(w_1, \ldots, w_{i-1})) \\ &= \text{softmax}(\mathbf{W}\mathbf{h}_i + \mathbf{b}) \end{aligned}$$

where $\boldsymbol{\varphi}_{\text{RNN}}$ is an RNN that reads a sequence of words and returns a fixed sized vector encoding, $\mathbf{W}$ is a weight matrix, and $\mathbf{b}$ is a bias.

In this work, we will use a mixture model over M different models for generating words in place of the single softmax over words (Miyamoto and Cho, 2016; Neubig and Dyer, 2016):

$$\begin{aligned} p(w_i \mid \mathbf{h}_i) &= \sum_{m_i=1}^{M} p(w_i, m_i \mid \mathbf{h}_i) \\ &= \sum_{m_i=1}^{M} p(m_i \mid \mathbf{h}_i) p(w_i \mid \mathbf{h}_i, m_i), \end{aligned}$$

where $m_i \in [1, M]$ indicates the model used to generate word w_i. To ensure tractability for training and inference, we assume that m_i is conditionally independent of all $\boldsymbol{m}_{<i}$, given the sequence of word forms $\boldsymbol{w}_{<i}$.

We use three ($M = 3$) component models: (1) directly sampling a word from a finite vocabulary ($m_i = \text{WORD}$), (2) generating a word as a sequence of characters ($m_i = \text{CHARS}$), and (3) generating as a sequence of (abstract) morphemes which are then stitched together using a hand-written morphological transducer that maps from abstract morpheme sequences to surface forms ($m_i = \text{MORPHS}$). Figure 1 illustrates the model components, and we describe in more detail here:

Word generator. Select a word by directly sampling from a multinomial distribution over surface form words. Here the vocabulary is the $|V_w|$ most common full-form words seen during training. All less frequent words are assigned zero probability by this model, and must be generated by one of the remaining models.

Character sequence generator. Generate a word as a sequence of characters. Each character is predicted conditioned on the LM hidden state $\mathbf{h}_i$ and the partial word generated so far, encoded

with an RNN. The product of these probabilities is the total probability assigned to a full word form.

Morpheme sequence generator. Similarly to the character sequence generator, we can generate a word as a sequence of morphemes. We first generate a root r, followed by a sequence of affixes $a_1, a_2, \ldots$. For example the word "devours" might be generated as devour+3P+SG+EOW. Since multiple sequences of abstract morphemes may in general give rise to a single output form,[1] we marginalize these, i.e.,

$$p(w_i \mid \mathbf{h}_i, m_i = \text{MORPHS}) = \sum_{\boldsymbol{a}_i \in \{\boldsymbol{a} \mid \text{GEN}(\boldsymbol{a}) = w_i\}} p_{morphs}(\boldsymbol{a}_i \mid \mathbf{h}_i).$$

where $\text{GEN}(\boldsymbol{a})$ gives the surface word form produced from the morpheme sequence $\boldsymbol{a}$.

Due to the model's ability to produce output at the character level, it is able to produce any output sequence at all within the language's alphabet. This is critical as it allows the model to generate unknown words, such as novel names or declensions thereof. Furthermore, the morphological level facilitates the model's generation of word forms whose lemmas may be known, but whose surface form was nevertheless unattested in the training data. Finally the word-level generation model handles generating words that the model has seen many times during training.

2.2 Morphologically aware context vectors

Word vectors are typically learned with a single, independent vector for each word type. This independence means, for example, that the vectors for the word "apple" and the word "apples" are completely unrelated. Seeing the word "apple" gives no information at all about the word "apples".

Ideally we would like to share information between such related words. Nevertheless, sometimes words have idiomatic usage, so we'd like not to tie them together too tightly.

We accomplish this by again using three different types of word vectors for each word in the vocabulary. The first is a standard per-type word vector. The second is the output of a character-level RNN using Long Short-Term Memory (LSTM) units (Hochreiter and Schmidhuber, 1997). The third is the output of a morphology-level LSTM over a lemma and a sequence of affixes, as output by a morphological analyzer.

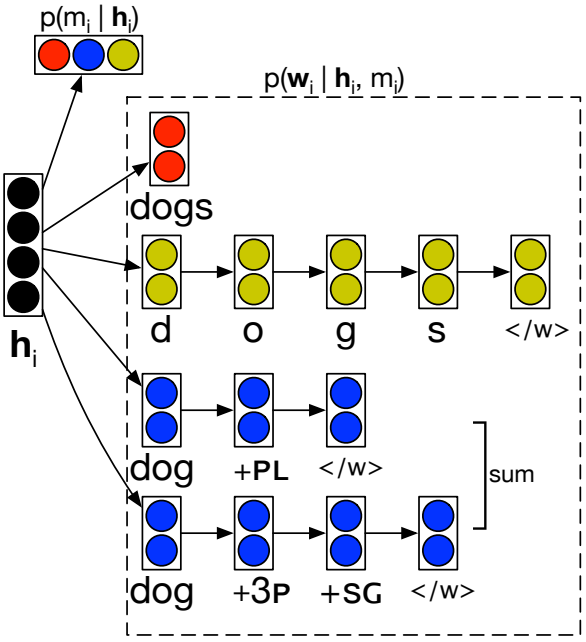


Figure 1: We allow the model to generate an output word at the word, morpheme, or character level, and marginalize over these three options to find the total probability of a word.

Typically language models first generate a word w_i given some (initially empty) prior context c_{i-1}, and then that word is combined with the context to generate a new context c_i that includes the new word. Since we have just used one or more of our three modes to generate each word, intuitively we would like to use the same mode(s) to generate the embedding used to progress the context.

Unfortunately, doing so introduces dependencies among the latent variables $p(\text{mode} \mid c_i)$ in our model, making exact inference intractable. As such, we instead drop the dependency on how a word was generated and instead represent the word at all three levels, regardless of the mode(s) actually used to generate it, and combine them by concatenating the three representations. A visual representation of the embedding process is shown in Figure 2.

Additionally, should a morphological analyzer produce more than one valid analysis for a surface form, we independently produce embeddings for each candidate analysis, and combine them using a per-dimension maximum operation. Mathemati-

[1] In general analyzers encode many-to-many relations, but our model assumes that any sequence of morphs in the underlying language generates a single surface form. This is generally true, although free spelling variants of a morph (e.g., American *-ize* vs. British *-ise* as well as alternative realizations like *shined/shone* and *learned/learnt*) violate this assumption.

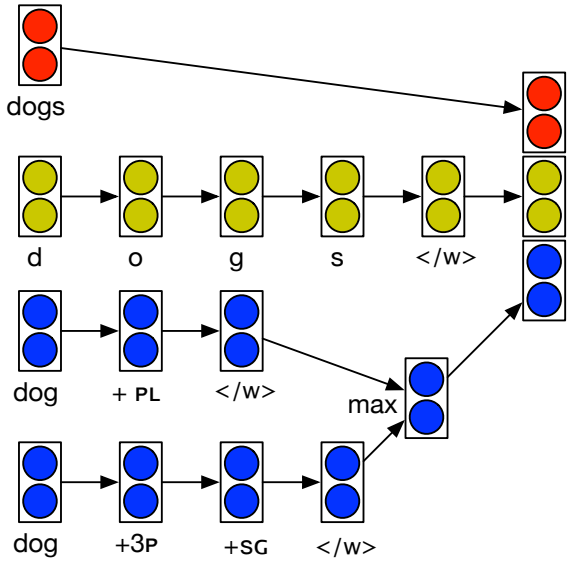


Figure 2: We concatenate word- morpheme- and character-level vectors to build a better input vector for our RNNLM.

cally, the ith dimension of the morphological embedding e_m is given by

$$e_{mi} = \max_j e_{a_j i}$$

where e_{a_j} is the embedding of the jth possible analysis, as computed by the LSTM over the lemma and its sequence of affixes.

The intuition behind the use of all analyses plus a pooling operation can be seen by observing the case of the word "does", which could be *do+3-person+singular* or *doe+plural*. If this word appears after the word "he", what we care about more is whether "does" could feasibly be a third person singular verb, thus agreeing with the subject. The max-pooling operation captures this intuition by ensuring that if a feature is active for one of these two analyses, it will also be active in the pooled representation. This procedure affords us the capability to efficiently marginalize over all three possible values of the latent variable at each step, and compute the full marginal of the word w_i given the context c_{i-1} during generation.

This formulation allows words with the same stem to share vector information through the character or morphological embeddings, but still affords each word the ability to capture idiomatic usages of individual words through the word embeddings. Furthermore, it allows a language model to explicitly capture morphological information, for example that third person singular subjects should co-occur with third person singular verbs. Finally, the character-level segment of the embedding allows the model to at least attempt to build sensible embeddings for completely unknown words. For example in Russian where names can decline with case this formulation allows the model to know that *Obame* is probably dative, even if it's an OOV at the word level, and even if the morphological analyzer is unable to produce any valid analyses.

We combine our three-layer input vectors, our factored output model, and a standard LSTM backbone to create a morphologically-enabled RNNLM that, as we will see in the next section, performs well on morphologically complex languages.

3 Intrinsic Evaluation

We demonstrate the effectiveness of our model by experimenting on three languages: Finnish, Turkish, and Russian. For Finnish we use version 8 of the Europarl corpus, for Turkish we use the SETIMES2 corpus, and for Russian we use version 12 of the News Commentary corpus. Statistics of our experimental corpora can be found in Table 1.

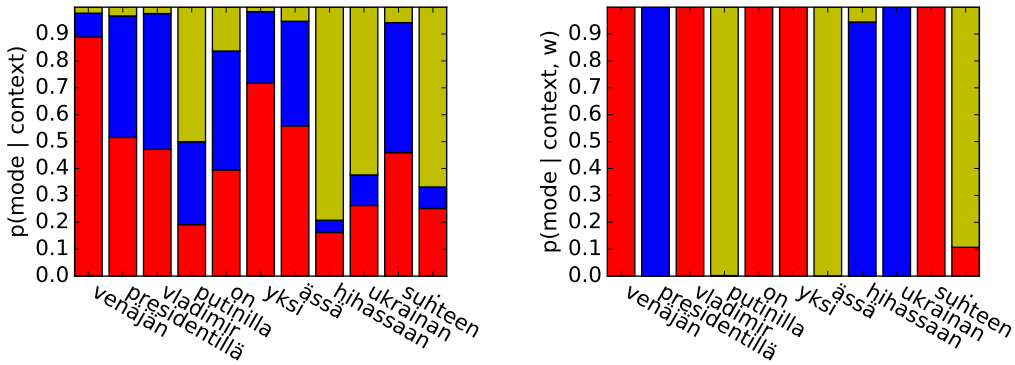
Each data set was pre-processed by UNKing all but the top ≈20k words and lemmas by frequency. No characters or affixes were UNKed. This step is not strictly required—our model is, after all, capable of producing arbitrary words— but it speeds up training immensely by reducing the size of the word and lemma softmaxes. Since the morphology and/or character-level embeddings can still capture information about the original forms of these words, the degradation in modelling performance is minimal.

For morphological analysis we use Omorfi[2] for Finnish, the analyzer of Oflazer (1994) for Turkish, and PyMorphy[3] for Russian.

3.1 Baseline Models

Since models are not accurately comparable unless they share output vocabularies, our baselines must also allow for the generation of arbitrary word forms, including out-of-vocabulary items. We compare to three such models: an improved Kneser-Ney (Kneser and Ney, 1995) 4-gram baseline, with an additional character-level backoff model for OOVs, an RNNLM with character-level backoff, and a pure character-based RNN language model (Sutskever et al., 2011).

[2] https://github.com/flammie/omorfi
[3] https://github.com/kmike/pymorphy



Finnish: "Russian President Vladimir Putin has an ace up his sleeve in the Ukrainian relationship."

Figure 3: An example of the priors (left) and posteriors (right) over modes used to generate each word in a sample sentences. Probability given to the word-, morpheme-, and character-level models are shown in red, blue, and gold respectively. More examples can be found in the appendix.

Since Kneser-Ney language models (and other count-based models) are typically word-level and do not model out-of-vocabulary items, we employ a two-level approach with separate Kneser-Ney models at the word and character levels. We train the word-level model after UNKing low frequency words, and we train the character-level model on the same list of low frequency words. Now when we want to predict a word w_i given some context c we can use the word-level model to directly predict $p(w_i|c)$ unless w_i is an out-of-vocabulary item. In that case we model $p(w_i \mid c)$ as

$$p(w_i \mid c) = p(\text{UNK} \mid c) \cdot p(w_i \mid \text{UNK})$$

where the first factor is the probability of the word-level model emitting UNK, and the second is the probability of the actual out-of-vocabulary word form under the character-level model.

Secondly we compare to a similar hybrid RNN model that first predicts the word-level probability for each word, and if it predicted UNK then also predicts a sequence of characters using a separate network. This model uses 256-dimensional character and word embeddings, and a 1024-dimensional recurrent hidden layer.

Finally we also compare to a standard RNN language model trained purely on the character level. For this baseline we also use 256-dimensional character embeddings and a 1024-dimensional recurrent hidden layer.

3.2 Multi-factored Models

For our model we use 128-dimensional word and root embeddings, 64-dimensional affix and character embeddings, 128-dimensional word-internal recurrent hidden layers for characters and morphemes, and a 256-dimensional recurrent hidden layer for the main inter-word LSTM.

The network is trained to stochastically optimize the log likelihood of the training data using Adam (Kingma and Ba, 2014). After each 10k training examples (Finnish, Turkish) or 100k training examples (Russian) we evaluate the model on a development set.[4] If the perplexity on the development set represents a new best, we save the current model to disk, thereby mitigating overfitting via early stopping. No other regularization is used.

For each language we run four variants of our model. In order to preserve the ability to model and emit any word in the modelled language, it is essential that we keep the character-level part of our model intact. The morpheme- and word-level models, however, may be removed without compromising the generality of the model. As such, we present our model using only character-level input and outputs (C), using character- and morpheme-level inputs and outputs (CM), using character- and word-level inputs, but no morphology (CW), and using all three levels as per the full model (CMW).

3.3 Results and Analysis

Our experimental results (Table 2) show that our multi-modal model significantly outperforms all three baselines: a naïve n-gram model, a purely character-level RNNLM, and a hybrid RNNLM

[4] We evaluate less frequently on Russian since the dev set is much larger.

	Finnish	Russian	Turkish
Train Sents	2.1M	1.1M	188K
Train Words	38M	26M	3.9M
Dev Sents	1K	38K	1K
Dev Words	16K	705K	16K
Test Sents	500	91K	3K
Test Words	7.6K	1.6M	51K
Word Vocab	20K	21K	42K
Lemma Vocab	20K	20K	13K
Affix Vocab	140	34	180
Char Vocab	229	150	80

Table 1: Details of our data sets. Each cell indicates the number of sentences and the number of words in each set.

for open-vocabulary language models. Furthermore, they confirm that morphological analyzers can improve performance of such language models on particularly morphologically rich languages. We observe that across all three languages the space-aware character-level model outperforms the purely character-based model that treats spaces just as any other character. Furthermore we see that the Kneser-Ney language model performs admirably well on this task, underscoring the difference in setting between the familiar, traditional closed-vocabulary LMs, and the open-vocabulary language modelling task. Additionally we find that the relative success of the n-gram model and the hybrid model over the character only models underscores the importance of access to word-level information, even when using a less sophisticated model.

Table 3 shows some examples of sentences on which our model heavily outperforms the RNN baseline and vice-versa. We find that the sentences on which our model peforms well contain much less frequent word forms. For each sentence we examine the frequency with which each token appears in our training corpus. The sentences on which our model performs best have a median token frequency of 305 times, while the sentences where the RNN performs better has an average token frequency of 3031 times. Overall our model has better log-likelihood than the RNN baseline on 88.1% of sentences.

Our methods outperform the n-gram model in all languages with either set of just two models, CM or CW. The same models outperform the hybrid baseline in Turkish and Russian, and achieves comparable results in Finnish. Finally, in the agglutinative languages, using all three modes performs best, while in Russian, a fusional language, characters and words alone edge out the model with morphology. We hypothesize that our morphology model is better able to model the long strings of morphemes found in Turkish and Finnish, but gains little from the more idiosyncratic fusional morphemes of Russian.

Some examples of the priors and posteriors of the modes used to generate some randomly selected sentences from the held out test set can be seen in Figure 3 and the appendix (Figure 4). The figures show that all of the models tend a priori to prefer to generate words directly when possible, but that context can certainly influence its priors. In Finnish, after seeing the word *Vladmir*, the model suddenly assigns significantly more weight to the character-level model to generate the following word, which is likely to be a surname. In Russian, after the preposition *o*, the following noun is required to be in a rare case. As such, the model suddenly assigns more probability mass to the following word being generated using the morpheme-level model.

The posteriors tell a similarly encouraging story. In Finnish we see that the word *presidentillä* is overwhelmingly likely to be produced by the morphology model due to its peculiar adessive ("at") case marking. *Vladmir* is common enough in the data that it can be generated wholly, but the last name *Putin* is again inflected into the adessive case, forming *Putinilla*. Unfortunately the morphological analyzer is unfamiliar with the stem *Putin*, forcing the word to be generated by the character-level model. In our Turkish example, all of the short words are generated at the word level, while the primary nouns *internetten* ("internet") and *ders* ("lecture") are possible to generate either as words or as a sequence of morphemes. The verb, which has much more complex morphology (progressive past tense with a third person singular agent), is generated via the morphological model.

4 Extrinsic Evaluation

In addition to evaluating our model intrinsically using perplexity, we evaluate it on two downstream tasks. The first is machine translation between English and Turkish. The second is Turkish morphological analysis disambiguation.

(a) Finnish

Model	Dev	Test
KN4+OOV	2.04	1.94
RNN+OOV	2.03	1.92
PureC	2.69	2.63
C	2.40	2.32
CM	1.95	1.85
CW	2.03	1.94
CWM	**1.91**	**1.81**

(b) Turkish

Model	Dev	Test
KN4+OOV	2.01	2.06
RNN+OOV	1.99	2.05
PureC	2.21	2.30
C	2.05	2.16
CM	1.88	1.99
CW	1.78	1.85
CWM	**1.74**	**1.82**

(c) Russian

Model	Dev	Test
KN4+OOV	1.68	1.70
RNN+OOV	1.62	1.66
PureC	1.91	2.05
C	1.85	1.87
CM	1.47	1.50
CW	**1.44**	**1.47**
CWM	1.49	1.52

Table 2: Intrinsic evaluation of language models for three morphologically rich languages. Entropy for each test set is given in bits per character on the tokenized data. Lower is better, with 0.0 being perfect.

Komünizm **peşinde koşan** Arnavut pek yok
Parlaklığını kaybeden **mücevher**: **Kirlilik Karadeniz'i** esir alıyor
Olayların **baskısıyla karşılaşan** rejim tutumunu **yavaşça yumuşattı**, 1991 yılında çok partili
→ seçimleri düzenledi ve sonunda da ertesi yıl **tümden** iktidarı bıraktı.

Southeast European Times için Belgrad'dan Dusan Kosanoviç'in haberi - 24/06/04
23 Temmuz'dan bu yana Balkanlar'la ilgili iş ve ekonomi haberlerine genel bakış:
AB'nin Genişlemeden Sorumlu Komisyon Üyesi Olli Rehn (solda) Arnavutluk Başbakanı Sali
→ Berişa ile 15 Mart Perşembe günü Tiran'da bir araya geldi.

Table 3: Some examples of Turkish sentence on which our morphological model heavily outperforms the baseline RNNLM (top) and some examples of the opposite (bottom). The sentences that our model performs well on have many particularly rare words, whereas the sentences the RNNLM performs well on were seen hundreds or thousands of times in the training corpus. Words in bold were seen fewer than 25 times in the training corpus. Arrows indicate line wrapping.

4.1 Machine Translation

As an extrinsic evaluation we test whether our language model improves machine translation between Turkish and English. While we could transform our model into a source-conditioned translation model, we choose here to focus on testing our model as an external unconditional language model, leaving the conditional version for future work. Since neural machine translation systems struggle with low-resource languages (Koehn and Knowles, 2017), we choose to introduce the score of our LM as an additional feature to a cdec (Dyer et al., 2010) hierarchical MT system. We train on the WMT 2016 Turkish–English data set, and perform n-best reranking after re-tuning weights with the new feature.

The results, shown in Table 4 demonstrate small but significant gains in both directions, particularly into Turkish, where modelling productive morphology should be more important.

Lang. Pair	System	BLEU
TR-EN	Baseline	15.0
	Morph. Input	**15.2**
EN-TR	Baseline	10.1
	Morph. Output	**10.5**

Table 4: Machine Translation Results

4.2 Morphological Disambiguation

Our model is a joint model over words and the latent processes giving rise to those words (i.e., which generation process was selected and, for the morpheme process, which morpheme sequence was generated). While our model is not directly trained to perform morphological disambiguation, it still performs this task quite admirably. Given a trained morphological language model, a sentence $\mathbf{s}$, and a set of morphological analyses $\mathbf{z}$, we can query the model to find $p(\mathbf{s}, \mathbf{z}) = p(w_1, w_2, \ldots, w_N)$ for a given sentence. Most notably, each w_i may have a set of possible morphological analyses $\{a_1, a_2, \ldots a_{M_i}\}$ from which we would like to choose the most likely *a posteriori*. To perform this task, we simply query the model M_i times, each time hiding all but the jth possible analysis from the model. We can then re-normalize the resulting probabilities to find $p(a_j|\mathbf{s})$ for each $j \in 1 \ldots M_i$.

To make training and inference with our model

Model	Supervised?	Ambiguous Words	All words
Random Chance	no	34.08%	52.66%
Unidirectional	no	55.15%	80.28%
Bidirectional	no	**63.85%**	**84.11%**
Shen et. al	yes	91.03%	96.43%

Table 5: Morphological disambiguation accuracy results for Turkish.

tractable, we have assumed independence between previous adjacent events and the next word generation given the previous surface word forms (§2.1). Thus, the posterior probability over the analysis is only determined by the left context—subsequent decisions are independent of the process used to generate a word at time t. However, since disambiguating information may be present in either direction, we introduce a model variant that conditions on information in both directions. Bidirectional dependencies mean that we can no longer use the chain rule to factor the probability distribution from left-to-right. Rather we have to switch to a globally normalized, undirected model (i.e., a Markov random field) to define the probabilities of selecting the mode of generation and generation probability (conditional on the mode). The factors used to parameterize the model are defined in terms of two LSTMs, one encoding from left-to-right the prefix of the ith word ($\mathbf{h}_i$, defined exactly as above), and a second encoding from right-to-left its suffix ($\mathbf{h}'_i$). These two vector representations are used to compute a score using a locally normalized mixture model for each word. Intuitively, the morphological analysis generated at each time step should be compatible with both the preceding words and the following words.

Optimizing this model with the same MLE criterion we used in the direct model is, unfortunately, intractable since a normalizer would need to be computed. Instead, we use a pseudo-likelihood objective (Besag, 1975).

$$\begin{aligned}\mathcal{L}_{\text{PL}} &= \prod_i p(w_i \mid \boldsymbol{w}_{-i}) \\ &= \prod_i \sum_m p(m_i = m \mid \boldsymbol{w}_{-i}) p(w_i \mid m, \boldsymbol{w}_{-i})\end{aligned}$$

We note that although this model has a very different semantics from the directed one, the PL training objective is identical to the directed model's, the only difference is that features are based both on the past and future, rather than only the past.

Similarly to training, evaluating sentence likelihoods using this model is intractable, but posterior inference over m_i and a_i is feasible since the normalization factors cancel and therefore do not need to be computed.

For our experiments we use the data set of Yuret and Türe (2006) who manually disambiguated from among the possible forms identified by an FST. We significantly out-perform the simple baseline of randomly guessing, and our results are competitive with Yatbaz and Yuret (2009), although they evaluated on a different dataset so they are not directly comparable. Furthermore, we also compare to a supervised model (Shen et al., 2016). While unsupervised techniques can't hope to exceed supervised accuracies, this comparison provides insight into the difficulty of the problem. See Table 5 for results.

5 Related Work

Purely Character-based or Subword-based LMs have a rich history going all the way back to Markov (1906)'s work modelling Russian character-by-character with his namesake models. More recently Sutskever et al. (2011) were the first to apply RNNs to character-level language modelling, leveraging their ability to handle the long-range dependencies required to model language at the character level. It is also possible to alleviate the closed vocabulary problem by training models on automatically acquired subword units (Mikolov et al., 2012; Sennrich et al., 2015). While these approaches allow for an open vocabulary (or nearly open, in the case of subwords) they discard a large amount of higher-level information, inhibiting learning.

Character-aware language models, which combine character- and word-level information have shown promise (Kang et al., 2011; Ling et al., 2015; Kim et al., 2016). Unsupervised morphology has also been shown to improve the representations used by a log-bilinear LM (Botha and Blunsom, 2014). Jozefowicz et al. (2016) explore many interesting such architectures, and compare with fully character-based models.

While these models allow for the elegant encoding of novel word forms they lack an open vocabulary.

Open-vocabulary hybrid models alleviate this problem, extending the benefits of character-level representations to the generation. Such hybrid models with open vocabularies have been around since Brown et al. (1992). More recently, Chung et al. (2016) and Hwang and Sung (2016) describe methods of modelling sentences at both the word and character levels, using mechanisms to allow both a word-internal model that captures short-range dependencies and a word-external model to capture longer-range dependencies. These models have been successfully applied to machine translation by Luong and Manning (2016), who use a character-level model to predict translations of out of vocabulary words. Our work falls in this category—we combine multiple representation levels while maintaining the ability to generate any character sequence. In contrast to these previous works, we demonstrate the utility of incorporating morphological information in these open-vocabulary models.

Mixture model language generation where the mixture coefficients are predicted by a neural net are becoming quite common. Neubig and Dyer (2016) use this strategy to combine a count-based model and a neural language model. Ling et al. (2016) interpolate between character- and word-based models to translate between natural language text and computer code. Merity et al. (2016) also use multiple output models, allowing a word to either be generated by a standard softmax or by copying a word from earlier in the input sentence.

6 Conclusion

We have demonstrated a technique for language modelling that works particularly well on morphologically rich languages where having an open vocabulary is desirable. We achieve this by using a multi-modal architecture that allows words to be input and output at the word, morpheme, or character levels. We show that knowledge of the existence of word boundaries is of critical importance for language modelling tasks, even when otherwise operating entirely at the character level, resulting in a surprisingly large reduction in per-character entropy across all languages studied.

Furthermore, we demonstrate that if we have access to a morphological analyzer we can leverage it to further improve our LM, reinforcing the notion that the explicit inclusion of linguistic information can indeed aid learning of neural models.

Acknowledgements

We would like to thank Sebastian Mielke for his insightful discussion and feedback on this work.

This work is sponsored by Defense Advanced Research Projects Agency Information Innovation Office (I2O). Program: Low Resource Languages for Emergent Incidents (LORELEI). Issued by DARPA/I2O under Contract No. HR0011-15-C0114. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Julian Besag. 1975. Statistical analysis of non-lattice data. *The statistician* pages 179–195.

Jan A Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *ICML*. pages 1899–1907.

Peter F Brown, Vincent J Della Pietra, Robert L Mercer, Stephen A Della Pietra, and Jennifer C Lai. 1992. An estimate of an upper bound for the entropy of english. *Computational Linguistics* 18(1):31–40.

Victor Chahuneau, Noah A Smith, and Chris Dyer. 2013. Knowledge-rich morphological priors for bayesian language models. Association for Computational Linguistics.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. 2016. Hierarchical multiscale recurrent neural networks. *arXiv preprint arXiv:1609.01704* .

Chris Dyer, Jonathan Weese, Hendra Setiawan, Adam Lopez, Ferhan Ture, Vladimir Eidelman, Juri Ganitkevitch, Phil Blunsom, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the ACL 2010 System Demonstrations*. Association for Computational Linguistics, pages 7–12.

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

Kyuyeon Hwang and Wonyong Sung. 2016. Character-level language modeling with hierarchical recurrent neural networks. *arXiv preprint arXiv:1609.03777* .

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410* .

Moonyoung Kang, Tim Ng, and Long Nguyen. 2011. Mandarin word-character hybrid-input neural network language model. In *INTERSPEECH*. pages 625–628.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*. pages 2741–2749.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*. IEEE, volume 1, pages 181–184.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. *arXiv preprint arXiv:1706.03872* .

Wang Ling, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, Andrew Senior, Fumin Wang, and Phil Blunsom. 2016. Latent predictor networks for code generation. *arXiv preprint arXiv:1603.06744* .

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096* .

Minh-Thang Luong and Christopher D Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. *arXiv preprint arXiv:1604.00788* .

Andrey Andreyevich Markov. 1906. Extension of the law of large numbers to dependent quantities. *Izv. Fiz.-Matem. Obsch. Kazan Univ.(2nd Ser)* 15:135–156.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843* .

Tomáš Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, and Stefan Kombrink. 2012. Subword language modeling with neural networks .

Yasumasa Miyamoto and Kyunghyun Cho. 2016. Gated word-character recurrent language model. *arXiv preprint arXiv:1606.01700* .

Graham Neubig and Chris Dyer. 2016. Generalizing and hybridizing count-based and neural language models. *arXiv preprint arXiv:1606.00499* .

Kemal Oflazer. 1994. Two-level description of turkish morphology. *Literary and linguistic computing* 9(2):137–148.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909* .

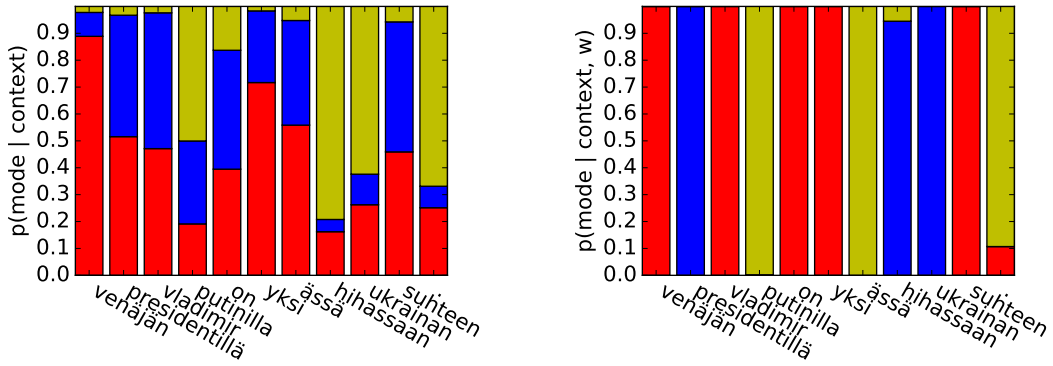
Qinlan Shen, Daniel Clothiaux, Emily Tagtow, Patrick Littell, and Chris Dyer. 2016. The role of context in neural morphological disambiguation .

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. pages 1017–1024.

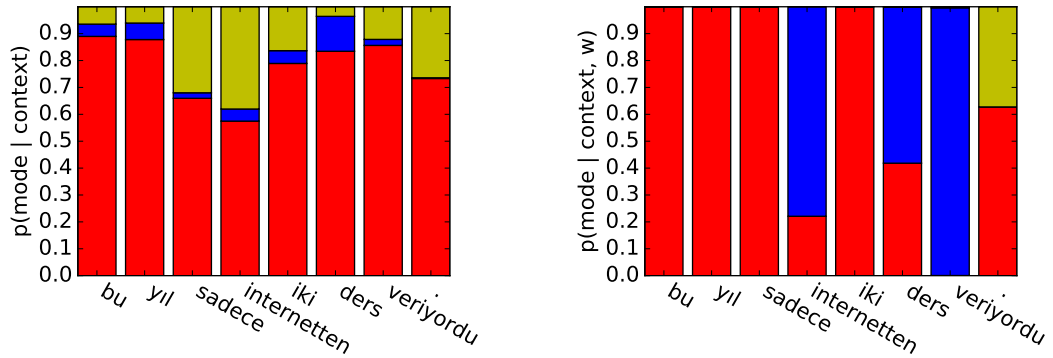
Mehmet Ali Yatbaz and Deniz Yuret. 2009. Unsupervised morphological disambiguation using statistical language models. In *Pro. of the NIPS 2009 Workshop on Grammar Induction, Representation of Language and Language Learning, Whistler, Canada*. pages 321–324.

Deniz Yuret and Ferhan Türe. 2006. Learning morphological disambiguation rules for turkish. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*. Association for Computational Linguistics, pages 328–334.

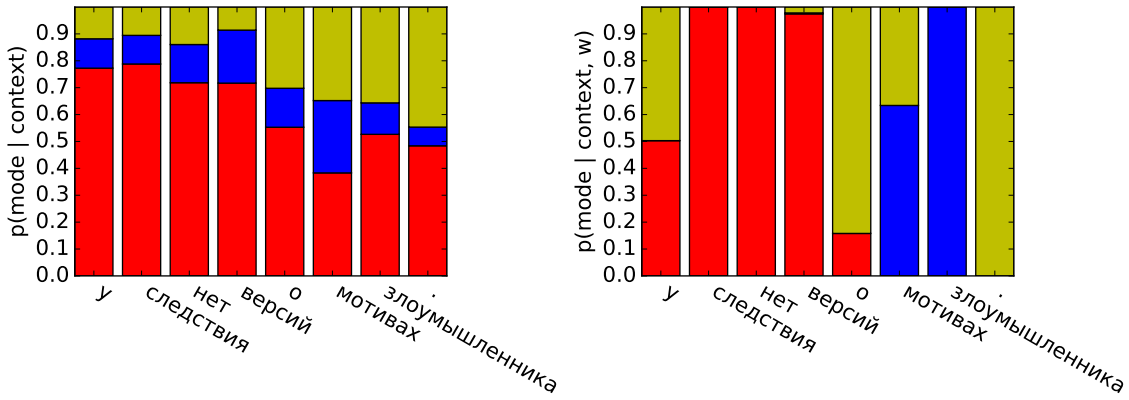
A Example sentences



Finnish: “Russian President Vladimir Putin has an ace up his sleeve in the Ukrainian relationship.”



Turkish: “He gave only two lectures on the internet this year.”



Russian: “The investigation does not theorize about the attacker’s motives.”

Figure 4: Some examples of the priors (left) and posteriors (right) over modes used to generate each word in some sample sentences. Probability given to the word-, morpheme-, and character-level models are shown in red, blue, and gold respectively. The Finnish example is a reprint of Figure 3

A Neural Layered Model for Nested Named Entity Recognition

Meizhi Ju[1,3], **Makoto Miwa**[2,3] and **Sophia Ananiadou**[1,3]
[1]National Centre for Text Mining, University of Manchester, United Kingdom
[2]Toyota Technological Institute, Japan
[3]Artificial Intelligence Research Center (AIRC),
National Institute of Advanced Industrial Science and Technology (AIST), Japan
{meizhi.ju, sophia.ananiadou}@manchester.ac.uk
makoto-miwa@toyota-ti.ac.jp

Abstract

Entity mentions embedded in longer entity mentions are referred to as nested entities. Most named entity recognition (NER) systems deal only with the flat entities and ignore the inner nested ones, which fails to capture finer-grained semantic information in underlying texts. To address this issue, we propose a novel neural model to identify nested entities by dynamically stacking flat NER layers. Each flat NER layer is based on the state-of-the-art flat NER model that captures sequential context representation with bidirectional long short-term memory (LSTM) layer and feeds it to the cascaded CRF layer. Our model merges the output of the LSTM layer in the current flat NER layer to build new representation for detected entities and subsequently feeds them into the next flat NER layer. This allows our model to extract outer entities by taking full advantage of information encoded in their corresponding inner entities, in an inside-to-outside way. Our model dynamically stacks the flat NER layers until no outer entities are extracted. Extensive evaluation shows that our dynamic model outperforms state-of-the-art feature-based systems on nested NER, achieving 74.7% and 72.2% on GENIA and ACE2005 datasets, respectively, in terms of F-score.[1]

1 Introduction

The task of named entity recognition (NER) involves the extraction from text of names of entities pertaining to semantic types such as *person (PER)*, *location (LOC)* and *geo-political entity (GPE)*. NER has drawn the attention of many researchers as the first step towards NLP applications such as entity linking (Gupta et al., 2017), relation extraction (Miwa and Bansal, 2016), event extraction (Feng et al., 2016) and co-reference resolution (Fragkou, 2017; Stone and Arora, 2017).

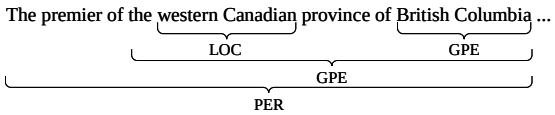


Figure 1: A sentence from ACE2005 (Walker et al., 2006) containing the nested 4 entities nested 3 levels deep.

Due to the properties of natural language, many named entities contain *nested entities*: embedded names which are included in other entities, illustrated in Figure 1. This phenomenon is quite common in many domains (Alex et al., 2007; Byrne, 2007; Wang, 2009; Màrquez et al., 2007). However, much of the work on NER copes only with non-nested entities which are also called *flat entities* and neglects nested entities. This leads to loss of potentially important information, with negative impacts on subsequent tasks.

Traditional approaches to NER mainly involve two types of approaches: supervised learning (Ling and Weld, 2012; Marcińczuk, 2015; Leaman and Lu, 2016) and hybrid approaches (Bhasuran et al., 2016; Rocktäschel et al., 2012; Leaman et al., 2015) that combine supervised learning with rules. Such approaches require either domain knowledge or heavy feature-engineering. Recent advances in neural networks enable NER without depending on external knowledge resources through automated learning high-level and abstract features from text (Lample et al., 2016; Ma and Hovy, 2016; Pahuja et al., 2017; Strubell et al., 2017).

In this paper, we propose a novel dynamic neural model for nested entity recognition, without relying on any external knowledge resources or linguistics features. Our model enables sequentially

[1]Code is available at https://github.com/meizhiju/layered-bilstm-crf

Proceedings of NAACL-HLT 2018, pages 1446–1459
New Orleans, Louisiana, June 1 - 6, 2018.

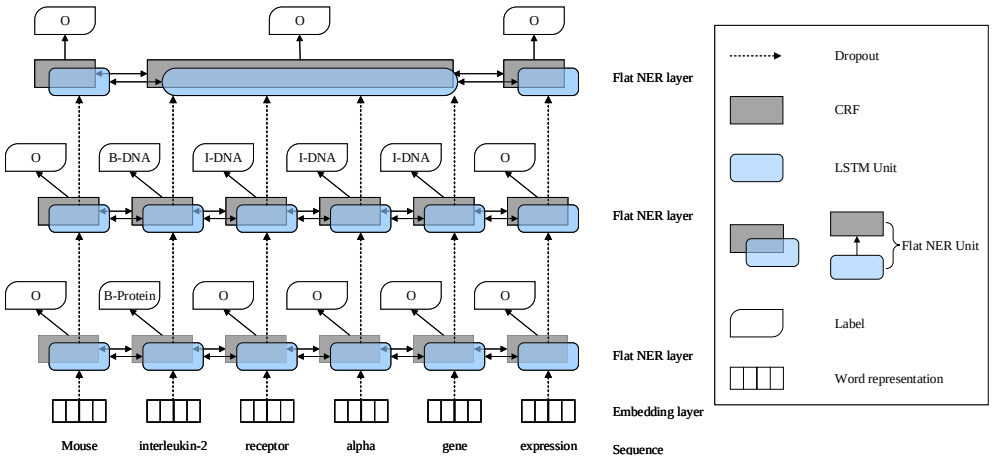


Figure 2: Overview of our layered model architecture. "interleukin-2" and "interleukin-2 receptor alpha gene" are nested entities.

stacking flat NER layers from bottom to up and identifying entities in an end-to-end manner. The number of stacked layers depends on the level of entity nesting and dynamically adjusts to the input sequences as the nested level varies from different sequences.

Given a sequence of words, our model first represents each word using a low-dimensional vector concatenated from its corresponding word and character sequence embeddings. Taking the sequence of the word representation as input, our flat NER layer enables capturing context representation by a long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) layer. The context representation is then fed to a CRF layer for label prediction. Subsequently, the context representation from the LSTM layer is merged to build representation for each detected entity, which is used as the input for the next flat NER layer. Our model stops detecting entities if no entities are predicted by the current flat NER layer. Through stacking flat NER layers in order, we are able to extract entities from inside to outside with sharing parameters among the different LSTM layers and CRF layers.

We gain 3.9 and 9.1 percentage point improvements regarding F-score over the state-of-the-art feature-based model on two nested entity corpora: GENIA (Kim et al., 2003) and ACE2005 (Walker et al., 2006), and analyze contributions of inner entities to outer entity detection, drawing several key conclusions.

In addition, experiments are conducted on a flatly annotated corpora JNLPBA (Kim et al., 2004). Our model can be a complete NER model as well for flat entities, on the condition that it is trained on annotations that do not account for nested entities. We obtain 75.55% in terms of F-score that is comparable to the state-of-the-art performance.

2 Neural Layered Model

Our nested NER model is designed based on a sequential stack of flat NER layers that detects nested entities in an end-to-end manner. Figure 2 provides the overview of our model. Our flat NER layers are inspired by the state-of-the-art model proposed in Lample et al. (2016). The layer utilizes one single bidirectional LSTM layer to represent word sequences and predict flat entities by putting one single CRF layer on top of the LSTM layer. Therefore, we refer to our model as Layered-BiLSTM-CRF model. If any entities are predicted, a new flat NER layer is introduced and the word sequence representation of each detected entity by the current flat NER layer is merged to compose a representation for the entity, which is then passed on to the new flat NER layer as its input. Otherwise, the model terminates stacking and hence finishes entity detection.

In this section, we provide a brief description of the model architecture: the flat NER layers and their stacking, the embedding layer and their training.

2.1 Flat NER layer

A flat NER layer consists of an LSTM layer and a CRF layer. The LSTM layer captures the bidi-

rectional context representation of sequences and subsequently feeds it to the CRF layer to globally decode label sequences.

LSTM is a variant of recurrent neural networks (RNNs) (Goller and Kuchler, 1996) that incorporates a memory cell to remember the past information for a long period of time. This enables capturing long dependencies, thus reducing the gradient vanishing/explosion problem existing in RNNs. We employ bidirectional LSTM with no peephole connection. We refer the readers to Hochreiter and Schmidhuber (1997) for more details of LSTM used in our work.

CRFs are used to globally predict label sequences for any given sequences. Given an input sequence $X = (x_1, x_2, \ldots, x_n)$ which is the output from the LSTM layer, we maximize the log-probability during training. In decoding, we set transition costs between illegal transitions, e.g., transition from O to I-PER, as infinite to restrict illegal labels. The expected label sequence $y = (y_1, y_2, \ldots, y_n)$ is predicted based on maximum scores in decoding.

2.2 Stacking flat NER layers

We stack a flat NER layer on the top of the current flat NER layer, aiming to extract outer entities. Concretely, we merge and average current context representation of the regions composed in the detected entities, as described in the following equation:

$$m_i = \frac{1}{end - start + 1} \sum_{i=start}^{end} z_i, \qquad (1)$$

where z_i denotes the representation of the i-th word from the flat NER layer, and m_i is the merged representation for an entity. The region starts from a position $start$ and ends at a position end of the sequence. This merged representation of detected entities allows us to treat each detected entity as a single token, and hence we are able to make the most of inner entity information to encourage outer entity recognition. If the region is detected as a non-entity, we keep the representation without any processing. The processed context representation of the flat NER layer is used as the input for the next flat NER layer.

2.3 Embedding layer

The input for the first NER layer is different from the remaining flat NER layers since the first layer has no previous layers. We thus represent each word by concatenating character sequence embeddings and word embeddings for the first flat NER layer. Figure 3 describes the architecture of the embedding layer to produce word representation.

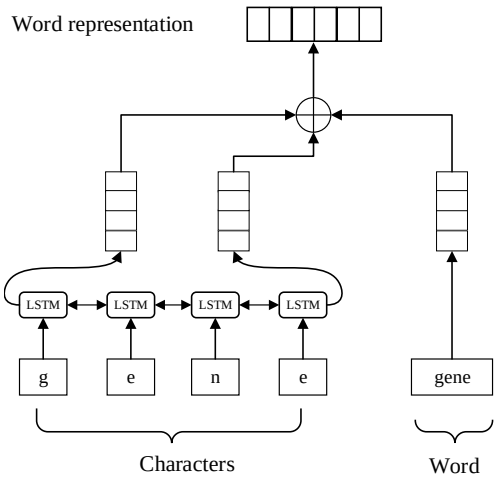


Figure 3: Word representation of a word 'gene'. We concatenate the outputs of character embedding from LSTM and word embedding to obtain its final word representation.

Following the successes of Ma and Hovy (2016) and Lample et al. (2016) in utilizing character embeddings on the flat NER task, we also represent each word with its character sequence to capture the orthographic and morphological features of the word. Each character is mapped to a randomly initialized vector through a character lookup table. We feed the character vectors comprising a word to a bidirectional LSTM layer and concatenate the forward and backward representation to obtain the word-level embedding.

Differently from the character sequence embeddings, the pretrained word embeddings are used to initialize word embeddings. When evaluating or applying the model, words that are outside of the pretrained embeddings and training dataset are mapped to an *unknown* (UNK) embedding, which is randomly initialized during training. To train the UNK embedding, we replace words whose frequency is 1 in the training dataset with the UNK embedding with a probability 0.5.

2.4 Training

We prepare the gold labels based on the conventional BIO (Beginning, Inside, Out of entities) tagging scheme to represent a label attached to each word.

As our model detects entities from inside to outside, we keep the same order in preparing the gold labels for each word sequence. We call it the *detection order rule*. Meantime, we define that each entity region in the sequence can only be tagged once with the same entity type, referred to as the *non-duplicate rule*. For instance, in Figure 2, "interleukin-2" is tagged first while "interleukin-2 receptor alpha gene" is subsequently tagged following the above two rules. When assigning the label *O* to non-entity regions, we only follow the detection order rule. As a result, two gold label sequences {O, B-Protein, O, O, O, O} and {O, B-DNA, I-DNA, I-DNA, I-DNA, O} are assigned to the given word sequence "Mouse interleukin-2 receptor alpha gene expression" as shown in Figure 2. With these rules, the number of labels for each word equals the nested level of entities in the given word sequence.

We employ mini-batch training and update the model parameters using back-propagation through time (BPTT) (Werbos, 1990) with Adam (Kingma and Ba, 2014). The model parameters include weights, bias, transition costs, and embeddings of characters. We disable updating the word embeddings.[2] During training, early stopping, L2-regularization and dropout (Hinton et al., 2012) are used to prevent overfitting. Dropout is employed to the input of each flat NER layer. Hyperparameters including batch size, number of hidden units in LSTM, character dimensions, dropout rate, Adam learning rate, gradient clipping and weight decay (L2) are all tuned with Bayesian optimization (Snoek et al., 2012).

3 Evaluation Settings

We employed three datasets for evaluation: GENIA[3] (Kim et al., 2003), ACE2005[4] (Walker et al., 2006) and JNLPBA[5] (Kim et al., 2004). We briefly explain the data and task settings and then introduce model and experimental settings.

3.1 Data and Task Settings

We performed nested entity extraction experiments on GENIA and ACE2005 while we conducted flat entity extraction on the JNLPBA dataset. For the details of data statistics and preprocessing, please refer to the supplementary materials.

GENIA involves 36 fine-grained entity categories among total 2,000 MEDLINE abstracts. Following the same task settings as in Finkel and Manning (2009) and Lu and Roth (2015), we collapsed all DNA subcategories as DNA. The same setting was applied to RNA, protein, cell line and cell type categories. We used same test portion as Finkel and Manning (2009), Lu and Roth (2015) and Muis and Lu (2017) for the direct comparison.

ACE2005 contains 7 fine-grained entity categories. We made same modifications described in Lu and Roth (2015) and Muis and Lu (2017) by keeping files from bn, bw, nw and wl and spitting them into training, development and testing datasets at random following same ratio 8:1:1, respectively.

JNLPBA defines both training and testing datasets. These two datasets are composed of 2,000 and 404 MEDLINE abstracts, respectively. JNLPBA is originally from the GENIA corpus. However, only flat and topmost entities in JNLPBA are kept while nested and discontinuous entities are removed. Like our preprocessing on the GENIA corpus, subcategories are collapsed and only 5 entity types are finally reserved. We randomly chose the 90% sentences of the original training dataset as our training dataset and the remaining as our development dataset.

Precision (P), recall (R) and F-score (F) were used for the evaluation metrics in our tasks. We define that if the numbers of gold entities and predictions are all zeros, the evaluation metrics all equal one hundred percent.

3.2 Model and Experimental Settings

Our model was implemented with Chainer[6] (Tokui et al., 2015). We initialized word embeddings in GENIA and JNLPBA with the pretrained embeddings trained on MEDLINE abstracts (Chiu et al., 2016). For ACE2005, we initialized each word with the pretrained embeddings which are trained by Miwa and Bansal (2016). Except for the word embeddings, parameters of word embeddings were initialized with a normal distribution. For LSTM, we initialized hidden states, cell state and all the bias terms as 0 except for the forget gate

[2] We tried updating and disabling updating word embeddings. The former trial did not work.

[3] http://www.geniaproject.org/genia-corpus/term-corpus

[4] https://catalog.ldc.upenn.edu/ldc2006t06

[5] http://www.nactem.ac.uk/tsujii/GENIA/ERtask/report.html

[6] https://chainer.org/

bias that was set as 1. For other hyper-parameters, we chose the best hyper-parameters via Bayesian optimization. We refer the readers to the supplemental material for the settings of the hyper-parameters of the models and Bayesian optimization.

For ablation tests, we compared with our layered-BiLSTM-CRF model with two models that produce the input for next flat NER layer in different ways. The first model is called *layered-BiLSTM-CRF w/o layered out-of-entities* which uses the input of the current flat NER layer for out-of-entity words. We name the second model as *layered-BiLSTM-CRF w/o layered LSTM* as it skips all intermediate LSTM layers and only uses output of embedding layer to build the input for the next flat NER layer. Please refer to supplemental material for the introduced two models.[7]

To investigate the effectiveness of our model on different nested levels of entities, we evaluated the model performance on each flat NER layer on GENIA and ACE2005 test datasets.[8] When calculating precision and recall measurements, we collected the predictions and gold entities from the corresponding flat NER layer. Since predicted entities on a specific flat NER layer might be from other flat NER layers, we defined extended precision (EP), extended recall (ER) and extended F-score (EF) to measure the performance. We calculated EP by comparing the predicted entities in a specific flat NER layer with all the gold entities, and ER by comparing the gold entities in a specific flat NER layer with all the predicted entities. EF was calculated in the same way with F.

In addition to experiments on nested GENIA and ACE2005 datasets, flat entity recognition was conducted on the JNLPBA dataset. We trained our flat model that only kept the first flat NER layer and removed the following stacking layers. We follow the hyper-parameters settings by Lample et al. (2016) for this evaluation.

[7]We examined the contributions of predicted labels of the current flat NER layer to the next flat NER layer. For this, we introduced label embeddings into each test by combining the embedding with context representation. Experiments show that appending label embedding hurts the performance of our model while gain slight improvements in the rest 2 models on development datasets.

[8]We removed entities which were predicted in previous flat NER layers during evaluation.

4 Results and Analysis

4.1 Nested NER

Table 1 presents the comparisons of our model with related work including the state-of-the-art feature-based model by Muis and Lu (2017). Our model outperforms the state-of-the-art models with 74.7% and 72.2% in terms of F-score, achieving the new state-of-the-art in the nested NER tasks. For GENIA, our model gained more improvement in terms of recall with enabling extract more nested entities without reducing precision. On ACE2005, we improved recall with 12.2 percentage points and obtained 5.1% relative error reductions. Compared with GENIA, our model gained more improvements in ACE2005 in terms of F-score. Two possible reasons account for it. One reason is that ACE2005 contains more deeper nested entities (maximum nested level is 5) than GENIA (maximum nested level is 3) on the test dataset. This allows our model to capture the potentially 'nested' relations among nested entities. The other reason is that ACE2005 has more nested entities (37.45%) compared with GENIA (21.56%).

Table 2 shows the results of models on the development datasets of GENIA and ACE2005, respectively. From this table, we can see that our model, which only utilizes context representation for preparation of input for the next flat NER layer, performs better than the rest two models. This demonstrates that introducing input of the current flat NER layer such as skipping either representation for any non-entity or words or all intermediate LSTM layers hurts performance. Compared with the layered-BiLSTM-CRF model, the drop of the performance in the layered-BiLSTM-CRF w/o layered out-of-entities model reflects the skip of representation for out-of-entity words leads to the decline in performance. This is because the representation of non-entity words didn't incorporate the current context representation as we used the input rather than the output to represent them. By analogy, the layered BiLSTM-CRF w/o layer LSTM model skips representation for both entities and non-entity words, resulting in worse performance. This is because, when skipping all intermediate LSTM layers, input of the first flat NER layer, i.e., word embeddings, is passed to the remaining flat NER layers. Since word embeddings do not contain context representation, we fail to incorporate the context representation when we use

Settings	GENIA			ACE2005		
	P (%)	R (%)	F (%)	P (%)	R (%)	F (%)
Finkel and Manning (2009)	75.4	65.9	70.3	-	-	-
Lu and Roth (2015)	72.5	65.2	68.7	66.3	59.2	62.5
Muis and Lu (2017)	75.4	66.8	70.8	69.1	58.1	63.1
Our model	**78.5**	**71.3**	**74.7**	**74.2**	**70.3**	**72.2**

Table 1: Comparisons of our model with the state-of-the-art models on nested NER.

Settings	GENIA			ACE2005		
	P (%)	R (%)	F (%)	P (%)	R (%)	F (%)
Layered-BiLSTM-CRF	78.27	75.97	77.10	75.37	69.41	72.27
Layered-BiLSTM-CRF w/o layered non-entities	76.55	77.01	76.78	72.90	65.54	69.02
Layered-BiLSTM-CRF w/o layered LSTM	75.76	74.60	75.18	69.94	61.94	65.70

Table 2: Performances of ablation tests on development datasets.

Entity type	P (%)	R (%)	F (%)
DNA	74.43	69.68	71.98
RNA	90.29	79.48	84.54
Protein	80.48	73.20	76.67
Cell Line	77.83	65.65	71.22
Cell Type	76.36	68.07	71.97
Overall	78.59	71.33	74.79

Table 3: Results of all entities for each type in GENIA test dataset.

Entity type	P (%)	R (%)	F (%)
PER	78.82	77.37	78.09
LOC	54.54	43.47	48.38
ORG	63.25	54.20	58.38
GPE	76.92	78.98	77.94
VEH	61.53	48.48	54.23
WEA	66.66	53.73	59.50
FAC	49.19	35.26	41.07
Overall	74.27	70.34	72.25

Table 4: Results of all entities for each type in ACE2005 test dataset.

the word embeddings as the input for the flat NER layers. Therefore, we have no chance to take advantage of the context representation and instead we only manage to use the word embeddings as the input for flat NER layers in this case.

Table 3 and Table 4 describe the performance for each entity type in GENIA and ACE2005 test datasets, respectively. In GENIA, our model performed best in recognizing entities with type *RNA*. This is because most of the entities pertaining to *RNA* mainly end up either with "mRNA" or *RNA*. These two words are informative indicators of *RNA* entities. For entities in rest entity types, their performances are close to the overall performance. One possible reason is that there are many instances to model them. This also accounts for the high performances of entity types such as *PER*, *GPE* in ACE2005. The small amounts of instances of entity types like *FAC* in ACE2005 is one reason for their under overall performances. We refer readers to supplemental material for statistics details.

When evaluating our model on top level which contains only outermost entities, the precision, recall and F-score were 78.19%, 75.17% and 76.65% on GENIA test dataset. For ACE2005, the corresponding precision, recall and F-score were 68.37%, 68.57% and 68.47%. Compared with the overall performance listed in Table 1, we obtained higher top level performance on GENIA but lower performance in ACE2005. We discuss details of this phenomena in the following tables.

Table 5 shows the performances of each flat NER layer in GENIA test dataset. Among all the stacking flat NER layers, our model resulted in the best performance regarding standard evaluation metrics on the first flat NER layer which contains the predictions for the gold innermost entities. When the model went to deeper flat NER layers, the performance dropped gradually as the number of gold entities decreased. However, the performance for predictions on each flat

Layer	P (%)	R (%)	F (%)	EP (%)	ER (%)	EF (%)	#Predictions	#Gold Entities
Layer 1	72.86	69.82	71.31	78.46	71.06	74.57	4,783	4,991
Layer 2	56.88	27.59	37.15	81.15	73.98	77.39	276	569
Layer 3	0.00	0.00	0.00	0.00	60.00	0.00	1	15

Table 5: Results of layer evaluation on GENIA test dataset.

Layer	P (%)	R (%)	F (%)	EP (%)	ER (%)	EF (%)	#Predictions	#Gold Entities
Layer 1	74.46	73.39	73.92	75.84	73.77	74.79	2,894	2,936
Layer 2	60.28	50.49	54.95	66.19	58.41	62.05	423	505
Layer 3	51.02	24.51	33.11	51.02	37.25	43.06	49	102
Layer 4	0.00	0.00	0.00	0.00	10.00	0.00	0	10
Layer 5	0.00	0.00	0.00	0.00	0.00	0.00	0	1

Table 6: Results of layer evaluation on ACE2005 test dataset.

NER layer was different in terms of extended evaluation metrics. For the first two flat NER layers, performance of extended evaluation is better than the performance of standard evaluation. It indicates that gold entities correspond to some of the predictions on the specific flat NER layer are from other flat NER layers. This may lead to the zero performances for the last flat NER layer. In addition, performance on the second flat NER layer was higher than it was on the first flat NER layer in terms of extended F-score. This demonstrates that our model is able to obtain higher performance on top level of entities than innermost entities.

Table 6 lists the results of each flat NER layer on ACE2005 test dataset. Similar to GENIA, the first flat NER layer achieved better performance than the rest flat NER layers. Performances decreased in a bottom-to-up manner regarding model architecture. This phenomena was the same with the extended evaluation performances, which reflects that some of the predictions in a specific flat NER layer were detected in other flat NER layers. Unlike rising tendency (except last flat NER layer) regarding extend F-score in GENIA, performance in ACE2005 was in downtrend. This accounts for the fact that F-score on top level was lower than it on the fist flat NER layer. Even though the decline trend in extended F-score, the first flat NER layer contained the largest proportion of predictions for the gold entities, the overall performance on all nested entities showed in Table 1 was still high. Unlike GENIA, our model in ACE2005 stopped before reaching the maximum nested level of entities. It indicates our model failed to model the appropriate nested levels. This is one of the reasons that account for the zero predictions on the last flat NER layer. One reason is that our model The sparse instances on the high nested levels could be another reason that resulted in the zero performances on the last flat NER layer.

4.2 Flat NER

Compared with the state-of-the-art work on JNLPBA (Gridach, 2017) which achieved 75.87% in terms of F-score, our model obtained 75.55% in F-score. Since both the model by Gridach (2017) and our flat model are based on Lample et al. (2016), so it is reasonable that both models were able to get comparable performance.

4.3 Error analysis

We showed the error types and their statistics both for all nested entities and each flat NER layer on GENIA and ACE2005 test datasets. From ACE2005 test dataset, 28% of predictions were incorrect in 200 sentences which were selected at random. Among these errors, 39% of them were because their text spans were assigned with other entity types. We call this type of errors *type error*. The main reason is that most of them are pronouns and co-refer to other entities which are absent in the sentence. Taking this sentence "whether that is true now, we can not say" as an example, "we" is annotated as *ORG* while our model labeled it as *PER*. Lack of context information such as the absence of co-referent entities leads our model to make the wrong decisions. In addition, 30% of the errors were caused by that incorrect predictions were predicted as only parts of gold entities with correct entity types. This error type is referred to as *partial prediction error*. This might be due to these gold entities tend to clauses or inde-

pendent sentences, thus possibly containing many modifiers. For example, in this sentence "A man who has been to Baghdad many times and can tell us with great knowledge exactly what it's going to be like to fight on those avenues in that sprawling city of Baghdad - Judy .", "A man who has been to Baghdad many times and can tell us with great knowledge exactly what it's going to be like to fight on those avenues in that sprawling city of Baghdad" is annotated as *PER* while our model could only extract "A man who has been to Baghdad many times" and predicted it as *PER*.

Errors on the first flat NER layer, we got 41% in type error and 11% of partial prediction error, respectively. Apart from this, our model recognized predictions from other flat NER layers, leading to 5% errors. We define this error type as *layer error*. Unlike the first flat NER layer, 26% of errors were caused by layer error. Additionally, 17% of the errors belong to type error. In particular, 22% errors were due to the type error. As for the last flat NER layer, 40% errors were caused by partial prediction error. The rest errors were different from the mentioned error types. One possible reason is that we have less gold entities to train this flat NER layer compared with previous flat NER layers. Another reason might be the error propagation.

Similarly, 200 sentences were randomly selected from GENIA test dataset. We got 20% errors of predictions in the subset. Among these errors, 17% and 24% of errors were separately due to type error and partial prediction error. In addition, 24% of the predictions on the first flat NER layer were incorrect. Among them, the top error types were layer error, partial prediction error and type error, accounting for 21%, 18% and 13%, respectively. Errors on the second flat NER layer were mainly caused by type error and the and partial prediction error.

5 Related Work

The success of neural networks has boosted the performance of flat named NER in different domains (Lample et al., 2016; Ma and Hovy, 2016; Gridach, 2017; Strubell et al., 2017). Such models achieved the state of the art without any handcrafted features and external knowledge resources.

Contrary to flat NER, much fewer attempts have emphasized the nested entity recognition. Existing approaches to nested NER (Shen et al., 2003; Alex et al., 2007; Finkel and Manning, 2009; Lu and Roth, 2015; Xu and Jiang, 2016; Muis and Lu, 2017) mainly rely on hand-crafted features. They also failed to take advantage of the dependencies among nested entities. Our model enables capturing dependencies and automatic learning high-level abstract features from texts.

Early work regarding nested NER involve mainly hybrid systems that combined rules with supervised learning algorithms. For example, Shen et al. (2003), Zhou et al. (2004) and Zhang et al. (2004) employed a Hidden Markov Model to GENIA to extract inner entities and then used rule-based methods to obtain the outer entities. Furthermore, Gu (2006) extracted nested entities based on SVM which were trained separately on both inner entities and outermost entities without putting the hidden relations between nested entities into consideration. All these methods failed to capture the dependencies between nested entities. One trial work is that Alex et al. (2007) separately built a inside-out and outside-in layered CRFs which were able to use the current guesses as the input for next layer. They also cascaded separate CRFs of each entity type by using output from previous CRFs as features of current CRFs, yielding best performance in their work. One of the main drawbacks in the cascading approach was that it failed to handle nested entities sharing the same entity type, which were quite common in natural languages.

Finkel and Manning (2009) proposed a discriminative constituency tree to represent each sentence where the root node was used for connection. All entities were treated as phrases and represented as subtrees following the whole tree structure. Unlike our linguistic features independent model, Finkel and Manning (2009) used a CRF-based approach driven by entity-level features to detect nested entities

Later on, Lu and Roth (2015) built hyper-graphs that allow edges to connect multiple nodes to represent both the nested entities and their references (a.k.a. mentions). One issue in their approach is the spurious structures of hyper-graphs as they enumerate combinations of nodes, types and boundaries to represent entities. In addition, they fail to encode the dependencies among embedded entities using hyper-graphs. In contrast, our model enables nested entity representation by merging representation of multiple tokens composed in the entity and considers it as the longer

entity representation. This allows us to represent outer entities based on inner entity representation, thus managing to capture the relations between inner and outer entities, and hence overcoming the spurious entity structure problem.

As an improvement in overcoming spurious structure issue in Lu and Roth (2015), Muis and Lu (2017) further incorporated mention separators along with features to yield better performance on nested entities. Both Lu and Roth (2015) and Muis and Lu (2017) rely on hand-crafted features to extract nested entities without incorporating hidden dependencies in nested entities. In contrast, we make the most of dependencies of nested entities in our model to encourage outer entity recognition by automatic learning of high-level and abstract features from sequences.

Shared tasks dealing with nested entities like SemEval-2007 Task 9[9] and GermEval-2014[10] were held in order to advance the state-of-the-art on this issue. Additionally, as subtasks in KBP 2015[11] and KBP 2016[12], one of the aims in tri-lingual Entity Discovery and Linking Track (EDL) track was extracting nested entities from textual documents varying from English, Chinese and Spanish. Following this task, Xu and Jiang (2016) firstly developed a new tagging scheme which is based on fixed-size ordinally-forgetting encoding (FOFE) method for text fragment representation. All the entities along their contexts were represented using this novel tagging scheme. Different from the extensively used LSTM-RNNs in sequence labeling task, a feed-forward neural network was used to predict labels on entity level for each fragment in any of given sequences. Additionally, Li et al. (2017) used the model proposed in Lample et al. (2016) to the extract both flat entities and components composed in nested and discontinuous entities. Another BiLSTM was applied to combine the components to get nested and discontinuous entities. However, these methods failed to capture and utilize the inner entity representation to facilitate outer entity detection.

[9] http://nlp.cs.swarthmore.edu/semeval/tasks/index.php

[10] https://sites.google.com/site/germeval2014ner/

[11] https://tac.nist.gov//2015/KBP/

[12] https://tac.nist.gov//2016/KBP/

6 Conclusion

This paper presented a dynamic layered model which takes full advantage of inner entity information to encourage outer entity recognition in an end-to-end manner. Our model is based on a flat NER layer consisting of LSTM and CRF, so our model is able to capture context representation of input sequences and globally decode predicted labels at a flat NER layer without relying on feature-engineering. Our model automatically stacks the flat NER layers with sharing the parameters of LSTM and CRF in the layers. The stacking continues until the current flat NER layer predicts sequences as all outside of entities, which enables stopping dynamically stacked flat NER layers. Each flat NER layer receives the merged context representation as input for outer entity recognition, based on the predicted entities from the previous flat NER layer. With this dynamic end-to-end model, our model is able to outperform existing models, achieving the-state-of-art on two nested NER tasks. In addition, the model can be flexibly simplified as a flat NER model by removing components cascaded after the first NER layer.

Extensive evaluation shows that utilization of inner entities significantly encourages outer entities detection with improvements of 3.9 and 9.1 percentage points in F-score on GENIA and ACE2005, respectively. Additionally, utilization of only current context representation contributes to the performance improvement than use of context representation from multi-layers.

Acknowledgments

We thank the anonymous reviewers for their valuable comments. The first author is financially supported by the University of Manchesters 2016 Presidents Doctoral Scholar Award. Sophia Ananiadou acknowledges BBSRC BB/P025684/1 Japan Partnering Award and BB/M006891/1 Empathy. This research has also been carried out with funding from AIRC/AIST and results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References

Beatrice Alex, Barry Haddow, and Claire Grover. 2007. Recognising nested named entities in biomedical text. In *Proceedings of the Workshop on*

BioNLP 2007: Biological, Translational, and Clinical Language Processing. Association for Computational Linguistics, Association for Computational Linguistics, Stroudsburg, PA, USA, pages 65–72. http://dl.acm.org/citation.cfm?id=1572392.1572404.

Balu Bhasuran, Gurusamy Murugesan, Sabenabanu Abdulkadhar, and Jeyakumar Natarajan. 2016. Stacked ensemble combined with fuzzy matching for biomedical named entity recognition of diseases. *Journal of biomedical informatics* 64(Supplement C):1–9. https://doi.org/https://doi.org/10.1016/j.jbi.2016.09.009.

Kate Byrne. 2007. Nested named entity recognition in historical archive text. In *ICSC*. IEEE Computer Society, pages 589–596. http://dblp.uni-trier.de/db/conf/semco/icsc2007.html#Byrne07.

Billy Chiu, Gamal Crichton, Anna Korhonen, and Sampo Pyysalo. 2016. How to train good word embeddings for biomedical NLP. *Proceedings of the 15th Workshop on Biomedical Natural Language Processing* pages 166–174. http://www.aclweb.org/anthology/W16-2922.

HC Cho, N Okazaki, M Miwa, and J Tsujii. 2010. Nersuite: a named entity recognition toolkit. *Tsujii Laboratory, Department of Information Science, University of Tokyo, Tokyo, Japan* http://nersuite.nlplab.org/.

Xiaocheng Feng, Lifu Huang, Duyu Tang, Heng Ji, Bing Qin, and Ting Liu. 2016. A language-independent neural network for event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Berlin, Germany, volume 2, pages 66–71. http://anthology.aclweb.org/P16-2011.

Jenny Rose Finkel and Christopher D Manning. 2009. Nested named entity recognition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*. Association for Computational Linguistics, Association for Computational Linguistics, Singapore, pages 141–150. http://www.aclweb.org/anthology/D/D09/D09-1015.

Pavlina Fragkou. 2017. Applying named entity recognition and co-reference resolution for segmenting english texts. *Progress in Artificial Intelligence* 6(4):325–346. https://doi.org/10.1007/s13748-017-0127-3.

Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Neural Networks, 1996., IEEE International Conference on*. IEEE, volume 1, pages 347–352. https://doi.org/10.1109/ICNN.1996.548916.

Mourad Gridach. 2017. Character-level neural network for biomedical named entity recognition. *Journal of biomedical informatics* 70:85–91. https://doi.org/10.1016/j.jbi.2017.05.002.

Baohua Gu. 2006. Recognizing nested named entities in GENIA corpus. In *Proceedings of the HLT-NAACL BioNLP Workshop on Linking Natural Language and Biology*. Association for Computational Linguistics, New York, New York, LNLBioNLP '06, pages 112–113. http://www.aclweb.org/anthology/W/W06/W06-3318.

Nitish Gupta, Sameer Singh, and Dan Roth. 2017. Entity linking via joint encoding of types, descriptions, and context. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 2671–2680. https://www.aclweb.org/anthology/D17-1284.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR* abs/1207.0580. http://arxiv.org/abs/1207.0580.

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

Jin-Dong Kim, Tomoko Ohta, Yuka Tateisi, and Junichi Tsujii. 2003. GENIA corpus a semantically annotated corpus for bio-textmining. *Bioinformatics* 19(suppl. 1):i180–i182.

Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. 2004. Introduction to the bio-entity recognition task at JNLPBA. In *Proceedings of the international joint workshop on natural language processing in biomedicine and its applications*. Association for Computational Linguistics, pages 70–75.

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

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 260–270. http://www.aclweb.org/anthology/N16-1030.

Robert Leaman and Zhiyong Lu. 2016. Taggerone: joint named entity recognition and normalization with semi-markov models. *Bioinformatics* 32(18):2839–2846. https://doi.org/10.1093/bioinformatics/btw343.

Robert Leaman, Chih-Hsuan Wei, and Zhiyong Lu. 2015. tmChem: a high performance approach for chemical named entity recognition and normalization. *Journal of cheminformatics* 7(1):S3. https://doi.org/10.1186/1758-2946-7-S1-S3.

Fei Li, Meishan Zhang, Bo Tian, Bo Chen, Guohong Fu, and Donghong Ji. 2017. Recognizing irregular entities in biomedical text via deep neural networks. *Pattern Recognition Letters* 105:105–113. https://doi.org/10.1016/j.patrec.2017.06.009.

Xiao Ling and Daniel S. Weld. 2012. Fine-grained entity recognition. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*. AAAI Press, AAAI'12, pages 94–100. http://dl.acm.org/citation.cfm?id=2900728.2900742.

Wei Lu and Dan Roth. 2015. Joint mention extraction and classification with mention hypergraphs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 857–867. http://aclweb.org/anthology/D15-1102.

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf pages 1064–1074. http://www.aclweb.org/anthology/P16-1101.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, pages 55–60. http://www.aclweb.org/anthology/P14-5010.

Michał Marcińczuk. 2015. Automatic construction of complex features in conditional random fields for named entities recognition. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*. INCOMA Ltd. Shoumen, BULGARIA, Hissar, Bulgaria, pages 413–419. http://www.aclweb.org/anthology/R15-1054.

Lluís Màrquez, Luis Villarejo, M. A. Martí, and Mariona Taulé. 2007. Semeval-2007 task 09: Multilevel semantic annotation of catalan and spanish. In *Proceedings of the 4th International Workshop on Semantic Evaluations*. Association for Computational Linguistics, Stroudsburg, PA, USA, SemEval '07, pages 42–47. http://dl.acm.org/citation.cfm?id=1621474.1621482.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures pages 1105–1116. http://www.aclweb.org/anthology/P16-1105.

Aldrian Obaja Muis and Wei Lu. 2017. Labeling gaps between words: Recognizing overlapping mentions with mention separators. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 2598–2608. https://www.aclweb.org/anthology/D17-1276.

Vardaan Pahuja, Anirban Laha, Shachar Mirkin, Vikas Raykar, Lili Kotlerman, and Guy Lev. 2017. Joint learning of correlated sequence labeling tasks using bidirectional recurrent neural networks pages 548–552. https://doi.org/10.21437/Interspeech.2017-1247.

Tim Rocktäschel, Michael Weidlich, and Ulf Leser. 2012. ChemSpot: a hybrid system for chemical named entity recognition. *Bioinformatics* 28(12):1633–1640.

Dan Shen, Jie Zhang, Guodong Zhou, Jian Su, and Chew-Lim Tan. 2003. Effective adaptation of a hidden markov model-based named entity recognizer for biomedical domain. In *Proceedings of the ACL 2003 workshop on Natural language processing in biomedicine-Volume 13*. Association for Computational Linguistics, Association for Computational Linguistics, Sapporo, Japan, pages 49–56. https://doi.org/10.3115/1118958.1118965.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*. Curran Associates Inc., USA, pages 2951–2959. http://dl.acm.org/citation.cfm?id=2999325.2999464.

M. Stone and R. Arora. 2017. Identifying nominals with no head match co-references using deep learning. *CoRR* abs/1710.00936. https://arxiv.org/abs/1710.00936.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and accurate entity recognition with iterated dilated convolutions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 2670–2680. https://www.aclweb.org/anthology/D17-1283.

Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. 2015. Chainer: a next-generation open source framework for deep learning. In *Proceedings of workshop on machine learning systems (LearningSys) in the twenty-ninth annual conference on neural information processing systems (NIPS)*. volume 5.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. ACE 2005 multilingual training corpus. *Linguistic Data Consortium, Philadelphia* 57.

Yefeng Wang. 2009. Annotating and recognising named entities in clinical notes. In *Proceedings of the ACL-IJCNLP 2009 Student Research Workshop*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACLstudent '09, pages 18–26. http://dl.acm.org/citation.cfm?id=1667884.1667888.

Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE* 78(10):1550–1560. https://doi.org/10.1109/5.58337.

Mingbin Xu and Hui Jiang. 2016. A FOFE-based local detection approach for named entity recognition and mention detection. *arXiv preprint arXiv:1611.00801* https://arxiv.org/abs/1611.00801.

Jie Zhang, Dan Shen, Guodong Zhou, Jian Su, and Chew-Lim Tan. 2004. Enhancing HMM-based biomedical named entity recognition by studying special phenomena. *Journal of biomedical informatics* 37(6):411–422. https://doi.org/https://doi.org/10.1016/j.jbi.2004.08.005.

Guodong Zhou, Jie Zhang, Jian Su, Dan Shen, and Chewlim Tan. 2004. Recognizing names in biomedical texts: a machine learning approach. *Bioinformatics* 20(7):1178–1190. https://doi.org/10.1093/bioinformatics/bth060.

A Data Statistics and Preprocessing

Statistics of GENIA, ACE2005 and JNLPBA are described in Tables 7, 8 and 9, respectively.

We used NERSuite (Cho et al., 2010) for GENIA to perform tokenization while Stanford CoreNLP (Manning et al., 2014) was used for ACE2005. The JNLPBA dataset has already been went through tokenization and sentence splitting, so we did not apply any preprocessing.

For GENIA, we had to manually revolve the following two issues, in addition to the above preprocessing.

One of the issues we had in this corpus is the removal of discontinuous entities during parsing. In provided GENIA XML file, each flat entity is annotated with 'lex' (lexical) and 'sem' (semantics) attributes while discontinuous and nested entities may have none, one or two attributes when these entities embed with each other, making it difficult to extract the strictly nested ones. Taken the text "recombinant human nm23-H1, -H2, mouse nm23-M1, and -M2 proteins" as an example, there are six discontinuous entities, "recombinant human nm23-H1 protein", "recombinant human H2 protein", "recombinant mouse nm23-M1 protein", "recombinant mouse nm23-M2 protein", "mouse nm23-M2" and "human nm23-H2", and two nested entities, "mouse nm23-M1" and "human nm23-H1". We extract these nested entities based on symbol * appeared 'lex' attribute which

Item	Train	Dev.	Test
Documents	1,599	189	212
Sentences	15,022	1,669	1,855
Split percentage	81%	9%	10%
DNA	7,921	1061	1,283
RNA	730	140	117
Protein	29,032	2,338	3,098
Cell Line	3,149	340	460
Cell Type	6,021	563	617
Outermost entity	42,462	4,020	4,942
Nested level	4	3	3
Entities in level 1	42,846	4,060	4,991
Entities in level 2	3,910	381	569
Entities in level 3	91	1	15
Entities in level 4	1	0	0
Entity avg. length	2.87	3.13	2.93
Multi-token entity	33951	3554	4203
Overall entities	46,853	4,442	5,575

Table 7: Statistics of GENIA.

Item	Train	Dev.	Test
Documents	370	43	51
Sentences	9,849	1,221	1,478
FAC	924	83	173
GPE	4,725	486	671
LOC	763	81	69
ORG	3,702	479	559
PER	13,050	1,668	1,949
VEH	624	81	66
WEA	652	94	67
Outermost entity	18,455	2,285	2,724
Nested level	6	4	5
Entities in level 1	19,676	2,429	2,936
Entities in level 2	3,934	448	505
Entities in level 3	731	85	102
Entities in level 4	90	10	10
Entities in level 5	7	0	1
Entities in level 6	2	0	0
Entity avg. length	2.28	2.33	2.28
Multi-token entity	10,577	1,323	1,486
Overall entities	24,440	2,972	3,554

Table 8: Statistics of ACE2005.

Item	Train	Dev.	Test
Sentences	16,691	1,855	3,856
Split percentage	90%	10%	-
DNA	8,649	884	1,056
RNA	863	88	118
Protein	27,263	3,006	5,067
Cell Line	3,459	371	500
Cell Type	6,045	673	1,921
Overall entities	46,279	5,022	8,662

Table 9: Statistics of JNLPBA.

Hyper params	Range	Best
Batch size	[16 – 256]	67
No. of hidden units	200, 250, 300	200
Dim. of char. emb.	[15 – 50]	35
Dropout rate	[0.1 – 0.5]	0.2144
Learning rate	[0.001 – 0.02]	0.00754
Gradient clipping	[5 – 50]	27
Weight decay (L2)	$[10^{-8} – 10^{-3}]$	4.54^{-5}

Table 10: Value range and best value of tuned hyper parameters in GENIA.

Hyper params	Range	Best
Batch size	[16 – 256]	91
No. of hidden units	200, 250, 300	200
Dim. of char. emb.	[15 – 50]	28
Dropout rate	[0.1 – 0.5]	0.1708
Learning rate	[0.001 – 0.02]	0.00426
Gradient clipping	[5 – 50]	11
Weight decay (L2)	$[10^{-8} – 10^{-3}]$	9.43^{-5}

Table 11: Value range and best value of tuned hyper parameters in ACE2005.

Hyper Parameters	Initialized Value
Acquisition Function	gp_hedge
n-calls	10
n_random_state	None
n_random_starts	10
Acquisition Optimizer	lbfgs
n_restarts_optimizer	100
noise	gaussian
n_points	50000
xi	0.1
n_jobs	1

Table 12: Hyper parameters used of Bayesian Optimization.

is an connection indicator of the separated texts in discontinuous entities. Meanwhile, each of the separated texts has no 'sem' attribute unless itself is an innermost entity. Unfortunately, there are some inconsistent cases such as "c-fos and c-jun transcripts" where symbol * should be in the 'lex' attribute as the discontinuous entity "c-fos transcript" is connected by "c-fos" and "transcript" while "c-jun transcript" is connected by "c-jun" and "transcript". These two entities share the same text "transcript". However, each of them is annotated with two attributes: 'lex' and 'sem', following the same annotation for flat entities. Although it is possible to ignore the latter entity based on 'lex' attribute and its belonging sentence, this rule fails to deal with entity "c-jun gene" in the example of "c-fos and c-jun genes" as the 'lex' of "c-jun gene" is mistaken as "c-jun genes". Therefore, in this case, we ignored "c-fos transcript" and instead kept the "c-jun transcripts" as a flat entity.

Another issue is the incomplete tokenization. The label assignment to one word was conducted on the word-level instead of character level, but there are entities that correspond to parts of words. An example is "NF-YA subunit", which contains two protein entities: "NF-Y" and "A subunit". To cope with this problem, we treat both two entities as false negative entities in training dataset as there are only 13 such entities in the training data set.

B Bayesian Optimization Setting

The hyper-parameters which were tuned for our model are listed in Table 10 and Table 11. These hyper-parameters are tuned by Bayesian optimization with the hyper parameters listed in Table 12.

C Model Structure

Figure 4 shows the model architecture when we skip all intermediate LSTM layers and only word embeddings are used to produce the input for the next flat NER layer.

Figure 5 describes the model architecture when we skip the representation of non-entity words to prepare the input for the next flat NER layer. Concretely, we merge and average representation following Equation 1. For the predicted non-entity words, however, we skip the LSTM layer and directly use their corresponding representation from the input rather than the output context representation.

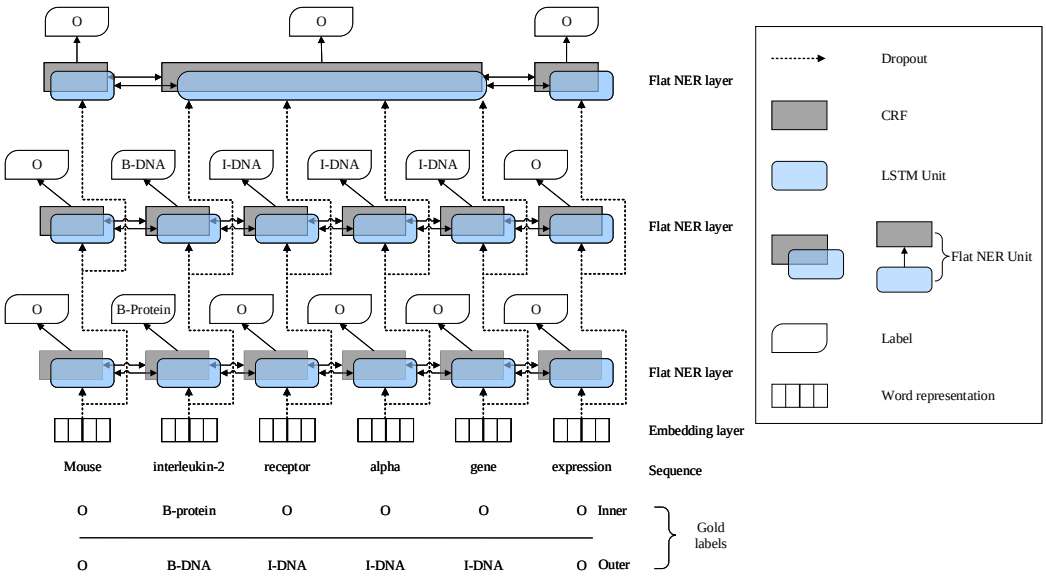


Figure 4: Overview of the model architecture with skipping representation for non-entity words. “interleukin-2” and “interleukin-2 receptor alpha gene” are nested entities.

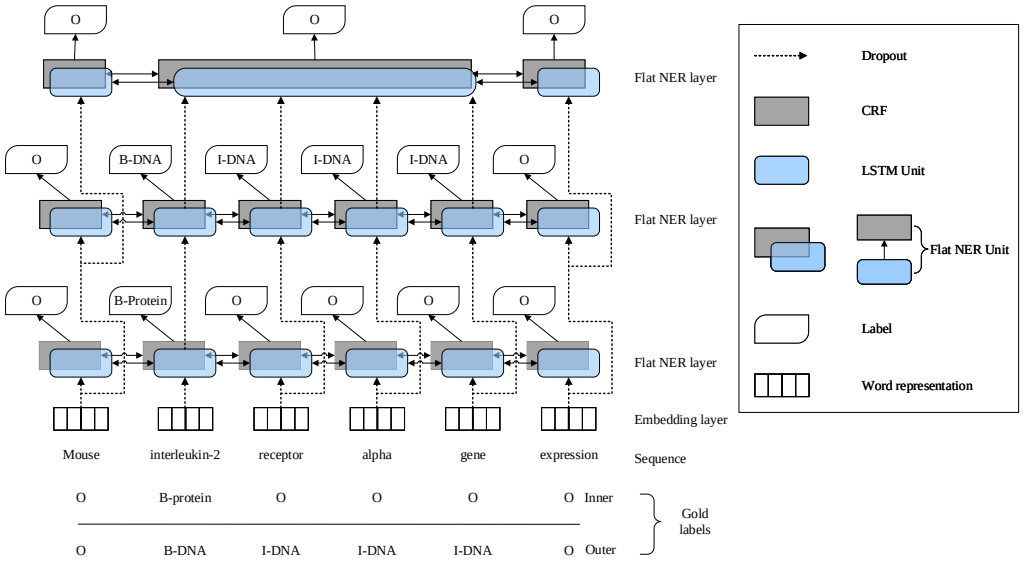


Figure 5: Overview of the model architecture with skipping representation for whole sequence. “interleukin-2” and “interleukin-2 receptor alpha gene” are nested entities.

DR-BiLSTM: Dependent Reading Bidirectional LSTM for Natural Language Inference*

Reza Ghaeini[1†], **Sadid A. Hasan**[2], **Vivek Datla**[2], **Joey Liu**[2], **Kathy Lee**[2], **Ashequl Qadir**[2], **Yuan Ling**[2], **Aaditya Prakash**[2], **Xiaoli Z. Fern**[1] **and Oladimeji Farri**[2]
[1]Oregon State University, Corvallis, OR, USA
[2]Artificial Intelligence Laboratory, Philips Research North America, Cambridge, MA, USA
{ghaeinim,xfern}@eecs.oregonstate.edu
{sadid.hasan,vivek.datla,joey.liu,kathy.lee_1,ashequl.qadir}@philips.com
{yuan.ling,aaditya.prakash,dimeji.farri}@philips.com

Abstract

We present a novel deep learning architecture to address the natural language inference (NLI) task. Existing approaches mostly rely on simple reading mechanisms for independent encoding of the premise and hypothesis. Instead, we propose a novel *dependent reading* bidirectional LSTM network (DR-BiLSTM) to efficiently model the relationship between a premise and a hypothesis during encoding and inference. We also introduce a sophisticated ensemble strategy to combine our proposed models, which noticeably improves final predictions. Finally, we demonstrate how the results can be improved further with an additional preprocessing step. Our evaluation shows that DR-BiLSTM obtains the best single model and ensemble model results achieving the new state-of-the-art scores on the Stanford NLI dataset.

1 Introduction

Natural Language Inference (NLI; a.k.a. Recognizing Textual Entailment, or RTE) is an important and challenging task for natural language understanding (MacCartney and Manning, 2008). The goal of NLI is to identify the logical relationship (*entailment*, *neutral*, or *contradiction*) between a premise and a corresponding hypothesis. Table 1 shows few example relationships from the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015).

Recently, NLI has received a lot of attention from the researchers, especially due to the availability of large annotated datasets like SNLI (Bowman et al., 2015). Various deep learning models have been proposed that achieve successful results for this task (Gong et al., 2017; Wang et al., 2017; Chen et al., 2017; Yu and Munkhdalai, 2017a; Parikh et al., 2016; Zhao et al., 2016; Sha et al., 2016). Most of these existing NLI models use attention mechanism to jointly interpret and align the premise and hypothesis. Such models use simple reading mechanisms to encode the premise and hypothesis independently. However, such a complex task require explicit modeling of dependency relationships between the premise and the hypothesis during the encoding and inference processes to prevent the network from the loss of relevant, contextual information. In this paper, we refer to such strategies as *dependent reading*.

P[a]	A senior is waiting at the window of a restaurant that serves sandwiches.	Relationship
H[b]	A person waits to be served his food.	Entailment
	A man is looking to order a grilled cheese sandwich.	Neutral
	A man is waiting in line for the bus.	Contradiction
[a]**P**, Premise. [b]**H**, Hypothesis.		

Table 1: Examples from the SNLI dataset.

There are some alternative reading mechanisms available in the literature (Sha et al., 2016; Rocktäschel et al., 2015) that consider dependency aspects of the premise-hypothesis relationships. However, these mechanisms have two major limitations:

- So far, they have only explored dependency aspects during the encoding stage, while ignoring its benefit during inference.
- Such models only consider encoding a hy-

* ArXiv version of this work can be found here (arxiv.org/pdf/1802.05577.pdf).

† This work was conducted as part of an internship program at Philips Research.

Proceedings of NAACL-HLT 2018, pages 1460–1469
New Orleans, Louisiana, June 1 - 6, 2018.

pothesis depending on the premise, disregarding the dependency aspects in the opposite direction.

We propose a dependent reading bidirectional LSTM (DR-BiLSTM) model to address these limitations. Given a premise u and a hypothesis v, our model first encodes them considering dependency on each other ($u|v$ and $v|u$). Next, the model employs a soft attention mechanism to extract relevant information from these encodings. The augmented sentence representations are then passed to the inference stage, which uses a similar dependent reading strategy in both directions, i.e. $u \rightarrow v$ and $v \rightarrow u$. Finally, a decision is made through a multi-layer perceptron (MLP) based on the aggregated information.

Our experiments on the SNLI dataset show that DR-BiLSTM achieves the best single model and ensemble model performance obtaining improvements of a considerable margin of 0.4% and 0.3% over the previous state-of-the-art single and ensemble models, respectively.

Furthermore, we demonstrate the importance of a simple preprocessing step performed on the SNLI dataset. Evaluation results show that such preprocessing allows our single model to achieve the same accuracy as the state-of-the-art ensemble model and improves our ensemble model to outperform the state-of-the-art ensemble model by a remarkable margin of 0.7%. Finally, we perform an extensive analysis to clarify the strengths and weaknesses of our models.

2 Related Work

Early studies use small datasets while leveraging lexical and syntactic features for NLI (MacCartney and Manning, 2008). The recent availability of large-scale annotated datasets (Bowman et al., 2015; Williams et al., 2017) has enabled researchers to develop various deep learning-based architectures for NLI.

Parikh et al. (2016) propose an attention-based model (Bahdanau et al., 2014) that decomposes the NLI task into sub-problems to solve them in parallel. They further show the benefit of adding intra-sentence attention to input representations. Chen et al. (2017) explore sequential inference models based on chain LSTMs with attentional input encoding and demonstrate the effectiveness of syntactic information. We also use similar attention mechanisms. However, our model is distinct from these models as they do not benefit from dependent reading strategies.

Rocktäschel et al. (2015) use a word-by-word neural attention mechanism while Sha et al. (2016) propose re-read LSTM units by considering the dependency of a hypothesis on the information of its premise ($v|u$) to achieve promising results. However, these models suffer from weak inferencing methods by disregarding the dependency aspects from the opposite direction ($u|v$). Intuitively, when a human judges a premise-hypothesis relationship, s/he might consider back-and-forth reading of both sentences before coming to a conclusion. Therefore, it is essential to encode the premise-hypothesis dependency relations from both directions to optimize the understanding of their relationship.

Wang et al. (2017) propose a bilateral multi-perspective matching (BiMPM) model, which resembles the concept of matching a premise and hypothesis from both directions. Their matching strategy is essentially similar to our attention mechanism that utilizes relevant information from the other sentence for each word sequence. They use similar methods as Chen et al. (2017) for encoding and inference, without any dependent reading mechanism.

Although NLI is well studied in the literature, the potential of dependent reading and interaction between a premise and hypothesis is not rigorously explored. In this paper, we address this gap by proposing a novel deep learning model (DR-BiLSTM). Experimental results demonstrate the effectiveness of our model.

3 Model

Our proposed model (DR-BiLSTM) is composed of the following major components: input encoding, attention, inference, and classification. Figure 1 demonstrates a high-level view of our proposed NLI framework.

Let $u = [u_1, \cdots, u_n]$ and $v = [v_1, \cdots, v_m]$ be the given premise with length n and hypothesis with length m respectively, where $u_i, v_j \in \mathbb{R}^r$ is an word embedding of r-dimensional vector. The task is to predict a label y that indicates the logical relationship between premise u and hypothesis v.

3.1 Input Encoding

RNNs are the natural solution for variable length sequence modeling, consequently, we utilize a

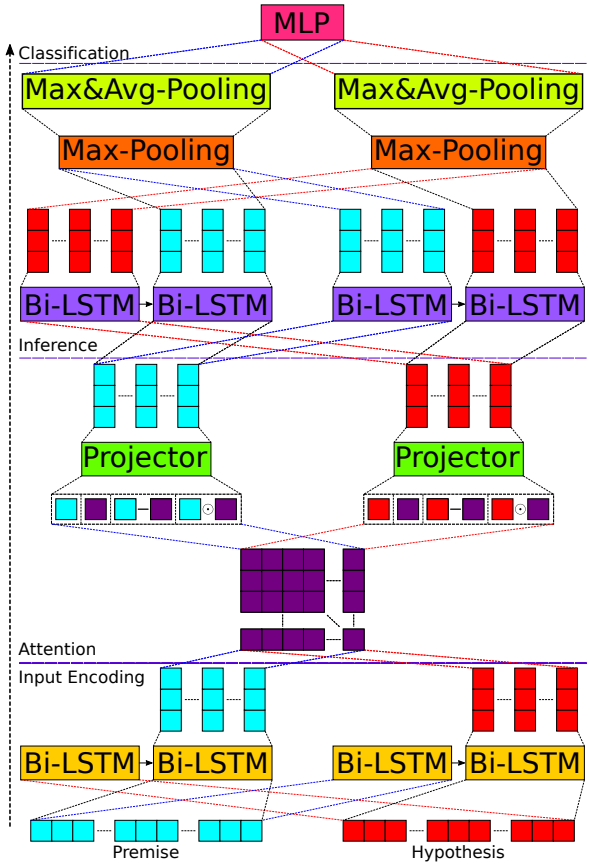


Figure 1: A high-level view of DR-BiLSTM. The data (premise u and hypothesis v, depicted with cyan and red tensors respectively) flows from bottom to top. Relevant tensors are shown with the same color and elements with the same colors share parameters.

bidirectional LSTM (BiLSTM) (Hochreiter and Schmidhuber, 1997) for encoding the given sentences. For ease of presentation, we only describe how we encode u depending on v. The same procedure is utilized for the reverse direction $(v|u)$.

To dependently encode u, we first process v using the BiLSTM. Then we read u through the BiLSTM that is initialized with previous reading final states (memory cell and hidden state). Here we represent a word (e.g. u_i) and its context depending on the other sentence (e.g. v). Equations 1 and 2 formally represent this component.

$$\begin{aligned} \bar{v}, s_v &= \textit{BiLSTM}(v, 0) \\ \hat{u}, - &= \textit{BiLSTM}(u, s_v) \end{aligned} \tag{1}$$

$$\begin{aligned} \bar{u}, s_u &= \textit{BiLSTM}(u, 0) \\ \hat{v}, - &= \textit{BiLSTM}(v, s_u) \end{aligned} \tag{2}$$

where $\{\bar{u} \in \mathbb{R}^{n \times 2d}, \hat{u} \in \mathbb{R}^{n \times 2d}, s_u\}$ and $\{\bar{v} \in \mathbb{R}^{m \times 2d}, \hat{v} \in \mathbb{R}^{m \times 2d}, s_v\}$ are the independent reading sequences, dependent reading sequences, and BiLSTM final state of independent reading of u and v respectively. Note that, "$-$" in these equations means that we do not care about the associated variable and its value. BiLSTM inputs are the word embedding sequences and initial state vectors. $\hat{u}$ and $\hat{v}$ are passed to the next layer as the output of the input encoding component.

The proposed encoding mechanism yields a richer representation for both premise and hypothesis by taking the history of each other into account. Using a max or average pooling over the independent and dependent readings does not further improve our model. This was expected since dependent reading produces more promising and relevant encodings.

3.2 Attention

We employ a soft alignment method to associate the relevant sub-components between the given premise and hypothesis. In deep learning models, such purpose is often achieved with a soft attention mechanism. Here we compute the unnormalized attention weights as the similarity of hidden states of the premise and hypothesis with Equation 3 (energy function).

$$e_{ij} = \hat{u}_i \hat{v}_j^T, \quad i \in [1, n], j \in [1, m] \tag{3}$$

where $\hat{u}_i$ and $\hat{v}_j$ are the dependent reading hidden representations of u and v respectively which are computed earlier in Equations 1 and 2. Next, for each word in either premise or hypothesis, the relevant semantics in the other sentence is extracted and composed according to e_{ij}. Equations 4 and 5 provide formal and specific details of this procedure.

$$\tilde{u}_i = \sum_{j=1}^{m} \frac{\exp(e_{ij})}{\sum_{k=1}^{m} \exp(e_{ik})} \hat{v}_j, \quad i \in [1, n] \tag{4}$$

$$\tilde{v}_j = \sum_{i=1}^{n} \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{kj})} \hat{u}_i, \quad j \in [1, m] \tag{5}$$

where $\tilde{u}_i$ represents the extracted relevant information of $\hat{v}$ by attending to $\hat{u}_i$ while $\tilde{v}_j$ represents the extracted relevant information of $\hat{u}$ by attending to $\hat{v}_j$.

To further enrich the collected attentional information, a trivial next step would be to pass the concatenation of the tuples $(\hat{u}_i, \tilde{u}_i)$ or $(\hat{v}_j, \tilde{v}_j)$ which provides a linear relationship between them. However, the model would suffer from the absence of *similarity* and *closeness* measures. Therefore, we calculate the difference and element-wise product for the tuples $(\hat{u}_i, \tilde{u}_i)$ and $(\hat{v}_j, \tilde{v}_j)$ that represent the similarity and closeness information respectively (Chen et al., 2017; Kumar et al., 2016).

The difference and element-wise product are then concatenated with the computed vectors, $(\hat{u}_i, \tilde{u}_i)$ or $(\hat{v}_j, \tilde{v}_j)$, respectively. Finally, a feed-forward neural layer with ReLU activation function projects the concatenated vectors from $8d$-dimensional vector space into a d-dimensional vector space (Equations 6 and 7). This helps the model to capture deeper dependencies between the sentences besides lowering the complexity of vector representations.

$$\begin{aligned} a_i &= [\hat{u}_i, \tilde{u}_i, \hat{u}_i - \tilde{u}_i, \hat{u}_i \odot \tilde{u}_i] \\ p_i &= \textit{ReLU}(W_p a_i + b_p) \end{aligned} \tag{6}$$

$$\begin{aligned} b_j &= [\hat{v}_j, \tilde{v}_j, \hat{v}_j - \tilde{v}_j, \hat{v}_j \odot \tilde{v}_j] \\ q_j &= \textit{ReLU}(W_p b_j + b_p) \end{aligned} \tag{7}$$

Here $\odot$ stands for element-wise product while $W_p \in \mathbb{R}^{8d \times d}$ and $b_p \in \mathbb{R}^d$ are the trainable weights and biases of the projector layer respectively.

3.3 Inference

During this phase, we use another BiLSTM to aggregate the two sequences of computed matching vectors, p and q from the attention stage (Section 3.2). This aggregation is performed in a sequential manner to avoid losing effect of latent variables that might rely on the sequence of matching vectors.

Instead of aggregating the sequences of matching vectors individually, we propose a similar dependent reading approach for the inference stage. We employ a BiLSTM reading process (Equations 8 and 9) similar to the input encoding step discussed in Section 3.1. But rather than passing just the dependent reading information to the next step, we feed both independent reading ($\bar{p}$ and $\bar{q}$) and dependent reading ($\hat{p}$ and $\hat{q}$) to a max pooling layer, which selects maximum values from each sequence of independent and dependent readings ($\bar{p}_i$ and $\hat{p}_i$) as shown in Equations 10 and 11. The main intuition behind this architecture is to maximize the inferencing ability of the model by considering both independent and dependent readings.

$$\begin{aligned} \bar{q}, s_q &= \textit{BiLSTM}(q, 0) \\ \hat{p}, - &= \textit{BiLSTM}(p, s_q) \end{aligned} \tag{8}$$

$$\begin{aligned} \bar{p}, s_p &= \textit{BiLSTM}(p, 0) \\ \hat{q}, - &= \textit{BiLSTM}(q, s_p) \end{aligned} \tag{9}$$

$$\tilde{p} = \textit{MaxPooling}(\bar{p}, \hat{p}) \tag{10}$$

$$\tilde{q} = \textit{MaxPooling}(\bar{q}, \hat{q}) \tag{11}$$

Here $\{\bar{p} \in \mathbb{R}^{n \times 2d}, \hat{p} \in \mathbb{R}^{n \times 2d}, s_p\}$ and $\{\bar{q} \in \mathbb{R}^{m \times 2d}, \hat{q} \in \mathbb{R}^{m \times 2d}, s_q\}$ are the independent reading sequences, dependent reading sequences, and BiLSTM final state of independent reading of p and q respectively. BiLSTM inputs are the word embedding sequences and initial state vectors.

Finally, we convert $\tilde{p} \in \mathbb{R}^{n \times 2d}$ and $\tilde{q} \in \mathbb{R}^{m \times 2d}$ to fixed-length vectors with pooling, $U \in \mathbb{R}^{4d}$ and $V \in \mathbb{R}^{4d}$. As shown in Equations 12 and 13, we employ both max and average pooling and describe the overall inference relationship with concatenation of their outputs.

$$U = [\textit{MaxPooling}(\tilde{p}), \textit{AvgPooling}(\tilde{p})] \tag{12}$$

$$V = [\textit{MaxPooling}(\tilde{q}), \textit{AvgPooling}(\tilde{q})] \tag{13}$$

3.4 Classification

Here, we feed the concatenation of U and V ($[U, V]$) into a multilayer perceptron (MLP) classifier that includes a hidden layer with *tanh* activation and *softmax* output layer. The model is trained in an end-to-end manner.

$$\textit{Output} = \textit{MLP}([U, V]) \tag{14}$$

4 Experiments and Evaluation

4.1 Dataset

The Stanford Natural Language Inference (SNLI) dataset contains $570K$ human annotated sentence pairs. The premises are drawn from the Flickr30k (Plummer et al., 2015) corpus, and then the hypotheses are manually composed for each relationship class (*entailment*, *neutral*, *contradiction*, and -). The "-" class indicates that there is no consensus decision among the annotators, consequently, we remove them during the training and evaluation following the literature. We use the same data split as provided in Bowman et al. (2015) to report comparable results with other models.

4.2 Experimental Setup

We use pre-trained 300-D Glove $840B$ vectors (Pennington et al., 2014) to initialize our word embedding vectors. All hidden states of BiLSTMs during input encoding and inference have 450 dimensions ($r = 300$ and $d = 450$). The weights are learned by minimizing the log-loss on the training data via the Adam optimizer (Kingma and Ba, 2014). The initial learning rate is 0.0004. To avoid overfitting, we use dropout (Srivastava et al., 2014) with the rate of 0.4 for regularization, which is applied to all feedforward connections. During training, the word embeddings are updated to learn effective representations for the NLI task. We use a fairly small batch size of 32 to provide more exploration power to the model. Our observation indicates that using larger batch sizes hurts the performance of our model.

4.3 Ensemble Strategy

Ensemble methods use multiple models to obtain better predictive performance. Previous works typically utilize trivial ensemble strategies by either using majority votes or averaging the probability distributions over the same model with different initialization seeds (Wang et al., 2017; Gong et al., 2017).

By contrast, we use weighted averaging of the probability distributions where the weight of each model is learned through its performance on the SNLI development set. Furthermore, the differences between our models in the ensemble originate from: 1) variations in the number of dependent readings (i.e. 1 and 3 rounds of dependent reading), 2) projection layer activation (*tanh* and *ReLU* in Equations 6 and 7), and 3) different initialization seeds.

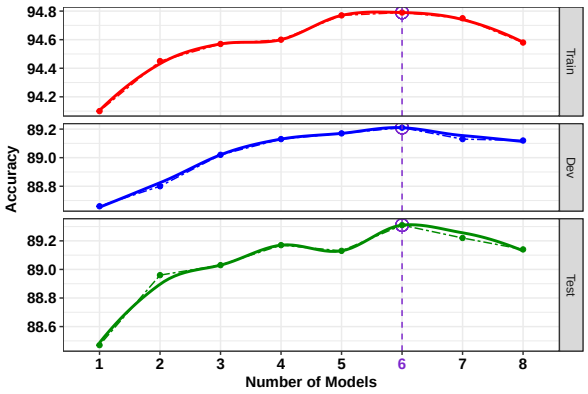


Figure 2: Performance of n ensemble models reported for training (red, top), development (blue, middle), and test (green, bottom) sets of SNLI. For n number of models, the best performance on the development set is used as the criteria to determine the final ensemble. The best performance on development set (89.22%) is observed using 6 models and is henceforth considered as our final DR-BiLSTM (Ensemble) model.

The main intuition behind this design is that the effectiveness of a model may depend on the complexity of a premise-hypothesis instance. For a simple instance, a simple model could perform better than a complex one, while a complex instance may need further consideration toward disambiguation. Consequently, using models with different rounds of dependent readings in the encoding stage should be beneficial.

Figure 2 demonstrates the observed performance of our ensemble method with different number of models. The performance of the models are reported based on the best obtained accuracy on the development set. We also study the effectiveness of other ensemble strategies e.g. majority voting, and averaging the probability distributions. But, our ensemble strategy performs the best among them (see Section 1 in the supplementary material for additional details).

4.4 Preprocessing

We perform a trivial preprocessing step on SNLI to recover some out-of-vocabulary words found in the development set and test set. Note that our vocabulary contains all words that are seen in the training set, so there is no out-of-vocabulary word in it. The SNLI dataset is not immune to human

errors, specifically, misspelled words. We noticed that misspelling is the main reason for some of the observed out-of-vocabulary words. Consequently, we simply fix the unseen misspelled words using Microsoft spell-checker (other approaches like edit distance can also be used). Moreover, while dealing with an unseen word during evaluation, we try to: 1) replace it with its lower case, or 2) split the word when it contains a "-" (e.g. "marsh-like") or starts with "un" (e.g. "unloading"). If we still could not find the word in our vocabulary, we consider it as an *unknown* word. In the next subsection, we demonstrate the importance and impact of such trivial preprocessing (see Section 2 in the supplementary material for additional details).

4.5 Results

Table 2 shows the accuracy of the models on training and test sets of SNLI. The first row represents a baseline classifier presented by Bowman et al. (2015) that utilizes handcrafted features. All other listed models are deep-learning based. The gap between the traditional model and deep learning models demonstrates the effectiveness of deep learning methods for this task. We also report the estimated human performance on the SNLI dataset, which is the average accuracy of five annotators in comparison to the gold labels (Gong et al., 2017). It is noteworthy that recent deep learning models surpass the human performance in the NLI task.

As shown in Table 2, previous deep learning models (rows 2-19) can be divided into three categories: 1) sentence encoding based models (rows 2-7), 2) single inter-sentence attention-based models (rows 8-16), and 3) ensemble inter-sentence attention-based models (rows 17-19). We can see that inter-sentence attention-based models perform better than sentence encoding based models, which supports our intuition. Natural language inference requires a deep interaction between the premise and hypothesis. Inter-sentence attention-based approaches can provide such interaction while sentence encoding based models fail to do so.

To further enhance the modeling of interaction between the premise and hypothesis for efficient disambiguation of their relationship, we introduce the dependent reading strategy in our proposed DR-BiLSTM model. The results demonstrate the effectiveness of our model. DR-BiLSTM (Single) achieves 88.5% accuracy on the test set which is noticeably the best reported result among the existing single models for this task. Note that the difference between DR-BiLSTM and Chen et al. (2017) is statistically significant with a p-value of < 0.001 over the *Chi-square* test[1].

Model	Accuracy	
	Train	Test
(Bowman et al., 2015) (Feature)	99.7%	78.2%
(Bowman et al., 2015)	83.9%	80.6%
(Vendrov et al., 2015)	98.8%	81.4%
(Mou et al., 2016)	83.3%	82.1%
(Bowman et al., 2016)	89.2%	83.2%
(Liu et al., 2016b)	84.5%	84.2%
(Yu and Munkhdalai, 2017a)	86.2%	84.6%
(Rocktäschel et al., 2015)	85.3%	83.5%
(Wang and Jiang, 2016)	92.0%	86.1%
(Liu et al., 2016a)	88.5%	86.3%
(Parikh et al., 2016)	90.5%	86.8%
(Yu and Munkhdalai, 2017b)	88.5%	87.3%
(Sha et al., 2016)	90.7%	87.5%
(Wang et al., 2017) (Single)	90.9%	87.5%
(Chen et al., 2017) (Single)	92.6%	88.0%
(Gong et al., 2017) (Single)	91.2%	88.0%
(Chen et al., 2017) (Ensemble)	93.5%	88.6%
(Wang et al., 2017) (Ensemble)	93.2%	88.8%
(Gong et al., 2017) (Ensemble)	92.3%	88.9%
Human Performance (Estimated)	97.2%	87.7%
DR-BiLSTM (Single)	94.1%	**88.5%**
DR-BiLSTM (Single)+Process	94.1%	**88.9%**
DR-BiLSTM (Ensemble)	94.8%	**89.3%**
DR-BiLSTM (Ensem.)+Process	94.8%	**89.6%**

Table 2: Accuracies of the models on the training set and test set of SNLI. DR-BiLSTM (Ensemble) achieves the accuracy of 89.3%, the best result observed on SNLI, while DR-BiLSTM (Single) obtains the accuracy of 88.5%, which considerably outperforms the previous non-ensemble models. Also, utilizing a trivial preprocessing step yields to further improvements of 0.4% and 0.3% for single and ensemble DR-BiLSTM models respectively.

To further improve the performance of NLI systems, researchers have built ensemble models. Previously, ensemble systems obtained the best performance on SNLI with a huge margin. Table 2 shows that our proposed single model achieves competitive results compared to these reported ensemble models. Our ensemble model considerably outperforms the current state-of-the-art by obtaining 89.3% accuracy.

Up until this point, we discussed the performance of our models where we have not con-

[1]Chi-square test (χ^2 test) is used to determine if there is a significant difference between two categorical variables (i.e. models' outputs).

sidered preprocessing for recovering the out-of-vocabulary words. In Table 2, "DR-BiLSTM (Single) + Process", and "DR-BiLSTM (Ensem.) + Process" represent the performance of our models on the preprocessed dataset. We can see that our preprocessing mechanism leads to further improvements of 0.4% and 0.3% on the SNLI test set for our single and ensemble models respectively. In fact, our single model ("DR-BiLSTM (Single) + Process") obtains the state-of-the-art performance over both reported single and ensemble models by performing a simple preprocessing step. Furthermore, "DR-BiLSTM (Ensem.) + Process" outperforms the existing state-of-the-art remarkably (0.7% improvement). For more comparison and analyses, we use "DR-BiLSTM (Single)" and "DR-BiLSTM (Ensemble)" as our single and ensemble models in the rest of the paper.

4.6 Ablation and Configuration Study

We conducted an ablation study on our model to examine the importance and effect of each major component. Then, we study the impact of BiLSTM dimensionality on the performance of the development set and training set of SNLI. We investigate all settings on the development set of the SNLI dataset.

Model	Dev Acc[a]	p-value
DR-BiLSTM	**88.69%**	-
DR-BiLSTM - hidden MLP	88.45%	<0.001
DR-BiLSTM - average pooling	88.50%	<0.001
DR-BiLSTM - max pooling	88.39%	<0.001
DR-BiLSTM - elem. prd[b]	88.51%	<0.001
DR-BiLSTM - difference	88.24%	<0.001
DR-BiLSTM - diff[c] & elem. prd	87.96%	<0.001
DR-BiLSTM - inference pooling	88.46%	<0.001
DR-BiLSTM - dep. infer[d]	88.43%	<0.001
DR-BiLSTM - dep. enc[e]	88.26%	<0.001
DR-BiLSTM - dep. enc & infer	88.20%	<0.001

[a] Dev Acc, Development Accuracy.
[b] elem. prd, element-wise product.
[c] diff, difference.
[d] dep. infer, dependent reading inference.
[e] dep. enc, dependent reading encoding.

Table 3: Ablation study results. Performance of different configurations of the proposed model on the development set of SNLI along with their p-values in comparison to DR-BiLSTM (Single).

Table 3 shows the ablation study results on the development set of SNLI along with the statistical significance test results in comparison to the proposed model, DR-BiLSTM. We can see that all modifications lead to a new model and their differences are statistically significant with a p-value of < 0.001 over *Chi square* test.

Table 3 shows that removing any part from our model hurts the development set accuracy which indicates the effectiveness of these components. Among all components, three of them have noticeable influences: max pooling, difference in the attention stage, and dependent reading.

Most importantly, the last four study cases in Table 3 (rows 8-11) verify the main intuitions behind our proposed model. They illustrate the importance of our proposed dependent reading strategy which leads to significant improvement, specifically in the encoding stage. We are convinced that the importance of dependent reading in the encoding stage originates from its ability to focus on more important and relevant aspects of the sentences due to its prior knowledge of the other sentence during the encoding procedure.

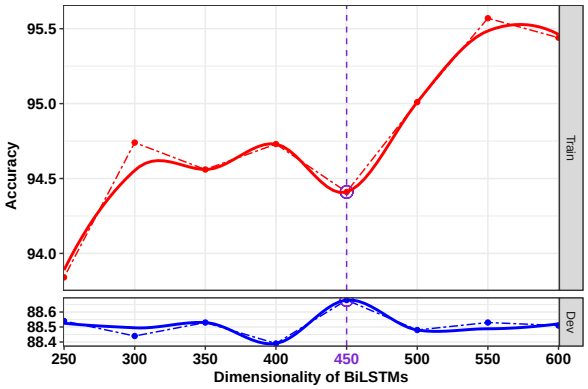


Figure 3: Impact of BiLSTM dimensionality in the proposed model on the training set (red, top) and development set (blue, bottom) accuracies of the SNLI dataset.

Figure 3 shows the behavior of the proposed model accuracy on the training set and development set of SNLI. Since the models are selected based on the best observed development set accuracy during the training procedure, the training accuracy curve (red, top) is not strictly increasing. Figure 3 demonstrates that we achieve the best performance with 450-dimensional BiLSTMs. In other words, using BiLSTMs with lower dimensionality causes the model to suffer from the lack of space for capturing proper information and dependencies. On the other hand, using higher dimensionality leads to overfitting which hurts the performance on the development set. Hence, we use 450-dimensional BiLSTM in our proposed

model.

4.7 Analysis

We first investigate the performance of our models categorically. Then, we show a visualization of the energy function in the attention stage (Equation 3) for an instance from the SNLI test set.

To qualitatively evaluate the performance of our models, we design a set of annotation tags that can be extracted automatically. This design is inspired by the reported annotation tags in Williams et al. (2017). The specifications of our annotation tags are as follows:

- **High Overlap:** premise and hypothesis sentences share more than 70% tokens.
- **Regular Overlap:** sentences share between 30% and 70% tokens.
- **Low Overlap:** sentences share less than 30% tokens.
- **Long Sentence:** either sentence is longer than 20 tokens.
- **Regular Sentence:** premise or hypothesis length is between 5 and 20 tokens.
- **Short Sentence:** either sentence is shorter than 5 tokens.
- **Negation:** negation is present in a sentence.
- **Quantifier:** either of the sentences contains one of the following quantifiers: much, enough, more, most, less, least, no, none, some, any, many, few, several, almost, nearly.
- **Belief:** either of the sentences contains one of the following belief verbs: know, believe, understand, doubt, think, suppose, recognize, forget, remember, imagine, mean, agree, disagree, deny, promise.

Table 4 shows the frequency of aforementioned annotation tags in the SNLI test set along with the performance (accuracy) of ESIM (Chen et al., 2017), DR-BiLSTM (Single), and DR-BiLSTM (Ensemble). Table 4 can be divided into four major categories: 1) gold label data, 2) word overlap, 3) sentence length, and 4) occurrence of special words. We can see that DR-BiLSTM (Ensemble) performs the best in all categories which matches our expectation. Moreover, DR-BiLSTM (Single) performs noticeably better than ESIM in most of the categories except "Entailment", "High Overlap", and "Long Sentence", for which our model is not far behind (gaps of 0.2%, 0.5%, and 0.9%, respectively). It is noteworthy that DR-BiLSTM (Single) performs better than ESIM in more frequent categories. Specifically, the performance of our model in "Neutral", "Negation", and "Quantifier" categories (improvements of 1.4%, 3.5%, and 1.9%, respectively) indicates the superiority of our model in understanding and disambiguating complex samples. Our investigations indicate that ESIM generates somewhat uniform attention for most of the word pairs while our model could effectively attend to specific parts of the given sentences and provide more meaningful attention. In other words, the dependent reading strategy enables our model to achieve meaningful representations, which leads to better attention to obtain further gains on such categories like Negation and Quantifier sentences (see Section 3 in the supplementary material for additional details).

Annotation Tag	Freq[a]	ESIM	DR(S)[b]	DR(E)[c]
Entailment	34.3%	90.0%	89.8%	90.9%
Neutral	32.8%	83.7%	85.1%	85.6%
Contradiction	32.9%	90.0%	90.5%	91.4%
High Overlap	24.3%	91.2%	90.7%	92.1%
Reg. Overlap	33.7%	87.1%	87.9%	88.8%
Low Overlap	45.4%	87.0%	87.8%	88.4%
Long Sentence	6.4%	92.2%	91.3%	91.9%
Reg. Sentence	74.9%	87.8%	88.4%	89.2%
Short Sentence	19.9%	87.6%	88.1%	89.3%
Negation	2.1%	82.2%	85.7%	87.1%
Quantifier	8.7%	85.5%	87.4%	87.6%
Belief	0.2%	78.6%	78.6%	78.6%

[a]Freq, Frequency.
[b]DR(S), DR-BiLSTM (Single).
[c]DR(E), DR-BiLSTM (Ensemble).

Table 4: Categorical performance analyses (accuracy) of ESIM (Chen et al., 2017), DR-BiLSTM (DR(S)) and Ensemble DR-BiLSTM (DR(E)) on the SNLI test set.

Finally, we show a visualization of the normalized attention weights (energy function, Equation 3) of our model in Figure 4. We show a sentence pair, where the premise is "*Male in a blue jacket decides to lay the grass.*", and the hypothesis is "*The guy in yellow is rolling on the grass.*", and its logical relationship is *contradiction*. Figure 4 indicates the model's ability in attending to critical pairs of words like <Male, guy>, <decides, rolling>, and <lay, rolling>. Finally, high attention between {decides, lay} and

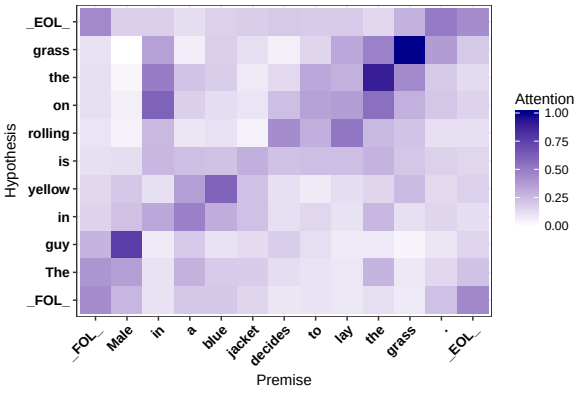


Figure 4: Normalized attention weights for a sample from the SNLI test set. Darker color illustrates higher attention.

{rolling}, and {Male} and {guy} leads the model to correctly classify the sentence pair as *contradiction* (for more samples with attention visualizations, see Section 4 in the supplementary material).

5 Conclusion

We propose a novel natural language inference model (DR-BiLSTM) that benefits from a dependent reading strategy and achieves the state-of-the-art results on the SNLI dataset. We also introduce a sophisticated ensemble strategy and illustrate its effectiveness through experimentation. Moreover, we demonstrate the importance of a simple preprocessing step on the performance of our proposed models. Evaluation results show that the preprocessing step allows our DR-BiLSTM (single) model to outperform all previous single and ensemble methods. Similar superior performance is also observed for our DR-BiLSTM (ensemble) model. We show that our ensemble model outperforms the existing state-of-the-art by a considerable margin of 0.7%. Finally, we perform an extensive analysis to demonstrate the strength and weakness of the proposed model, which would pave the way for further improvements in this domain.

References

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

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. pages 632–642. http://aclweb.org/anthology/D/D15/D15-1075.pdf.

Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. http://aclweb.org/anthology/P/P16/P16-1139.pdf.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. pages 1657–1668. https://doi.org/10.18653/v1/P17-1152.

Yichen Gong, Heng Luo, and Jian Zhang. 2017. Natural language inference over interaction space. *CoRR* abs/1709.04348. http://arxiv.org/abs/1709.04348.

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

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

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*. pages 1378–1387. http://jmlr.org/proceedings/papers/v48/kumar16.html.

Pengfei Liu, Xipeng Qiu, Jifan Chen, and Xuanjing Huang. 2016a. Deep fusion lstms for text semantic matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. http://aclweb.org/anthology/P/P16/P16-1098.pdf.

Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. 2016b. Learning natural language inference using bidirectional LSTM model and inner-attention. *CoRR* abs/1605.09090. http://arxiv.org/abs/1605.09090.

Bill MacCartney and Christopher D. Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *COLING 2008,*

22nd International Conference on Computational Linguistics, Proceedings of the Conference, 18-22 August 2008, Manchester, UK. pages 521–528. http://www.aclweb.org/anthology/C08-1066.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*. http://aclweb.org/anthology/P/P16/P16-2022.pdf.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. pages 2249–2255. http://aclweb.org/anthology/D/D16/D16-1244.pdf.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Bryan A. Plummer, Liwei Wang, Chris M. Cervantes, Juan C. Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2015. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. pages 2641–2649. https://doi.org/10.1109/ICCV.2015.303.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *CoRR* abs/1509.06664. http://arxiv.org/abs/1509.06664.

Lei Sha, Baobao Chang, Zhifang Sui, and Sujian Li. 2016. Reading and thinking: Re-read LSTM unit for textual entailment recognition. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*. pages 2870–2879. http://aclweb.org/anthology/C/C16/C16-1270.pdf.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958. http://dl.acm.org/citation.cfm?id=2670313.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2015. Order-embeddings of images and language. *CoRR* abs/1511.06361. http://arxiv.org/abs/1511.06361.

Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with LSTM. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*. pages 1442–1451. http://aclweb.org/anthology/N/N16/N16-1170.pdf.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*. pages 4144–4150. https://doi.org/10.24963/ijcai.2017/579.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2017. coverage challenge corpus for sentence understanding through inference. *CoRR* abs/1704.05426. http://arxiv.org/abs/1704.05426.

Hong Yu and Tsendsuren Munkhdalai. 2017a. Neural semantic encoders. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*. pages 397–407. http://aclanthology.info/papers/E17-1038/neural-semantic-encoders.

Hong Yu and Tsendsuren Munkhdalai. 2017b. Neural tree indexers for text understanding. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*. pages 11–21. http://aclanthology.info/papers/E17-1002/neural-tree-indexers-for-text-understanding.

Kai Zhao, Liang Huang, and Mingbo Ma. 2016. Textual entailment with structured attentions and composition. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*. pages 2248–2258. http://aclweb.org/anthology/C/C16/C16-1212.pdf.

KBGAN: Adversarial Learning for Knowledge Graph Embeddings

Liwei Cai
Department of Electronic Engineering
Tsinghua University
Beijing 100084 China
cai.lw123@gmail.com

William Yang Wang
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106 USA
william@cs.ucsb.edu

Abstract

We introduce KBGAN, an adversarial learning framework to improve the performances of a wide range of existing knowledge graph embedding models. Because knowledge graphs typically only contain positive facts, sampling useful negative training examples is a non-trivial task. Replacing the head or tail entity of a fact with a uniformly randomly selected entity is a conventional method for generating negative facts, but the majority of the generated negative facts can be easily discriminated from positive facts, and will contribute little towards the training. Inspired by generative adversarial networks (GANs), we use one knowledge graph embedding model as a negative sample generator to assist the training of our desired model, which acts as the discriminator in GANs. This framework is independent of the concrete form of generator and discriminator, and therefore can utilize a wide variety of knowledge graph embedding models as its building blocks. In experiments, we adversarially train two translation-based models, TRANSE and TRANSD, each with assistance from one of the two probability-based models, DISTMULT and COMPLEX. We evaluate the performances of KBGAN on the link prediction task, using three knowledge base completion datasets: FB15k-237, WN18 and WN18RR. Experimental results show that adversarial training substantially improves the performances of target embedding models under various settings.

1 Introduction

Knowledge graph (Dong et al., 2014) is a powerful graph structure that can provide direct access of knowledge to users via various applications such as structured search, question answering, and intelligent virtual assistant. A common representation of knowledge graph beliefs is in the form of a discrete relational triple such as *LocatedIn(NewOrleans,Louisiana)*.

A main challenge for using discrete representation of knowledge graph is the lack of capability of accessing the similarities among different entities and relations. Knowledge graph embedding (KGE) techniques (e.g., RESCAL (Nickel et al., 2011), TRANSE (Bordes et al., 2013), DISTMULT (Yang et al., 2015), and COMPLEX (Trouillon et al., 2016)) have been proposed in recent years to deal with the issue. The main idea is to represent the entities and relations in a vector space, and one can use machine learning technique to learn the continuous representation of the knowledge graph in the latent space.

However, even steady progress has been made in developing novel algorithms for knowledge graph embedding, there is still a common challenge in this line of research. For space efficiency, common knowledge graphs such as Freebase (Bollacker et al., 2008), Yago (Suchanek et al., 2007), and NELL (Mitchell et al., 2015) by default only stores beliefs, rather than disbeliefs. Therefore, when training the embedding models, there is only the natural presence of the positive examples. To use negative examples, a common method is to remove the correct tail entity, and randomly sample from a uniform distribution (Bordes et al., 2013). Unfortunately, this approach is not ideal, because the sampled entity could be completely unrelated to the head and the target relation, and thus the quality of randomly generated negative examples is often poor (e.g, *LocatedIn(NewOrleans,BarackObama)*). Other approach might leverage external ontological constraints such as entity types (Krompaß et al., 2015) to generate negative examples, but such resource does not always exist or accessible.

In this work, we provide a generic solution to improve the training of a wide range of knowl-

Proceedings of NAACL-HLT 2018, pages 1470–1480
New Orleans, Louisiana, June 1 - 6, 2018.

Model	Score function $f(h,r,t)$	Number of parameters
TRANSE	$\|\|\mathbf{h}+\mathbf{r}-\mathbf{t}\|\|_{1/2}$	$k\|\mathcal{E}\|+k\|\mathcal{R}\|$
TRANSD	$\|\|(\mathbf{I}+\mathbf{r_p}\mathbf{h_p}^T)\mathbf{h}+\mathbf{r}-(\mathbf{I}+\mathbf{r_p}\mathbf{t_p}^T)\mathbf{t}\|\|_{1/2}$	$2k\|\mathcal{E}\|+2k\|\mathcal{R}\|$
DISTMULT	$<\mathbf{h},\mathbf{r},\mathbf{t}>(=\sum_{i=1}^{k}h_i r_i t_i)$	$k\|\mathcal{E}\|+k\|\mathcal{R}\|$
COMPLEX	$<\mathbf{h},\mathbf{r},\bar{\mathbf{t}}>(\mathbf{h},\mathbf{r},\mathbf{t}\in\mathbb{C}^k)$	$2k\|\mathcal{E}\|+2k\|\mathcal{R}\|$
TRANSH	$\|\|(\mathbf{I}-\mathbf{r_p}\mathbf{r_p}^T)\mathbf{h}+\mathbf{r}-(\mathbf{I}+\mathbf{r_p}\mathbf{r_p}^T)\mathbf{t}\|\|_{1/2}$	$k\|\mathcal{E}\|+2k\|\mathcal{R}\|$
TRANSR	$\|\|\mathbf{W_r}\mathbf{h}+\mathbf{r}-\mathbf{W_r}\mathbf{t}\|\|_{1/2}$	$k\|\mathcal{E}\|+(k^2+k)\|\mathcal{R}\|$
MANIFOLDE (hyperplane)	$\|(\mathbf{h}+\mathbf{r_{head}})^T(\mathbf{t}+\mathbf{r_{tail}})-D_r\|$	$k\|\mathcal{E}\|+(2k+1)\|\mathcal{R}\|$
RESCAL	$\mathbf{h}^T\mathbf{W_r}\mathbf{t}$	$k\|\mathcal{E}\|+k^2\|\mathcal{R}\|$
HOLE	$\mathbf{r}^T(\mathbf{h}\star\mathbf{t})$ ($\star$ is circular correlation)	$k\|\mathcal{E}\|+k\|\mathcal{R}\|$
CONVE	$f(vec(f([\bar{\mathbf{h}};\bar{\mathbf{r}}]*\omega))\mathbf{W})\mathbf{t}$	$k\|\mathcal{E}\|+k\|\mathcal{R}\|+kcmn$

Table 1: Some selected knowledge graph embedding models. The four models above the double line are considered in this paper. Except for COMPLEX, all boldface lower case letters represent vectors in $\mathbb{R}^k$, and boldface upper case letters represent matrices in $\mathbb{R}^{k\times k}$. $\mathbf{I}$ is the identity matrix.

edge graph embedding models. Inspired by the recent advances of generative adversarial deep models (Goodfellow et al., 2014), we propose a novel adversarial learning framework, namely, KBGAN, for generating better negative examples to train knowledge graph embedding models. More specifically, we consider probability-based, log-loss embedding models as the generator to supply better quality negative examples, and use distance-based, margin-loss embedding models as the discriminator to generate the final knowledge graph embeddings. Since the generator has a discrete generation step, we cannot directly use the gradient-based approach to back-propagate the errors. We then consider a one-step reinforcement learning setting, and use a variance-reduction REINFORCE method to achieve this goal. Empirically, we perform experiments on three common KGE datasets (FB15K-237, WN18 and WN18RR), and verify the adversarial learning approach with a set of KGE models. Our experiments show that across various settings, this adversarial learning mechanism can significantly improve the performance of some of the most commonly used translation based KGE methods. Our contributions are three-fold:

- We are the first to consider adversarial learning to generate useful negative training examples to improve knowledge graph embedding.
- This adversarial learning framework applies to a wide range of KGE models, without the need of external ontologies constraints.
- Our method shows consistent performance gains on three commonly used KGE datasets.

2 Related Work

2.1 Knowledge Graph Embeddings

A large number of knowledge graph embedding models, which represent entities and relations in a knowledge graph with vectors or matrices, have been proposed in recent years. RESCAL (Nickel et al., 2011) is one of the earliest studies on matrix factorization based knowledge graph embedding models, using a bilinear form as score function. TRANSE (Bordes et al., 2013) is the first model to introduce translation-based embedding. Later variants, such as TRANSH (Wang et al., 2014), TRANSR (Lin et al., 2015) and TRANSD (Ji et al., 2015), extend TRANSE by projecting the embedding vectors of entities into various spaces. DISTMULT (Yang et al., 2015) simplifies RESCAL by only using a diagonal matrix, and COMPLEX (Trouillon et al., 2016) extends DISTMULT into the complex number field. (Nickel et al., 2015) is a comprehensive survey on these models.

Some of the more recent models achieve strong performances. MANIFOLDE (Xiao et al., 2016) embeds a triple as a manifold rather than a point. HOLE (Nickel et al., 2016) employs circular correlation to combine the two entities in a triple. CONVE (Dettmers et al., 2017) uses a convolutional neural network as the score function. However, most of these studies use uniform sampling to generate negative training examples (Bordes et al., 2013). Because our framework is independent of the concrete form of models, all these models can be potentially incorporated into our framework, regardless of the complexity. As a proof of principle, our work focuses on simpler models. Table 1 summarizes the score functions and dimensions of all models mentioned above.

2.2 Generative Adversarial Networks and its Variants

Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) was originally proposed for generating samples in a continuous space such as images. A GAN consists of two parts, the *generator* and the *discriminator*. The generator accepts a noise input and outputs an image. The discriminator is a classifier which classifies images as "true" (from the ground truth set) or "fake" (generated by the generator). When training a GAN, the generator and the discriminator play a minimax game, in which the generator tries to generate "real" images to deceive the discriminator, and the discriminator tries to tell them apart from ground truth images. GANs are also capable of generating samples satisfying certain requirements, such as conditional GAN (Mirza and Osindero, 2014).

It is not possible to use GANs in its original form for generating discrete samples like natural language sentences or knowledge graph triples, because the discrete sampling step prevents gradients from propagating back to the generator. SeqGAN (Yu et al., 2017) is one of the first successful solutions to this problem by using reinforcement learning—It trains the generator using policy gradient and other tricks. IRGAN (Wang et al., 2017) is a recent work which combines two categories of information retrieval models into a discrete GAN framework. Likewise, our framework relies on policy gradient to train the generator which provides discrete negative triples.

The discriminator in a GAN is not necessarily a classifier. Wasserstein GAN or WGAN (Arjovsky et al., 2017) uses a regressor with clipped parameters as its discriminator, based on solid analysis about the mathematical nature of GANs. GoGAN (Juefei-Xu et al., 2017) further replaces the loss function in WGAN with marginal loss. Although originating from very different fields, the form of loss function in our framework turns out to be more closely related to the one in GoGAN.

3 Our Approaches

In this section, we first define two types of training objectives in knowledge graph embedding models to show how KBGAN can be applied. Then, we demonstrate a long overlooked problem about negative sampling which motivates us to propose KBGAN to address the problem. Finally, we dive into the mathematical, and algorithmic details of KBGAN.

3.1 Types of Training Objectives

For a given knowledge graph, let $\mathcal{E}$ be the set of entities, $\mathcal{R}$ be the set of relations, and $\mathcal{T}$ be the set of ground truth triples. In general, a knowledge graph embedding (KGE) model can be formulated as a *score function* $f(h,r,t), h,t \in \mathcal{E}, r \in \mathcal{R}$ which assigns a score to every possible triple in the knowledge graph. The estimated likelihood of a triple to be true depends only on its score given by the score function.

Different models formulate their score function based on different designs, and therefore interpret scores differently, which further lead to various training objectives. Two common forms of training objectives are particularly of our interest:

Marginal loss function is commonly used by a large group of models called translation-based models, whose score function models distance between points or vectors, such as TransE, TransH, TransR, TransD and so on. In these models, smaller distance indicates a higher likelihood of truth, but only qualitatively. The marginal loss function takes the following form:

$$L_m = \sum_{(h,r,t)\in\mathcal{T}} [f(h,r,t) - f(h',r,t') + \gamma]_+ \quad (1)$$

where γ is the margin, $[\cdot]_+ = \max(0,\cdot)$ is the hinge function, and (h',r,t') is a negative triple. The negative triple is generated by replacing the head entity or the tail entity of a positive triple with a random entity in the knowledge graph, or formally $(h',r,t') \in \{(h',r,t)|h' \in \mathcal{E}\} \cup \{(h,r,t')|t' \in \mathcal{E}\}$.

Log-softmax loss function is commonly used by models whose score function has probabilistic interpretation. Some notable examples are RESCAL, DistMult, ComplEx. Applying the softmax function on scores of a given set of triples gives the probability of a triple to be the best one among them: $p(h,r,t) = \frac{\exp f(h,r,t)}{\sum_{(h',r,t')} \exp f(h',r,t')}$. The loss function is the negative log-likelihood of this probabilistic model:

$$L_l = \sum_{(h,r,t)\in\mathcal{T}} -\log \frac{\exp f(h,r,t)}{\sum \exp f(h',r,t')}$$
$$(h',r,t') \in \{(h,r,t)\} \cup Neg(h,r,t) \quad (2)$$

where $Neg(h,r,t) \subset \{(h',r,t)|h' \in \mathcal{E}\} \cup \{(h,r,t')|t' \in \mathcal{E}\}$ is a set of sampled corrupted triples.

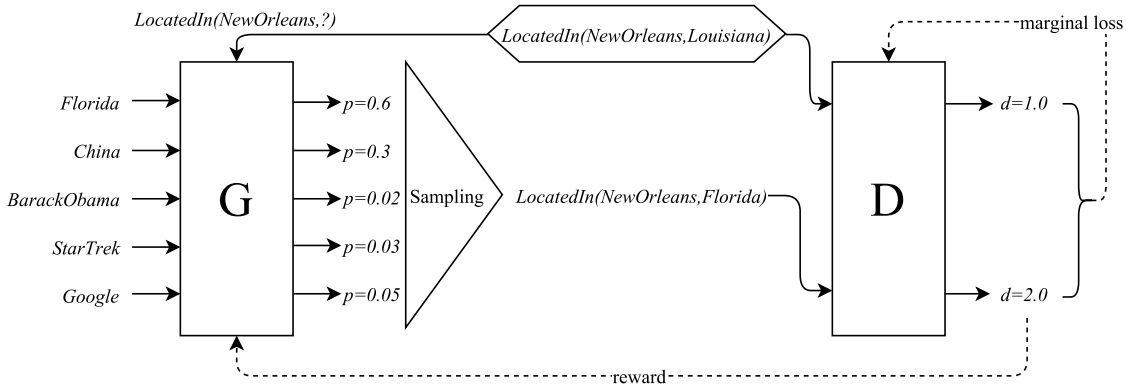


Figure 1: An overview of the KBGAN framework. The generator (G) calculates a probability distribution over a set of candidate negative triples, then sample one triples from the distribution as the output. The discriminator (D) receives the generated negative triple as well as the ground truth triple (in the hexagonal box), and calculates their scores. G minimizes the score of the generated negative triple by policy gradient, and D minimizes the marginal loss between positive and negative triples by gradient descent.

Other forms of loss functions exist, for example ConvE uses a triple-wise logistic function to model how likely the triple is true, but by far the two described above are the most common. Also, softmax function gives an probabilistic distribution over a set of triples, which is necessary for a generator to sample from them.

3.2 Weakness of Uniform Negative Sampling

Most previous KGE models use *uniform negative sampling* for generating negative triples, that is, replacing the head or tail entity of a positive triple with any of the entities in $\mathcal{E}$, all with equal probability. Most of the negative triples generated in this way contribute little to learning an effective embedding, because they are too obviously false.

To demonstrate this issue, let us consider the following example. Suppose we have a ground truth triple *LocatedIn(NewOrleans,Louisiana)*, and corrupt it by replacing its tail entity. First, we remove the tail entity, leaving *LocatedIn(NewOrleans,?)*. Because the relation *LocatedIn* constraints types of its entities, "?" must be a geographical region. If we fill "?" with a random entity $e \in \mathcal{E}$, the probability of e having a wrong type is very high, resulting in ridiculous triples like *LocatedIn(NewOrleans,BarackObama)* or *LocatedIn(NewOrleans,StarTrek)*. Such triples are considered "too easy", because they can be eliminated solely by types. In contrast, *LocatedIn(NewOrleans,Florida)* is a very useful negative triple, because it satisfies type constraints, but it cannot be proved wrong without detailed knowledge of American geography. If a KGE model is fed with mostly "too easy" negative examples, it would probably only learn to represent types, not the underlying semantics.

The problem is less severe to models using log-softmax loss function, because they typically samples tens or hundreds of negative triples for one positive triple in each iteration, and it is likely to have a few useful negatives among them. For instance, (Trouillon et al., 2016) found that a 100:1 negative-to-positive ratio results in the best performance for ComplEx. However, for marginal loss function, whose negative-to-positive ratio is always 1:1, the low quality of uniformly sampled negatives can seriously damage their performance.

3.3 Generative Adversarial Training for Knowledge Graph Embedding Models

Inspired by GANs, we propose an adversarial training framework named KBGAN which uses a KGE model with **softmax probabilities** to provide high-quality negative samples for the training of a KGE model whose training objective is **marginal loss function**. This framework is independent of the score functions of these two models, and therefore possesses some extent of universality. Figure 1 illustrates the overall structure of KBGAN.

In parallel to terminologies used in GAN literature, we will simply call these two models *generator* and *discriminator* respectively in the rest of this paper. We use softmax probabilistic models as the generator because they can adequately model the "sampling from a probability distribu-

Algorithm 1: The KBGAN algorithm

Data: training set of positive fact triples $\mathcal{T} = \{(h, r, t)\}$
Input: Pre-trained generator G with parameters θ_G and score function $f_G(h, r, t)$, and pre-trained discriminator D with parameters θ_D and score function $f_D(h, r, t)$
Output: Adversarially trained discriminator

$b \longleftarrow 0$; // baseline for policy gradient
repeat
  Sample a mini-batch of data $\mathcal{T}_{batch}$ from $\mathcal{T}$;
  $G_G \longleftarrow 0, G_D \longleftarrow 0$; // gradients of parameters of G and D
  $r_{sum} \longleftarrow 0$; // for calculating the baseline
  for $(h, r, t) \in \mathcal{T}_{batch}$ **do**
    Uniformly randomly sample N_s negative triples $Neg(h, r, t) = \{(h'_i, r, t'_i)\}_{i=1...N_s}$;
    Obtain their probability of being generated: $p_i = \frac{\exp f_G(h'_i, r, t'_i)}{\sum_{j=1}^{N_s} \exp f_G(h'_j, r, t'_j)}$;
    Sample one negative triple (h'_s, r, t'_s) from $Neg(h, r, t)$ according to $\{p_i\}_{i=1...N_s}$. Assume its probability to be p_s;
    $G_D \longleftarrow G_D + \nabla_{\theta_D}[f_D(h, r, t) - f_D(h'_s, r, t'_s) + \gamma]_+$; // accumulate gradients for D
    $r \longleftarrow -f_D(h'_s, r, t'_s), r_{sum} \longleftarrow r_{sum} + r$; // r is the reward
    $G_G \longleftarrow G_G + (r - b)\nabla_{\theta_G} \log p_s$; // accumulate gradients for G
  end
  $\theta_G \longleftarrow \theta_G + \eta_G G_G, \theta_D \longleftarrow \theta_D - \eta_D G_D$; // update parameters
  $b \leftarrow r_{sum}/|\mathcal{T}_{batch}|$; // update baseline
until *convergence*;

tion" process of discrete GANs, and we aim at improving discriminators based on marginal loss because they can benefit more from high-quality negative samples. Note that a major difference between GAN and our work is that, the ultimate goal of our framework is to produce a good discriminator, whereas GANS are aimed at training a good generator. In addition, the discriminator here is not a classifier as it would be in most GANs.

Intuitively, the discriminator should assign a relatively small distance to a high-quality negative sample. In order to encourage the generator to generate useful negative samples, the objective of the generator is to minimize the distance given by discriminator for its generated triples. And just like the ordinary training process, the objective of the discriminator is to minimize the marginal loss between the positive triple and the generated negative triple. In an adversarial training setting, the generator and the discriminator are alternatively trained towards their respective objectives.

Suppose that the generator produces a probability distribution on negative triples $p_G(h', r, t'|h, r, t)$ given a positive triple (h, r, t), and generates negative triples (h', r, t') by sampling from this distribution. Let $f_D(h, r, t)$ be the score function of the discriminator. The objective of the discriminator can be formulated as minimizing the following marginal loss function:

$$L_D = \sum_{(h,r,t)\in\mathcal{T}} [f_D(h, r, t) - f_D(h', r, t') + \gamma]_+ \quad (h', r, t') \sim p_G(h', r, t'|h, r, t) \quad (3)$$

The only difference between this loss function and Equation 1 is that it uses negative samples from the generator.

The objective of the generator can be formulated as maximizing the following expectation of negative distances:

$$R_G = \sum_{(h,r,t)\in\mathcal{T}} \mathbb{E}[-f_D(h', r, t')] \quad (h', r, t') \sim p_G(h', r, t'|h, r, t) \quad (4)$$

R_G involves a discrete sampling step, so we cannot find its gradient with simple differentiation. We use a simple special case of Policy Gradient Theorem[1] (Sutton et al., 2000) to obtain the gradient of R_G with respect to parameters of the generator:

$$\begin{aligned}\nabla_G R_G &= \sum_{(h,r,t)\in\mathcal{T}} \mathbb{E}_{(h',r,t')\sim p_G(h',r,t'|h,r,t)} \\ &[-f_D(h', r, t')\nabla_G \log p_G(h', r, t'|h, r, t)] \\ &\simeq \sum_{(h,r,t)\in\mathcal{T}} \frac{1}{N} \sum_{(h'_i,r,t'_i)\sim p_G(h',r,t'|h,r,t), i=1...N} \\ &[-f_D(h', r, t')\nabla_G \log p_G(h', r, t'|h, r, t)] \quad (5)\end{aligned}$$

[1] A proof can be found in the supplementary material

Model	Hyperparameters	Constraints or Regularizations
TRANSE	L_1 distance, $k = 50, \gamma = 3$	$\|\|\mathbf{e}\|\|_2 \leq 1, \|\|\mathbf{r}\|\|_2 \leq 1$
TRANSD	L_1 distance, $k = 50, \gamma = 3$	$\|\|\mathbf{e}\|\|_2 \leq 1, \|\|\mathbf{r}\|\|_2 \leq 1, \|\|\mathbf{e_p}\|\|_2 \leq 1, \|\|\mathbf{r_p}\|\|_2 \leq 1$
DISTMULT	$k = 50, \lambda = 1/0.1$	L2 regularization: $L_{reg} = L + \lambda\|\|\Theta\|\|_2^2$
COMPLEX	$2k = 50, \lambda = 1/0.1$	L2 regularization: $L_{reg} = L + \lambda\|\|\Theta\|\|_2^2$

Table 2: Hyperparameter settings of the 4 models we used. For DISTMULT and COMPLEX, $\lambda = 1$ is used for FB15k-237 and $\lambda = 0.1$ is used for WN18 and WN18RR. All other hyperparameters are shared among all datasets. L is the global loss defined in Equation (2). Θ represents all parameters in the model.

Dataset	#r	#ent.	#train	#val	#test
FB15k-237	237	14,541	272,115	17,535	20,466
WN18	18	40,943	141,442	5,000	5,000
WN18RR	11	40,943	86,835	3,034	3,134

Table 3: Statistics of datasets we used in the experiments. "r": relations.

where the second approximate equality means we approximate the expectation with sampling in practice. Now we can calculate the gradient of R_G and optimize it with gradient-based algorithms.

Policy Gradient Theorem arises from reinforcement learning (RL), so we would like to draw an analogy between our model and an RL model. The generator can be viewed as an *agent* which interacts with the *environment* by performing *actions* and improves itself by maximizing the *reward* returned from the environment in response of its actions. Correspondingly, the discriminator can be viewed as the environment. Using RL terminologies, (h, r, t) is the *state* (which determines what actions the actor can take), $p_G(h', r, t'|h, r, t)$ is the *policy* (how the actor choose actions), (h', r, t') is the *action*, and $-f_D(h', r, t')$ is the *reward*. The method of optimizing R_G described above is called REINFORCE (Williams, 1992) algorithm in RL. Our model is a simple special case of RL, called one-step RL. In a typical RL setting, each action performed by the agent will change its state, and the agent will perform a series of actions (called an *epoch*) until it reaches certain states or the number of actions reaches a certain limit. However, in the analogy above, actions does not affect the state, and after each action we restart with another unrelated state, so each epoch consists of only one action.

To reduce the variance of REINFORCE algorithm, it is common to subtract a *baseline* from the reward, which is an arbitrary number that only depends on the state, without affecting the expectation of gradients.[2] In our case, we replace $-f_D(h', r, t')$ with $-f_D(h', r, t') - b(h, r, t)$ in the equation above to introduce the baseline. To avoid introducing new parameters, we simply let b be a constant, the average reward of the whole training set: $b = \sum_{(h,r,t)\in\mathcal{T}} \mathbb{E}_{(h',r,t')\sim p_G(h',r,t'|h,r,t)}[-f_D(h', r, t')]$. In practice, b is approximated by the mean of rewards of recently generated negative triples.

Let the generator's score function to be $f_G(h, r, t)$, given a set of candidate negative triples $Neg(h, r, t) \subset \{(h', r, t)|h' \in \mathcal{E}\} \cup \{(h, r, t')|t' \in \mathcal{E}\}$, the probability distribution p_G is modeled as:

$$p_G(h', r, t'|h, r, t) = \frac{\exp f_G(h', r, t')}{\sum \exp f_G(h^*, r, t^*)} \quad (h^*, r, t^*) \in Neg(h, r, t) \quad (6)$$

Ideally, $Neg(h, r, t)$ should contain all possible negatives. However, knowledge graphs are usually highly incomplete, so the "hardest" negative triples are very likely to be false negatives (true facts). To address this issue, we instead generate $Neg(h, r, t)$ by uniformly sampling of N_s entities (a small number compared to the number of all possible negatives) from $\mathcal{E}$ to replace h or t. Because in real-world knowledge graphs, true negatives are usually far more than false negatives, such set would be unlikely to contain any false negative, and the negative selected by the generator would likely be a true negative. Using a small $Neg(h, r, t)$ can also significantly reduce computational complexity.

Besides, we adopt the "bern" sampling technique (Wang et al., 2014) which replaces the "1" side in "1-to-N" and "N-to-1" relations with higher probability to further reduce false negatives.

Algorithm 1 summarizes the whole adversarial training process. Both the generator and the dis-

[2] A proof of such fact can also be found in the supplementary material

criminator require pre-training, which is the same as conventionally training a single KBE model with uniform negative sampling. Formally speaking, one can pre-train the generator by minimizing the loss function defined in Equation (1), and pre-train the discriminator by minimizing the loss function defined in Equation (2). Line 14 in the algorithm assumes that we are using the vanilla gradient descent as the optimization method, but obviously one can substitute it with any gradient-based optimization algorithm.

4 Experiments

To evaluate our proposed framework, we test its performance for the link prediction task with different generators and discriminators. For the generator, we choose two classical probability-based KGE model, DISTMULT and COMPLEX, and for the discriminator, we also choose two classical translation-based KGE model, TRANSE and TRANSD, resulting in four possible combinations of generator and discriminator in total. See Table 1 for a brief summary of these models.

4.1 Experimental Settings

4.1.1 Datasets

We use three common knowledge base completion datasets for our experiment: FB15k-237, WN18 and WN18RR. FB15k-237 is a subset of FB15k introduced by (Toutanova and Chen, 2015), which removed redundant relations in FB15k and greatly reduced the number of relations. Likewise, WN18RR is a subset of WN18 introduced by (Dettmers et al., 2017) which removes reversing relations and dramatically increases the difficulty of reasoning. Both FB15k and WN18 are first introduced by (Bordes et al., 2013) and have been commonly used in knowledge graph researches. Statistics of datasets we used are shown in Table 3.

4.1.2 Evaluation Protocols

Following previous works like (Yang et al., 2015) and (Trouillon et al., 2016), for each run, we report two common metrics, mean reciprocal ranking (MRR) and hits at 10 (H@10). We only report scores under the *filtered* setting (Bordes et al., 2013), which removes all triples appeared in training, validating, and testing sets from candidate triples before obtaining the rank of the ground truth triple.

4.1.3 Implementation Details

[3] In the pre-training stage, we train every model to convergence for 1000 epochs, and divide every epoch into 100 mini-batches. To avoid overfitting, we adopt early stopping by evaluating MRR on the validation set every 50 epochs. We tried $\gamma = 0.5, 1, 2, 3, 4, 5$ and L_1, L_2 distances for TRANSE and TRANSD, and $\lambda = 0.01, 0.1, 1, 10$ for DISTMULT and COMPLEX, and determined the best hyperparameters listed on table 2, based on their performances on the validation set after pre-training. Due to limited computation resources, we deliberately limit the dimensions of embeddings to $k = 50$, similar to the one used in earlier works, to save time. We also apply certain constraints or regularizations to these models, which are mostly the same as those described in their original publications, and also listed on table 2.

In the adversarial training stage, we keep all the hyperparamters determined in the pre-training stage unchanged. The number of candidate negative triples, N_s, is set to 20 in all cases, which is proven to be optimal among the candidate set of $\{5, 10, 20, 30, 50\}$. We train for 5000 epochs, with 100 mini-batches for each epoch. We also use early stopping in adversarial training by evaluating MRR on the validation set every 100 epochs.

We use the self-adaptive optimization method Adam (Kingma and Ba, 2015) for all trainings, and always use the recommended default setting $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$.

4.2 Results

Results of our experiments as well as baselines are shown in Table 4. All settings of adversarial training bring a pronounced improvement to the model, which indicates that our method is consistently effective in various cases. TRANSE performs slightly worse than TRANSD on FB15k-237 and WN18, but better on WN18RR. Using DISTMULT or COMPLEX as the generator does not affect performance greatly.

TRANSE and TRANSD enhanced by KBGAN can significantly beat their corresponding baseline implementations, and outperform stronger baselines in some cases. As a prototypical and proof-of-principle experiment, we have never expected state-of-the-art results. Being simple models pro-

[3] The KBGAN source code is available at https://github.com/cai-lw/KBGAN

Method	FB15k-237 MRR	FB15k-237 H@10	WN18 MRR	WN18 H@10	WN18RR MRR	WN18RR H@10
TRANSE	-	42.8†	-	89.2	-	43.2†
TRANSD	-	45.3†	-	92.2	-	42.8†
DISTMULT	24.1‡	41.9‡	82.2	93.6	42.5‡	49.1‡
COMPLEX	24.0‡	41.9‡	**94.1**	94.7	**44.4**‡	**50.7**‡
TRANSE (pre-trained)	24.2	42.2	43.3	91.5	18.6	45.9
KBGAN (TRANSE + DISTMULT)	27.4	45.0	71.0	**94.9**	21.3	48.1
KBGAN (TRANSE + COMPLEX)	**27.8**	45.3	70.5	**94.9**	21.0	47.9
TRANSD (pre-trained)	24.5	42.7	49.4	92.8	19.2	46.5
KBGAN (TRANSD + DISTMULT)	**27.8**	**45.8**	77.2	94.8	21.4	47.2
KBGAN (TRANSD + COMPLEX)	27.7	**45.8**	77.9	94.8	21.5	46.9

Table 4: Experimental results. Results of KBGAN are results of its discriminator (on the left of the "+" sign). Underlined results are the best ones among our implementations. Results marked with † are produced by running Fast-TransX (Lin et al., 2015) with its default parameters. Results marked with ‡ are copied from (Dettmers et al., 2017). All other baseline results are copied from their original papers.

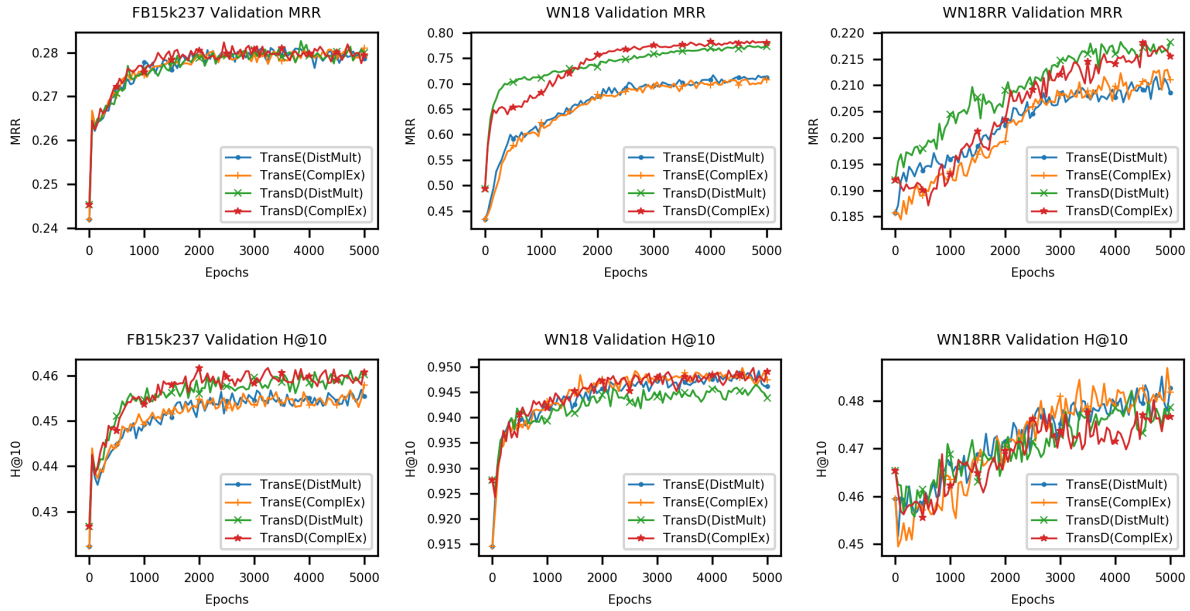


Figure 2: Learning curves of KBGAN. All metrics improve steadily as training proceeds.

posed several years ago, TRANSE and TRANSD has their limitations in expressiveness that are unlikely to be fully compensated by better training technique. In future researches, people may try employing more advanced models into KBGAN, and we believe it has the potential to become state-of-the-art.

To illustrate our training progress, we plot performances of the discriminator on validation set over epochs, which are displayed in Figure 2. As all these graphs show, our performances are always in increasing trends, converging to its maximum as training proceeds, which indicates that KBGAN is a robust GAN that can converge to good results in various settings, although GANs are well-known for difficulty in convergence. Fluctuations in these graphs may seem more prominent than other KGE models, but is considered normal for an adversially trained model. Note that in some cases the curve still tends to rise after 5000 epochs. We do not have sufficient computation resource to train for more epochs, but we believe that they will also eventually converge.

Positive fact	Uniform random sample	Trained generator
(condensation_NN_2, derivationally_related_form, **distill_VB_4**)	family_arcidae_NN_1 repast_NN_1 beater_NN_2 coverall_NN_1 cash_advance_NN_1	revivification_NN_1 mouthpiece_NN_3 *liquid_body_substance_NN_1* stiffen_VB_2 *hot_up_VB_1*
(**colorado_river_NN_2**, instance_hypernym, river_NN_1)	lunar_calendar_NN_1 umbellularia_californica_NN_1 tonality_NN_1 creepy-crawly_NN_1 moor_VB_3	*idaho_NN_1* *sayan_mountains_NN_1* *lower_saxony_NN_1* order_ciconiiformes_NN_1 jab_NN_3
(**meeting_NN_2**, hypernym, social_gathering_NN_1)	cellular_JJ_1 commercial_activity_NN_1 giant_cane_NN_1 streptomyces_NN_1 tranquillize_VB_1	*attach_VB_1* *bond_NN_6* heavy_spar_NN_1 satellite_NN_1 peep_VB_3

Table 5: Examples of negative samples in WN18 dataset. The first column is the positive fact, and the term in bold is the one to be replaced by an entity in the next two columns. The second column consists of random entities drawn from the whole dataset. The third column contains negative samples generated by the generator in the last 5 epochs of training. Entities in italic are considered to have semantic relation to the positive one

4.3 Case study

To demonstrate that our approach does generate better negative samples, we list some examples of them in Table 5, using the KBGAN (TRANSE + DISTMULT) model and the WN18 dataset. All hyperparameters are the same as those described in Section 4.1.3.

Compared to uniform random negatives which are almost always totally unrelated, the generator generates more *semantically* related negative samples, which is different from type relatedness we used as example in Section 3.2, but also helps training. In the first example, two of the five terms are physically related to the process of distilling liquids. In the second example, three of the five entities are geographical objects. In the third example, two of the five entities express the concept of "gather".

Because we deliberately limited the strength of generated negatives by using a small N_s as described in Section 3.3, the semantic relation is pretty weak, and there are still many unrelated entities. However, empirical results (when selecting the optimal N_s) shows that such situation is more beneficial for training the discriminator than generating even stronger negatives.

5 Conclusions

We propose a novel adversarial learning method for improving a wide range of knowledge graph embedding models—We designed a generator-discriminator framework with dual KGE components. Unlike random uniform sampling, the generator model generates higher quality negative examples, which allow the discriminator model to learn better. To enable backpropagation of error, we introduced a one-step REINFORCE method to seamlessly integrate the two modules. Experimentally, we tested the proposed ideas with four commonly used KGE models on three datasets, and the results showed that the adversarial learning framework brought consistent improvements to various KGE models under different settings.

References

Martin Arjovsky, Soumith Chintala, and Leon Bottou. 2017. Wasserstein gan. In *International Conference on Machine Learning*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*. pages 2787–2795.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2017. Convolutional 2d knowledge graph embeddings. *arXiv preprint arXiv:1707.01476* .

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 601–610.

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

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *The 53rd Annual Meeting of the Association for Computational Linguistics*.

Felix Juefei-Xu, Vishnu Naresh Boddeti, and Marios Savvides. 2017. Gang of gans: Generative adversarial networks with maximum margin ranking. *arXiv preprint arXiv:1704.04865* .

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *The 3rd International Conference on Learning Representations*.

Denis Krompaß, Stephan Baier, and Volker Tresp. 2015. Type-constrained representation learning in knowledge graphs. In *International Semantic Web Conference*. Springer, pages 640–655.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *The Twenty-ninth AAAI Conference on Artificial Intelligence*. pages 2181–2187.

Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.01784* .

Tom M Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Justin Betteridge, Andrew Carlson, Bhavana Dalvi Mishra, Matthew Gardner, Bryan Kisiel, Jayant Krishnamurthy, et al. 2015. Never-ending learning. In *The Twenty-ninth AAAI Conference on Artificial Intelligence*.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2015. A review of relational machine learning for knowledge graphs. *arXiv preprint arXiv:1503.00759* .

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio Poggio. 2016. Holographic embeddings of knowledge graphs. In *The Thirtieth AAAI Conference on Artificial Intelligence*. pages 1955–1961.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on Machine Learning*. pages 809–816.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*. ACM, pages 697–706.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*. pages 1057–1063.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*. pages 57–66.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*. pages 2071–2080.

Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *The 40th International ACM SIGIR Conference*.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *The Twenty-eighth AAAI Conference on Artificial Intelligence*. pages 1112–1119.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016. From one point to a manifold: Knowledge graph embedding for precise link prediction. In *The Twenty-Fifth International Joint Conference on Artificial Intelligence*.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. *The 3rd International Conference on Learning Representations* .

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *The Thirty-First AAAI Conference on Artificial Intelligence*. pages 2852–2858.

Multimodal Frame Identification with Multilingual Evaluation

Teresa Botschen[§†], **Iryna Gurevych**[*§†], **Jan-Christoph Klie**[*†],
Hatem Mousselly-Sergieh[*†], **Stefan Roth**[*§‡]
§ Research Training Group AIPHES
† Ubiquitous Knowledge Processing (UKP) Lab
‡ Visual Inference Lab
Department of Computer Science, Technische Universität Darmstadt
www.{aiphes, ukp, visinf}.tu-darmstadt.de

Abstract

An essential step in FrameNet Semantic Role Labeling is the Frame Identification (FrameId) task, which aims at disambiguating a situation around a predicate. Whilst current FrameId methods rely on textual representations only, we hypothesize that FrameId can profit from a richer understanding of the situational context. Such contextual information can be obtained from common sense knowledge, which is more present in images than in text. In this paper, we extend a state-of-the-art FrameId system in order to effectively leverage multimodal representations. We conduct a comprehensive evaluation on the English FrameNet and its German counterpart SALSA. Our analysis shows that for the German data, textual representations are still competitive with multimodal ones. However on the English data, our multimodal FrameId approach outperforms its unimodal counterpart, setting a new state of the art. Its benefits are particularly apparent in dealing with ambiguous and rare instances, the main source of errors of current systems. For research purposes, we release *(a)* the implementation of our system, *(b)* our evaluation splits for SALSA 2.0, and *(c)* the embeddings for synsets and IMAGINED words.[1]

1 Introduction

FrameNet Semantic Role Labeling analyzes sentences with respect to frame-semantic structures based on FrameNet (Fillmore et al., 2003). Typically, this involves two steps: First, Frame Identification (FrameId), capturing the context around a predicate (*frame evoking element*) and assigning a frame, basically a word sense label for a prototypical situation, to it. Second, Role Labeling, i.e. identifying the participants (*fillers*) of the predicate and connecting them with predefined frame-specific role labels. FrameId is crucial to the success of Semantic Role Labeling as FrameId errors account for most wrong predictions in current systems (Hartmann et al., 2017). Consequently, improving FrameId is of major interest.

The main challenge and source of prediction errors of FrameId systems are ambiguous predicates, which can evoke several frames, e.g., the verb *sit* evokes the frame *Change_posture* in a context like *'a person is sitting back on a bench'*, while it evokes *Being_located* when *'a company is sitting in a city'*. Understanding the predicate context, and thereby the context of the situation (here, *'Who / what is sitting where?'*), is crucial to identifying the correct frame for ambiguous cases.

State-of-the-art FrameId systems model the situational context using pretrained distributed word embeddings (see Hermann et al., 2014). Hence, it is assumed that the context of the situation is explicitly expressed in words. However, language understanding involves implicit knowledge, which is not mentioned but still seems obvious to humans, e.g., *'people can sit back on a bench, but companies cannot'*, *'companies are in cities'*. Such implicit common sense knowledge is obvious enough to be rarely expressed in sentences, but is more likely to be present in images. Figure 1 takes the ambiguous predicate *sit* to illustrate

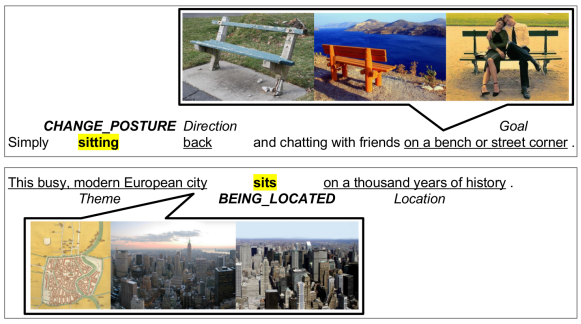


Figure 1: Example sentences demonstrating the potential benefit of images for ambiguous predicates.

[*] named alphabetically
[1] https://github.com/UKPLab/naacl18-multimodal-frame-identification

Proceedings of NAACL-HLT 2018, pages 1481–1491
New Orleans, Louisiana, June 1 - 6, 2018.

how images can provide access to implicit common sense knowledge crucial to FrameId.

When looking at the semantics of events, FrameId has commonalities with event prediction tasks. These aim at linking events and their participants to script knowledge and at predicting events in narrative chains. Ahrendt and Demberg (2016) argue that knowing about the participants helps to identify the event, which suggests the need for implicit context knowledge also for FrameId. This specifically applies to images, which can reflect properties of the participants of a situation in a inherently different way, see Fig. 1.

We analyze whether multimodal representations grounded in images can encode common sense knowledge to improve FrameId. To that end, we extend SimpleFrameId (Hartmann et al., 2017), a recent FrameId model based on distributed word embeddings, to the multimodal case and evaluate for English and German. Note that there is a general lack of evaluation of FrameId systems for languages other than English. This is problematic as they yield different challenges; German, for example, due to long distance dependencies. Also, word embeddings trained on different languages have different strengths in ambiguous words. We elaborate on insights from using different datasets by language.

Contributions. *(1)* We propose a pipeline and architecture of a FrameId system, extending state-of-the-art methods with the option of using implicit multimodal knowledge. It is flexible toward modality and language, reaches state-of-the-art accuracy on English FrameId data, clearly outperforming several baselines, and sets a new state of the art on German FrameId data. *(2)* We discuss properties of language and meaning with respect to implicit knowledge, as well as the potential of multimodal representations for FrameId. *(3)* We perform a detailed analysis of FrameId systems. First, we develop a new strong baseline. Second, we suggest novel evaluation metrics that are essential for assessing ambiguous and rare frame instances. We show our system's advantage over the strong baseline in this regard and by this improve upon the main source of errors. Third, we analyze gold annotated datasets for English and German showing their different strengths. Finally, we release the implementation of our system, our evaluation splits for SALSA 2.0, and the embeddings for synsets and IMAGINED words.

2 Related Work

2.1 Frame identification

State-of-the-art FrameId systems rely on pretrained word embeddings as input (Hermann et al., 2014). This proved to be helpful: those systems consistently outperform the previously leading FrameId system SEMAFOR (Das et al., 2014), which is based on a handcrafted set of features. The open source neural network-based FrameId system SimpleFrameId (Hartmann et al., 2017) is conceptually simple, yet yields competitive accuracy. Its input representation is a concatenation of the predicate's pretrained embedding and an embedding of the predicate context. The dimension-wise mean of the pretrained embeddings of all words in the sentence is taken as the context. In this work, we first aim at improving the representation of the predicate context using multimodal embeddings, and second at assessing the applicability to another language, namely German.

Common sense knowledge for language understanding. Situational background knowledge can be described in terms of frames (Fillmore, 1985) and scripts (Schank and Abelson, 2013). Ahrendt and Demberg (2016) report that knowing about a script's participants aids in predicting events linked to script knowledge. Transferring this insight to FrameId, we assume that a rich context representation helps to identify the sense of ambiguous predicates. Addressing ambiguous predicates where participants have different properties depending on the context, Feizabadi and Padó (2012) give some examples where the location plays a discriminating role as participant: motion verbs that have both a concrete motion sense and a more abstract sense in the cognitive domain, e.g., *struggle*, *lean*, *follow*.

Frame identification in German. Shalmaneser (Erk and Pado, 2006) is a toolbox for semantic role assignment on FrameNet schemata of English and German (integrated into the SALSA project for German). Shalmaneser uses a Naive Bayes classifier to identify frames, together with features for a bag-of-word context with a window over sentences, bigrams, and trigrams of the target word and dependency annotations. They report an F1 of 75.1 % on FrameNet 1.2 and 60 % on SALSA 1.0. These scores are difficult to compare against more recent work as the evaluation uses older versions of datasets and custom splits. Shalmaneser

requires software dependencies that are not available anymore, hindering application to new data. To the best of our knowledge, there is no FrameId system evaluated on SALSA 2.0.

Johannsen et al. (2015) present a simple, but weak translation baseline for cross-lingual FrameId. A SEMAFOR-based system is trained on English FrameNet and tested on German Wikipedia sentences, translated word-by-word to English. This translation baseline reaches an F1 score of 8.5 % on the German sentences when translated to English. The performance of this weak translation baseline is worse than that of another simple baseline: a 'most frequent sense baseline' – computing majority votes for German (and many other languages) – reaches an F1 score of 53.0 % on the German sentences. This shows that pure translation does not help with FrameId and, furthermore, indicates a large room for improvement for FrameId in languages other than English.

2.2 Multimodal representation learning

There is a growing interest in Natural Language Processing for enriching traditional approaches with knowledge from the visual domain, as images capture qualitatively different information compared to text. Regarding FrameId, to the best of our knowledge, multimodal approaches have not yet been investigated. For other tasks, multimodal approaches based on pretrained embeddings are reported to be superior to unimodal approaches. Textual embeddings have been enriched with information from the visual domain, e.g., for Metaphor Identification (Shutova et al., 2016), Question Answering (Wu et al., 2017), and Word Pair Similarity (Collell et al., 2017). The latter presents a simple, but effective way of extending textual embeddings with so-called multimodal IMAGINED embeddings by a learned mapping from language to vision. We apply the IMAGINED method to our problem.

In this work, we aim to uncover whether representations that are grounded in images can help to improve the accuracy of FrameId. Our application case of FrameId is more complex than a comparison on the word-pair level as it considers a whole sentence in order to identify the predicate's frame. However, we see a potential for multimodal IMAGINED embeddings to help: their mapping from text to multimodal representations is learned from images for nouns. Such nouns, in turn, are candidates for role fillers of predicates. In order to identify the correct sense of an ambiguous predicate, it could help to enrich the representation of the context situation with multimodal embeddings for the entities that are linked by the predicate.

3 Our Multimodal FrameId Model

Our system builds upon the SimpleFrameId (Hartmann et al., 2017) system for English FrameId based on textual word embeddings. We extend it to multimodal and multilingual use cases; see Fig. 2 for a sketch of the system pipeline. Same as SimpleFrameId, our system is based on pretrained embeddings to build the input representation out of the predicate context and the predicate itself.

However, different to SimpleFrameId, our representation of the predicate context is multimodal: beyond textual embeddings we also use IMAGINED and visual embeddings. More precisely, we concatenate all unimodal representations of the predicate context, which in turn are the unimodal mean embeddings of all words in the sentence. We use concatenation for fusing the different embeddings as it is the simplest yet successful fusion approach (Bruni et al., 2014; Kiela and Bottou, 2014). The input representation is processed by a two-layer Multilayer Perceptron (MLP, Rosenblatt, 1958), where we adapt the number of hidden nodes to the increased input size and apply dropout to all hidden layers to prevent overfitting (Srivastava et al., 2014). Each node in the output layer corresponds to one frame-label class. We use rectified linear units (Nair and Hinton, 2010) as activation function for the hidden layers, and a soft-

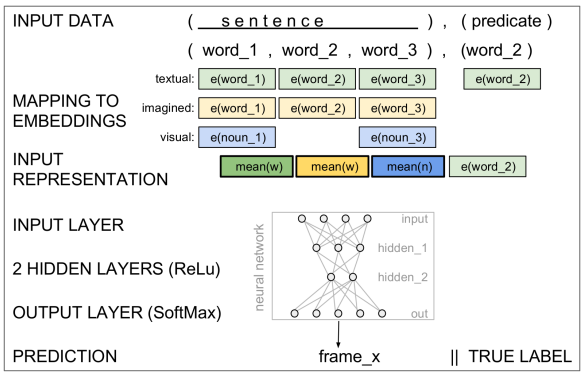


Figure 2: Sketch of the pipeline. *(1)* Data: sentence with predicate. *(2)* Mapping: words to embeddings. *(3)* Representation: concatenation of modality-specific means. *(4)* Classifier: neural network predicting frame.

max for the output layer yielding a multinomial distribution over frames. We take its $\arg\max$ as the final prediction at test time. Optionally, filtering based on the lexicon can be performed on the predicted probabilities for each frame label. The development set was used to determine the architecture and hyperparameters, see Sec. 6.

Majority baselines. We propose a new strong baseline based on a combination of two existing ones. These are: first, the most-frequent-sense baseline using the data majority (Data Baseline) to determine the most frequent frame for a predicate; second, the baseline introduced by Hartmann et al. (2017) using a lexicon (Lexicon Baseline) to consider the data counts of the Data Baseline only for those frames available for a predicate. We propose to combine them into a Data-Lexicon Baseline, which uses the lexicon for unambiguous predicates and for ambiguous ones it uses the data majority. This way, we trust the lexicon for unambiguous predicates but not for ambiguous ones, there we rather consider the data majority. Comparing a system to these baselines helps to see whether it just memorizes the data majority or the lexicon, or actually captures more.

All majority baselines strongly outperform the weak translation baseline of Johannsen et al. (2015) when training the system on English data and evaluating it on German data.

4 Preparation of Input Embeddings

Textual embeddings for words. We use the 300-dimensional GloVe embeddings (Pennington et al., 2014) for English, and the 100-dimensional embeddings of Reimers et al. (2014) for German. GloVe and Reimers have been trained on the Wikipedia of their targeted language and on additional newswire text to cover more domains, resulting in similarly low out-of-vocabulary scores.

Visual embeddings for synsets. We obtain visual embeddings for WordNet synsets (Fellbaum, 1998; , Ed.): we apply the pretrained VGG-m-128 Convolutional Neural Network model (Chatfield et al., 2014) to images for synsets from ImageNet (Deng et al., 2009), we extract the 128-dimensional activation of the last layer (before the softmax) and then we L_2-normalize it. We use the images of the WN9-IMG dataset (Xie et al., 2017), which links WordNet synsets to a collection of ten ImageNet images. We average the embeddings of all images corresponding to a synset, leading to a vocabulary size of 6555 synsets. All synsets in WN9-IMG are part of triples of the form entity-relation-entity, i.e. synset-relation-synset. Such synset entities that are participants of relations with other synset entities are candidates for incorporating the role fillers for predicates and, therefore, may help to find the correct frame for a predicate (see Sec. 5 for details about sense-disambiguation.)

Linguistic embeddings for synsets. We obtain 300-dimensional linguistic synset embeddings: we apply the AutoExtend approach (Rothe and Schütze, 2015) to GloVe embeddings and produce synset embeddings for all synsets having at least one synset lemma in the GloVe embeddings. This leads to a synset vocabulary size of 79 141. Linguistic synset embeddings are based on textual word embeddings and the synset information known by the knowledge base WordNet, thus they complement the visual synset embeddings.

IMAGINED embeddings for words. We use the IMAGINED method (Collell et al., 2017) for learning a mapping function: it maps from the word embedding space to the visual embedding space given those words that occur in both pretrained embedding spaces (7220 for English and 7739 for German). To obtain the English synset lemmas, we extract all lemmas of a synset and keep those that are nouns. We automatically translate English nouns to German nouns using the Google Translate API to obtain the corresponding German synset lemmas. The IMAGINED method is promising for cases where one embedding space (here, the textual one) has many instances without correspondence in the other embeddings space (here, the visual one), but the user still aims at obtaining instances of the first in the second space. We aim to obtain visual correspondences for the textual embeddings in order to incorporate regularities from images into our system. The mapping is a nonlinear transformation using a simple neural network. The objective is to minimize the cosine distance between each mapped representation of a word and the corresponding visual representation. Finally, a multimodal representation for any word can be obtained by applying this mapping to the word embedding.

5 Data and Preparation of Splits

English FrameId: Berkeley FrameNet. The Berkeley FrameNet (Baker et al., 1998; Ruppenhofer et al., 2016) is an ongoing project for building a large lexical resource for English with expert annotations based on frame semantics (Fillmore, 1976). It consists of two parts, a manually created lexicon that maps predicates to the frames they can evoke, and fully annotated texts (fulltext). The mapping can be used to facilitate the frame identification for a predicate in a sentence, e.g., a sentence in the fulltext corpus. Table 1 contains the lexicon statistics, Table 2 (top left) the dataset statistics. In this work, we use FrameNet 1.5 to ensure comparability with the previous state of the art, with the common evaluation split for FrameId systems introduced by Das and Smith (2011) (with the development split of Hermann et al., 2014). Due to having a single annotation as consent of experts, it is hard to estimate a performance bound of a single human for the fulltext annotation.

German FrameId: SALSA. The SALSA project (Burchardt et al., 2006; Rehbein et al., 2012) is a completed annotation project, which serves as the German counterpart to FrameNet. Its annotations are based on FrameNet up to version 1.2. SALSA adds proto-frames to properly annotate senses that are not covered by the English FrameNet. For a more detailed description of differences between FrameNet and SALSA, see Ellsworth et al. (2004); Burchardt et al. (2009). SALSA also provides a lexicon (see Table 1 for statistics) and fully annotated texts. There are two releases of SALSA: 1.0 (Burchardt et al., 2006) used for Shalmaneser (Erk and Pado, 2006) (cf. Sec. 2.1), and the final release 2.0 (Rehbein et al., 2012), which contains more annotations and adds nouns as predicates. We use the final release.

SALSA has no standard evaluation split; Erk and Pado (2006) used an undocumented random split. Also, it is not possible to follow the splitting method of Das and Smith (2011), as SALSA project distributions do not map to documents. We suggest splitting based on sentences, i.e. all annotations of a sentence are in the same set to avoid mixing training and test sets. We assign sentences to 100 buckets based on their IDs and create a 70/15/15 split for training, development, and test sets based on the bucket order. This procedure allows future work to be evaluated on the same data. Table 2 (bottom left) shows the dataset statistics.

Synsets in FrameNet and SALSA. To prepare the datasets for working with the synset embeddings, we sense-disambiguate all sentences using the API of BabelNet (Navigli and Ponzetto, 2010), which returns multilingual synsets. We thus depend on the state-of-the-art accuracy of BabelNet (Navigli and Ponzetto, 2012) when using synset embeddings on sense-disambiguated sentences. However, this dependence does not hold when applying IMAGINED embeddings to sentences, as the mapping from words to IMAGINED embeddings does not need any synsets labeled in the sentences. After sense-disambiguation some sentences do not contain any synset available in our synset embeddings. The statistics of those sentences that have at least one synset embedding (visual or linguistic AutoExtend) is given in Table 2 (right).

6 Experimental Setup

We contrast our system's performance for context representations based on unimodal (textual) versus multimodal (textual and visual) embeddings. Also, we compare English against German data. We run the prediction ten times to reduce noise in

lexicon	**frames**	**LUs**	**avg**(fr/pred)	**%amb.pred.**
FrameNet	1020	11 942	1.26	17.32
SALSA	1023	1827	2.82	57.56

Table 1: Lexicon statistics for FrameNet 1.5 and for SALSA 2.0: the total number of distinct **frames** and lexical units **LUs** (distinct predicate-frame combinations), the number of frames a predicate can evoke on average **avg**, and the % of **amb**iguous **pred**icates.

		sentences	**frames**	reduced sentences	
				syns-Vis	syns-AutoExt
FrameN	train	2819	15 406	1310	2714
	dev	707	4593	320	701
	test	2420	4546	913	2318
SALSA	train	16 852	26 081	4707	16 736
	dev	3561	5533	1063	3540
	test	3605	5660	1032	3570

Table 2: Dataset statistics for FrameNet 1.5 fulltext with Das split and for SALSA 2.0 with our split: number of **sentences** and **frames** (as used in our experiments). Right half (only used in further investigations): number of sentences when reduced to only those having synsets in the visual and in the linguistic AutoExtend embeddings.

the evaluation (cf. Reimers and Gurevych, 2017) and report the mean for each metric.

Use of lexicon. We evaluate our system in two settings: with and without lexicon, as suggested by Hartmann et al. (2017). In the with-lexicon setting, the lexicon is used to reduce the choice of frames for a predicate to only those listed in the lexicon. If the predicate is not in the lexicon, it corresponds to the without-lexicon setting, where the choice has to be done amongst all frames.

Evaluation metrics. FrameId systems are usually compared in terms of *accuracy*, which we adopt for comparability. As a multiclass classification problem, FrameId has to cope with a strong variation in the annotation frequency of frame classes. Minority classes are frames that occur only rarely; majority classes occur frequently. Note that the accuracy is biased toward majority classes, explaining the success of majority baselines on imbalanced datasets such as FrameNet.

Alternatively, the *F1 score* is sometimes reported as it takes a complementary perspective. The F-measure is the harmonic mean of precision and recall, measuring exactness and completeness of a model, respectively. In previous work, micro-averaging is used to compute F1 scores. Yet, similar to the accuracy, micro-averaging introduces a bias toward majority classes. We compute *F1-macro* instead, for which precision and recall are computed for each class and averaged afterwards, giving equal weight to all classes.

Taken together, this yields scores that underestimate (F1-macro) and overestimate (average accuracy) on imbalanced datasets. Previous work just used the overestimate such that a comparison is possible in terms of accuracy in the with-lexicon setting. We suggest to use F1-macro additionally to analyze rare, but interesting classes. Thus, a comparison within our work is possible for both aspects, giving a more detailed picture. Note that previous work reports one score whilst we report the mean score of ten runs.

Hyperparameters. We identified the best hyperparameters for the English and German data based on the respective development sets.[2] The Multilayer Perceptron architecture performed consistently better than a more complex Gated Recurrent Unit model (Cho et al., 2014). We found that more than two hidden layers did not bring any improvement over two layers; using dropout on the hidden layers helped to increase the accuracy. Among the various input representations, a concatenation of the representations of context and predicate was the best amongst others, including dependencies, lexicon indicators, and part-of-speech tags. Training is done using Nesterov-accelerated Adam (Nadam, Dozat, 2016) with default parameters. A batch size of 128 is used. Learning stops if the development accuracy has not improved for four epochs, and the learning rate is reduced by factor of two if there has not been any improvement for two epochs.

[2]Differences in hyperparameters to SimpleFrameId: 'nadam' as optimizer instead of 'adagrad', dropout on hidden layers and early stopping to regularize training. Different number of hidden units, optimized by grid search.

7 Results

First, we report our results on English data (see Table 3, top) and then, we compare against German data (see Table 3, bottom).

7.1 English FrameNet data

Baseline. Our new strong Data-Lexicon Baseline reaches a considerable accuracy of 86.32 %, which is hard to beat by trained models. Even the most recent state of the art only beats it by about two points: 88.41 % (Hermann et al., 2014). However, the accuracy of the baseline drops for ambiguous predicates (69.73 %) and the F1-macro score reveals its weakness toward minority classes (drop from 64.54 % to 37.42 %).

Unimodal. Our unimodal system trained and evaluated on English data slightly exceeds the accuracy of the previous state of the art (88.66 % on average versus 88.41 % for Hermann et al., 2014); our best run's accuracy is 89.35 %. Especially on ambiguous predicates, i.e. the difficult and therefore interesting cases, our average accuracy surpasses that of previous work by more than one point (the best run by almost three points). Considering the proposed F1-macro score for an assessment of the performance on minority classes and ambiguous predicates reveals our main improvement: Our system substantially outperforms the strong Data-Lexicon Baseline, demonstrating that our system differs from memorizing majorities and actually improves minority cases.

Multimodal. From a range of multimodal context representations as extensions to our system,

	model	with lexicon acc	with lexicon acc_amb	with lexicon F1-m	with lexicon F1-m_amb	without lexicon acc	without lexicon acc_amb	without lexicon F1-m	without lexicon F1-m_amb
FrameNet	Data Baseline	79.06	69.73	33.00	37.42	79.06	69.73	33.00	37.42
	Lexicon Baseline	79.89	55.52	65.61	30.95	–	–	–	–
	Data-Lexicon Baseline	86.32	69.73	64.54	37.42	–	–	–	–
	Hermann et al. (2014)	88.41	73.10	–	–	–	–	–	–
	Hartmann et al. (2017)	87.63	73.80	–	–	77.49	–	–	–
	our_uni	88.66	74.92	76.65	53.86	79.96	71.70	57.07	47.40
	our_mm (im, synsV)	**88.82**	**75.28**	**76.77**	**54.80**	**81.21**	**72.51**	**57.81**	**49.38**
SALSA	Data Baseline	77.00	70.51	37.40	28.87	77.00	70.51	37.40	28.87
	Lexicon Baseline	61.57	52.5	19.36	15.68	–	–	–	–
	Data-Lexicon Baseline	77.16	70.51	38.48	28.87	–	–	–	–
	our_uni	**80.76**	**75.59**	**48.42**	**41.38**	**80.59**	**75.52**	**47.64**	**41.17**
	our_mm (im)	80.71	75.58	48.29	41.19	80.51	75.51	47.36	40.93

Table 3: FrameId results (in %) on English (upper) and German (lower) with and without using the lexicon. Reported are **acc**uracy and **F1-m**acro, both also for **amb**iguous predicates (mean scores over ten runs). Models: *(a)* Data, Lexicon, and Data-Lexicon Baselines. *(b)* Previous models for English. *(c)* Ours: unimodal **our**-uni, multimodal on top of our-uni – **our**-mm – with IMAGINED embeddings (and synset visual embeddings for English). Best results highlighted in bold. The best run's results for English were:
our_uni: **acc**: 89.35 ; **acc_amb**: 76.45 ; **F1-m**: 76.95 ; **F1-m_amb**: 54.02 (with lexicon)
our_mm (im, synsV): **acc**: 89.09 ; **acc_amb**: 75.86 ; **F1-m**: 78.17 ; **F1-m_amb**: 57.48 (with lexicon)

the most helpful one is the concatenation of IMAGINED embeddings and visual synset embeddings: it outperforms the unimodal approach slightly in all measurements. We observe that the improvements are more pronounced for difficult cases, such as for rare and ambiguous cases (one point improvement in F1-macro), as well as in the absence of a lexicon (up to two points improvement).

Significance tests. We conduct a single sample t-test to judge the difference between previous state-of-the-art accuracy (Hermann et al., 2014) and our unimodal approach. The null hypothesis (expected value of our sample of ten accuracy scores equals previous state-of-the-art accuracy) is rejected at a significance level of $\alpha = 0.05$ ($p = 0.0318$). In conclusion, even our unimodal approach outperforms prior state of the art in terms of accuracy.

To judge the difference between our unimodal and our multimodal approach, we conduct a t-test for the means of the two independent samples. The null hypothesis states identical expected values for our two samples of ten accuracy scores. Regarding the setting with lexicon, the null hypothesis cannot be rejected at a significance level of $\alpha = 0.05$ ($p = 0.2181$). However, concerning accuracy scores without using the lexicon, the null hypothesis is rejected at a significance level of $\alpha = 0.05$ ($p < 0.0001$). In conclusion, the multimodal approach has a slight overall advantage and, interestingly, has a considerable advantage over the unimodal one when confronted with a more difficult setting of not using the lexicon.

7.2 German SALSA versus English data

German results. Our system evaluated on German data sets a new state of the art on this corpus with 80.76 % accuracy, outperforming the baselines (77.16 %; no other system evaluated on this dataset). The difference in F1-macro between the majority baselines and our system is smaller than for the English FrameNet. This indicates that the majorities learned from data are more powerful in the German case with SALSA than in the English case, when comparing against our system. Multimodal context representations cannot show an improvement for SALSA with this general dataset.

Lexicon. We report results achieved without the lexicon to evaluate independently of its quality (Hartmann et al., 2017). On English data, our systems outperforms Hartmann et al. (2017) by more than two points in accuracy and we achieve a large improvement over the Data Baseline. Comparing the F1-macro with and without lexicon, it can be seen that the additional information stored in the lexicon strongly increases the score by about 20 points for English data. For German data, the increase of F1-macro with lexicon versus without is small (one point).

8 Discussion

8.1 English data

Insights from the baseline. Many indicators point to our approach not just learning the data majority: our trained models have better F1-macro and especially much higher ambiguous F1-macro scores with lexicon. This clearly suggests that our system is capable of acquiring more expressiveness than the baselines do by counting majorities.

Impact of multimodal representations. Multimodal context representations improve results compared to unimodal ones. It helps to incorporate visual common sense knowledge about the situation's participants. Referring back to our example of the ambiguous predicate *sit*, the multimodal approach is able to transfer the knowledge to the test sentence *'Al-Anbar in general, and Ramadi in particular, are sat with the Americans in Jordan.'* by correctly identifying the frame *Being_located* whilst the unimodal approach fails with predicting *Change_posture*. The increase in performance when adding information from visual synset embeddings is not simply due to higher dimensionality of the embedding space. To verify, we further investigate extending the unimodal system with random word embeddings. This leads to a drop in performance compared to using just the unimodal representations or using these in combination with the proposed multimodal embeddings, especially in the setting without lexicon. Interestingly, replacing visual synset embeddings with linguistic synset embeddings (AutoExtend by Rothe and Schütze (2015), see Sec. 4) in further investigations also showed that visual embeddings yield better performance. This points out the potential for incorporating even more image evidence to extend our approach.

8.2 German versus English data

Difficulties for German data. The impact of multimodal context representations is more difficult to interpret for the German dataset. The fact that they have not helped here may be due to mismatches when translating the English nouns of a synset to German in order to train the IMAGINED embeddings. Here, we see room for future work to improve on simple translation by sense-based translations. In SALSA, a smaller portion of sentences has at least one synset embedding, see Table 2. For further investigations, we reduced the dataset to only sentences actually containing a synset embedding. Then, minor improvements of the multimodal approach were visible for SALSA. This points out that a dataset containing more words linking to implicit knowledge in images (visual synset embeddings) can profit more from visual and IMAGINED embeddings.

Impact of lexicon: English versus German. Even if both lexica approximately define the same number of frames (see Table 1), the number of defined lexical units (distinct predicate-frame combinations) in SALSA is smaller. This leads to a lexicon that is a magnitude smaller than the FrameNet lexicon. Thus, the initial situation for the German case is more difficult. The impact of the lexicon for SALSA is smaller than for FrameNet (best visible in the increase of F1-macro with using the lexicon compared to without), which can be explained by the larger percentage of ambiguous predicates (especially evoking proto-frames) and the smaller size of the lexicon. The evaluation on two different languages highlights the impact of an elaborate, manually created lexicon: it boosts the performance on frame classes that are less present in the training data. English FrameId benefits from the large high-quality lexicon, whereas German FrameId currently lacks a high-quality lexicon that is large enough to benefit the FrameId task.

Dataset properties: English versus German. To better understand the influence of the dataset on the prediction errors, we further analyze the errors of our approach (see Table 4) following Palmer

		with lexicon				without lexicon			
	model	**correct**	**e**_uns	**e**_unsLab	**e**_n	**correct**	**e**_uns	**e**_unsLab	**e**_n
FrameN	**our**_uni	89.35	0.40	3.04	7.22	80.36	1.32	7.68	10.65
	our_mm (im, synsV)	89.79	0.58	3.55	6.08	80.63	1.91	8.50	8.96
SALSA	**our**_uni	80.99	0.49	0.97	17.54	80.80	0.49	1.10	17.61
	our_mm (im)	81.24	1.94	1.88	14.94	80.96	1.94	2.05	15.05

Table 4: Error analysis of best uni- and multimodal systems. **correct**, **e**rrors: unseen, unseen label and normal.

and Sporleder (2010). A wrong prediction can either be a normal classification error, or it can be the result of an instance that was unseen at training time, which means that the error is due to the training set. The instance can either be completely unseen or unseen with the target label. We observe that FrameNet has larger issues with unseen data compared to SALSA, especially data that was unseen with one specific label but seen with another label. This is due to the uneven split of the documents in FrameNet, leading to data from different source documents and domains in the training and test split. SALSA does not suffer from this problem as much since the split was performed differently. It would be worth considering the same splitting method for FrameNet.

8.3 Future work

As stated previously, FrameId has commonalities with event prediction. Since identifying frames is only one way of capturing events, our approach is transferable to other schemes of event prediction and visual knowledge about participants of situations should be beneficial there, too. It would be interesting to evaluate the multimodal architecture on other predicate-argument frameworks, e.g., script knowledge or VerbNet style Semantic Role Labeling. In particular the exploration our findings on visual contributions to FrameId in the context of further event prediction tasks forms an interesting next step.

More precisely, future work should consider using implicit knowledge not only from images of the participants of the situation, but also from the entire scene in order to directly capture relations between the participants. This could provide access to a more holistic understanding of the scene. The following visual tasks with accompanying datasets could serve as a starting point: (a) visual Verb Sense Disambiguation with the VerSe dataset (Gella et al., 2016) and (b) visual SRL with several datasets, e.g., imSitu (Yatskar et al., 2016) (linked to FrameNet), V-COCO (Gupta and Malik, 2015) (verbs linked to COCO), VVN (Ronchi and Perona, 2015) (visual VerbNet) or even SRL grounded in video clips for the cooking-domain (Yang et al., 2016) and visual Situation Recognition (Mallya and Lazebnik, 2017). Such datasets could be used for extracting visual embeddings for verbs or even complex situations in order to improve the visual component in the embeddings for our FrameId system. Vice versa: visual tasks could profit from multimodal approaches (Baltrušaitis et al., 2017) in a similar sense as our textual task, FrameId, profits from additional information encoded in further modalities. Moreover, visual SRL might profit from our multimodal FrameId system to a similar extend as any FrameNet SRL task profits from correctly identified frames (Hartmann et al., 2017).

Regarding the combination of embeddings from different modalities, we suggest to experiment with different fusion strategies complementing the middle fusion (concatenation) and the mapping (IMAGINED method). This could be a late fusion at decision level operating like an ensemble.

9 Conclusion

In this work, we investigated multimodal representations for Frame Identification (FrameId) by incorporating implicit knowledge, which is better reflected in the visual domain. We presented a flexible FrameId system that is independent of modality and language in its architecture. With this flexibility it is possible to include textual and visual knowledge and to evaluate on gold data in different languages. We created multimodal representations from textual and visual domains and showed that for English FrameNet data, enriching the textual representations with multimodal ones improves the accuracy toward a new state of the art. For German SALSA data, we set a new state of the art with textual representations only and discuss why incorporating multimodal information is more difficult. For both datasets, our system is particularly strong with respect to ambiguous and rare classes, considerably outperforming our new Data-Lexicon Baseline and thus addressing a key challenge in FrameId.

Acknowledgments

This work has been supported by the DFG-funded research training group "Adaptive Preparation of Information form Heterogeneous Sources" (AIPHES, GRK 1994/1). We also acknowledge the useful comments of the anonymous reviewers.

References

Simon Ahrendt and Vera Demberg. 2016. Improving event prediction by representing script participants. In *Proceedings of NAACL-HLT*, pages 546–551, San Diego, USA.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, pages 86–90, Stroudsburg, PA, USA.

Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. 2017. Multimodal machine learning: A survey and taxonomy. *arXiv preprint arXiv:1705.09406*.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49(2014):1–47.

Aljoscha Burchardt, Katrin Erk, Anette Frank, Andrea Kowalski, Sebastian Padó, and Manfred Pinkal. 2006. The SALSA corpus: A German corpus resource for lexical semantics. In *Proceedings of LREC*, Genoa, Italy.

Aljoscha Burchardt, Katrin Erk, Anette Frank, Andrea Kowalski, Sebastian Padó, and Manfred Pinkal. 2009. Using FrameNet for the semantic analysis of German: Annotation, representation, and automation. In *Multilingual FrameNets in Computational Lexicography*, pages 209–244. Mouton de Gruyter, New York City, USA.

Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Return of the devil in the details: Delving deep into convolutional nets. In *Proceedings of the British Machine Vision Conference (BMVC 2015)*, Nottingham, Great Britain.

Kyunghyun Cho, Bart van Merriënboer, Gülçehre Çağlar, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar.

Guillem Collell, Ted Zhang, and Marie-Francine Moens. 2017. Imagined visual representations as multimodal embeddings. In *Thirty-First AAAI Conference on Artificial Intelligence*, pages 4378–4384, San Francisco, USA.

Dipanjan Das, Desai Chen, André F. T. Martins, Nathan Schneider, and Noah A. Smith. 2014. Frame-semantic parsing. *Computational Linguistics*, 40:1:9–56.

Dipanjan Das and Noah A. Smith. 2011. Semi-supervised frame-semantic parsing for unknown predicates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, pages 1435–1444, Stroudsburg, PA, USA.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, Miami, USA.

Timothy Dozat. 2016. Incorporating Nesterov momentum into Adam. In *International Conference on Representation Learning (ICLR): Posters*, pages 1–7, San Juan, Puerto Rico.

Christiane Fellbaum (Ed.). 1998. *WordNet: An Electronic Lexical Database*. The MIT Press, Cambridge, MA ; London.

Michael Ellsworth, Katrin Erk, Paul Kingsbury, and Sebastian Padó. 2004. PropBank, SALSA, and FrameNet: How design determines product. In *Proceedings of the LREC 2004 Workshop on Building Lexical Resources from Semantically Annotated Corpora*, Lisbon, Portugal.

Katrin Erk and Sebastian Pado. 2006. Shalmaneser - A flexible toolbox for semantic role assignment. In *Proceedings of LREC*, Genoa, Italy.

Parvin Sadat Feizabadi and Sebastian Padó. 2012. Automatic identification of motion verbs in WordNet and FrameNet. In *KONVENS*, pages 70–79, Vienna, Austria.

Christiane Fellbaum, editor. 1998. *WordNet: An electronic lexical database*. MIT Press, Cambridge, USA.

Charles J. Fillmore. 1976. Frame semantics and the nature of language. *Annals of the New York Academy of Sciences*, 280:20–32.

Charles J Fillmore. 1985. Frames and the semantics of understanding. *Quaderni di semantica*, 6(2):222–254.

Charles J. Fillmore, Christopher R. Johnson, and Miriam R.L. Petruck. 2003. Background to FrameNet. *International Journal of Lexicography*, 16(3):235–250.

Spandana Gella, Mirella Lapata, and Frank Keller. 2016. Unsupervised visual sense disambiguation for verbs using multimodal embeddings. In *Proceedings of NAACL-HLT*, pages 182–192, San Diego, USA.

Saurabh Gupta and Jitendra Malik. 2015. Visual semantic role labeling. *arXiv preprint arXiv:1505.04474*.

Silvana Hartmann, Ilia Kuznetsov, Teresa Martin, and Iryna Gurevych. 2017. Out-of-domain FrameNet semantic role labeling. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 471–482, Valencia, Spain.

Karl Moritz Hermann, Dipanjan Das, Jason Weston, and Kuzman Ganchev. 2014. Semantic frame identification with distributed word representations. In *Proceedings of the ACL*, pages 1448–1458, Baltimore, USA.

Anders Johannsen, Héctor Martínez Alonso, and Anders Søgaard. 2015. Any-language frame-semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, page 20622066, Lisbon, Portugal.

Douwe Kiela and Léon Bottou. 2014. Learning image embeddings using convolutional neural networks for improved multi-modal semantics. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 36–45, Doha, Qatar.

Arun Mallya and Svetlana Lazebnik. 2017. Recurrent models for situation recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 455–463, Venice, Italy.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning*, pages 807–814, Haifa, Israel.

Roberto Navigli and Simone Paolo Ponzetto. 2010. BabelNet: Building a very large multilingual semantic network. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 216–225, Uppsala, Sweden.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Alexis Palmer and Caroline Sporleder. 2010. Evaluating FrameNet-style semantic parsing: The role of coverage gaps in FrameNet. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 928–936, Beijing, China.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Empirical Methods in Natural Language*, pages 1532–1543, Doha, Qatar.

Ines Rehbein, Josef Ruppenhofer, Caroline Sporleder, and Manfred Pinkal. 2012. Adding nominal spice to SALSA - Frame-semantic annotation of German nouns and verbs. In *Proceedings of KONVENS 2012*, pages 89–97, Vienna, Austria.

Nils Reimers, Judith Eckle-Kohler, Carsten Schnober, Jungi Kim, and Iryna Gurevych. 2014. GermEval-2014: Nested named entity recognition with neural networks. In *Workshop Proceedings of the 12th Edition of the KONVENS Conference*, pages 117–120, Hildesheim, Germany.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 338–348, Copenhagen, Denmark.

Matteo Ruggero Ronchi and Pietro Perona. 2015. Describing common human visual actions in images. In *Proceedings of the British Machine Vision Conference (BMVC 2015)*, Swansea, Wales.

Frank Rosenblatt. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386.

Sascha Rothe and Hinrich Schütze. 2015. AutoExtend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of the ACL*, Beijing, China.

Josef Ruppenhofer, Michael Ellsworth, Miriam R.L. Petruck, Christopher R. Johnson, and Jan Scheffczyk. 2016. *FrameNet II: Extended Theory and Practice*, revised november 1, 2016 edition. International Computer Science Institute, Berkeley, USA.

Roger C Schank and Robert P Abelson. 2013. *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures*. Psychology Press.

Ekaterina Shutova, Douwe Kiela, and Jean Maillard. 2016. Black holes and white rabbits: Metaphor identification with visual features. In *Proceedings of NAACL-HLT*, pages 160–170, San Diego, USA.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Qi Wu, Damien Teney, Peng Wang, Chunhua Shen, Anthony Dick, and Anton van den Hengel. 2017. Visual question answering: A survey of methods and datasets. *Computer Vision and Image Understanding*, 163:21–40.

Ruobing Xie, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2017. Image-embodied knowledge representation learning. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 3140–3146, Melbourne, Australia.

Shaohua Yang, Qiaozi Gao, Changsong Liu, Caiming Xiong, Song-Chun Zhu, and Joyce Y Chai. 2016. Grounded semantic role labeling. In *Proceedings of NAACL-HLT*, pages 149–159, San Diego, USA.

Mark Yatskar, Luke Zettlemoyer, and Ali Farhadi. 2016. Situation recognition: Visual semantic role labeling for image understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5534–5542, Las Vegas, USA.

Learning Joint Semantic Parsers from Disjoint Data

Hao Peng[♢] **Sam Thomson**[♣] **Swabha Swayamdipta**[♣] **Noah A. Smith**[♢]
[♢] Paul G. Allen School of Computer Science & Engineering, University of Washington
[♣] School of Computer Science, Carnegie Mellon University
{hapeng,nasmith}@cs.washington.edu, {sthomson,swabha}@cs.cmu.edu

Abstract

We present a new approach to learning semantic parsers from multiple datasets, even when the target semantic formalisms are drastically different, and the underlying corpora do not overlap. We handle such "disjoint" data by treating annotations for unobserved formalisms as latent structured variables. Building on state-of-the-art baselines, we show improvements both in frame-semantic parsing and semantic dependency parsing by modeling them jointly. Our code is open-source and available at https://github.com/Noahs-ARK/NeurboParser.

1 Introduction

Semantic parsing aims to automatically predict formal representations of meaning underlying natural language, and has been useful in question answering (Shen and Lapata, 2007), text-to-scene generation (Coyne et al., 2012), dialog systems (Chen et al., 2013) and social-network extraction (Agarwal et al., 2014), among others. Various formal meaning representations have been developed corresponding to different semantic theories (Fillmore, 1982; Palmer et al., 2005; Flickinger et al., 2012; Banarescu et al., 2013). The distributed nature of these efforts results in a set of annotated resources that are similar in spirit, but not strictly compatible. A major axis of structural divergence in semantic formalisms is whether based on *spans* (Baker et al., 1998; Palmer et al., 2005) or *dependencies* (Surdeanu et al., 2008; Oepen et al., 2014; Banarescu et al., 2013; Copestake et al., 2005, *inter alia*). Depending on application requirements, either might be most useful in a given situation.

Learning from a union of these resources seems promising, since more data almost always translates into better performance. This is indeed the case for two prior techniques—parameter sharing (FitzGerald et al., 2015; Kshirsagar et al., 2015), and joint decoding across multiple formalisms using cross-task factors that score combinations of substructures from each (Peng et al., 2017). Parameter sharing can be used in a wide range of multitask scenarios, when there is no data overlap or even any similarity between the tasks (Collobert and Weston, 2008; Søgaard and Goldberg, 2016). But techniques involving joint decoding have so far only been shown to work for parallel annotations of dependency-based formalisms, which are structurally very similar to each other (Lluís et al., 2013; Peng et al., 2017). Of particular interest is the approach of Peng et al., where three kinds of semantic graphs are jointly learned on the same input, using parallel annotations. However, as new annotation efforts cannot be expected to use the same original texts as earlier efforts, the utility of this approach is limited.

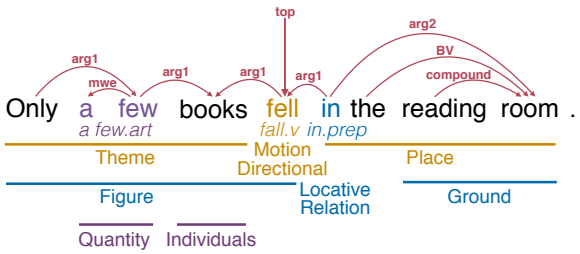


Figure 1: An example sentence from the FrameNet 1.5 corpus, shown with an author-annotated DM semantic dependency graph (above) and frame-semantic annotation (below). Two more gold frames (and their arguments) have been omitted for space.

We propose an extension to Peng et al.'s formulation which addresses this limitation by considering **disjoint resources**, each containing only a single kind of annotation. Moreover, we consider **structurally divergent** formalisms, one dealing with semantic spans and the other with semantic

Proceedings of NAACL-HLT 2018, pages 1492–1502
New Orleans, Louisiana, June 1 - 6, 2018.

dependencies. We experiment on frame-semantic parsing (Gildea and Jurafsky, 2002; Das et al., 2010), a span-based semantic role labeling (SRL) task (§2.1), and on a dependency-based minimum recursion semantic parsing (DELPH-IN MRS, or DM; Flickinger et al., 2012) task (§2.2). See Figure 1 for an example sentence with gold FrameNet annotations, and author-annotated DM representations.

Our joint inference formulation handles missing annotations by treating the structures that are not present in a given training example as latent variables (§3).[1] Specifically, semantic dependencies are treated as a collection of latent variables when training on FrameNet examples.

Using this latent variable formulation, we present an approach for relating spans and dependencies, by explicitly scoring affinities between pairs of potential spans and dependencies. Because there are a huge number of such pairs, we limit our consideration to only certain pairs—our design is inspired by the head rules of Surdeanu et al. (2008). Further possible span-dependency pairs are pruned using an ℓ_1-penalty technique adapted from sparse structure learning (§5). Neural network architectures are used to score frame-semantic structures, semantic dependencies, as well as cross-task structures (§4).

To summarize, our contributions include:

- using a latent variable formulation to extend cross-task scoring techniques to scenarios where datasets do not overlap;
- learning cross-task parts across structurally divergent formalisms; and
- using an ℓ_1-penalty technique to prune the space of cross task parts.

Our approach results in a new state-of-the-art in frame-semantic parsing, improving prior work by 0.8% absolute F_1 points (§6), and achieves competitive performance on semantic dependency parsing. Our code is available at `https://github.com/Noahs-ARK/NeurboParser`.

2 Tasks and Related Work

We describe the two tasks addressed in this work—frame-semantic parsing (§2.1) and semantic dependency parsing (§2.2)—and discuss how their structures relate to each other (§2.3).

2.1 Frame-Semantic Parsing

Frame-semantic parsing is a span-based task, under which certain words or phrases in a sentence evoke semantic **frames**. A **frame** is a group of events, situations, or relationships that all share the same set of participant and attribute types, called **frame elements** or **roles**. Gold supervision for frame-semantic parses comes from the FrameNet lexicon and corpus (Baker et al., 1998).

Concretely, for a given sentence, $\mathbf{x}$, a frame-semantic parse $\mathbf{y}$ consists of:

- a set of **targets**, each being a short span (usually a single token[2]) that evokes a frame;
- for each target t, the **frame** f that it evokes; and
- for each frame f, a set of non-overlapping **argument** spans in the sentence, each argument $a = (i, j, r)$ having a start token index i, end token index j and role label r.

The lemma and part-of-speech tag of a target comprise a **lexical unit** (or **LU**). The FrameNet ontology provides a mapping from an LU ℓ to the set of possible frames it could evoke, $\mathcal{F}_\ell$. Every frame $f \in \mathcal{F}_\ell$ is also associated with a set of roles, $\mathcal{R}_f$ under this ontology. For example, in Figure 1, the LU *"fall.v"* evokes the frame MOTION_DIRECTIONAL. The roles THEME and PLACE (which are specific to MOTION_DIRECTIONAL), are filled by the spans *"Only a few books"* and *"in the reading room"* respectively. LOCATIVE_RELATION has other roles (PROFILED_REGION, ACCESSIBILITY, DEIXIS, etc.) which are not realized in this sentence.

In this work, we assume gold targets and LUs are given, and parse each target independently, following the literature (Johansson and Nugues, 2007; FitzGerald et al., 2015; Yang and Mitchell, 2017; Swayamdipta et al., 2017, *inter alia*). Moreover, following Yang and Mitchell (2017), we perform frame and argument identification jointly. Most prior work has enforced the constraint that a role may be filled by at most one argument span, but following Swayamdipta et al. (2017) we do not impose this constraint, requiring only that arguments for the same target do not overlap.

[1] Following past work on support vector machines with latent variables (Yu and Joachims, 2009), we use the term "latent variable," even though the model is not probabilistic.

[2] 96.5% of targets in the training data are single tokens.

2.2 Semantic Dependency Parsing

Broad-coverage **semantic dependency parsing** (**SDP**; Oepen et al., 2014, 2015, 2016) represents sentential semantics with labeled bilexical dependencies. The SDP task mainly focuses on three semantic formalisms, which have been converted to dependency graphs from their original annotations. In this work we focus on only the DELPHIN MRS (**DM**) formalism.

Each semantic dependency corresponds to a labeled, directed edge between two words. A single token is also designated as the `top` of the parse, usually indicating the main predicate in the sentence. For example in Figure 1, the left-most arc has head *"Only"*, dependent *"few"*, and label `arg1`. In semantic dependencies, the head of an arc is analogous to the target in frame semantics, the destination corresponds to the argument, and the label corresponds to the role. The same set of labels are available for all arcs, in contrast to the frame-specific roles in FrameNet.

2.3 Spans vs. Dependencies

Early semantic role labeling was span-based (Gildea and Jurafsky, 2002; Toutanova et al., 2008, *inter alia*), with spans corresponding to syntactic constituents. But, as in syntactic parsing, there are sometimes theoretical or practical reasons to prefer dependency graphs. To this end, Surdeanu et al. (2008) devised heuristics based on syntactic head rules (Collins, 2003) to transform PropBank (Palmer et al., 2005) annotations into dependencies. Hence, for PropBank at least, there is a very direct connection (through syntax) between spans and dependencies.

For many other semantic representations, such a direct relationship might not be present. Some semantic representations are designed as graphs from the start (Hajič et al., 2012; Banarescu et al., 2013), and have no gold alignment to spans. Conversely, some span-based formalisms are not annotated with syntax (Baker et al., 1998; He et al., 2015),[3] and so head rules would require using (noisy and potentially expensive) predicted syntax.

Inspired by the head rules of Surdeanu et al. (2008), we design cross-task parts, without relying on gold or predicted syntax (which may be either unavailable or error-prone) or on heuristics.

3 Model

Given an input sentence $\mathbf{x}$, and target t with its LU ℓ, denote the set of valid frame-semantic parses (§2.1) as $\mathcal{Y}(\mathbf{x}, t, \ell)$, and valid semantic dependency parses as $\mathcal{Z}(\mathbf{x})$.[4] We learn a parameterized function S that scores candidate parses. Our goal is to *jointly* predict a frame-semantic parse and a semantic dependency graph by selecting the highest scoring candidates:

$$(\hat{\mathbf{y}}, \hat{\mathbf{z}}) = \underset{(\mathbf{y},\mathbf{z}) \in \mathcal{Y}(\mathbf{x},t,\ell) \times \mathcal{Z}(\mathbf{x})}{\arg\max} S(\mathbf{y}, \mathbf{z}, \mathbf{x}, t, \ell). \quad (1)$$

The overall score S can be decomposed into the sum of frame SRL score S_{f}, semantic dependency score S_{d}, and a cross-task score S_{c}:

$$S(\mathbf{y}, \mathbf{z}, \mathbf{x}, t, \ell) = S_{\text{f}}(\mathbf{y}, \mathbf{x}, t, \ell) + S_{\text{d}}(\mathbf{z}, \mathbf{x}) + S_{\text{c}}(\mathbf{y}, \mathbf{z}, \mathbf{x}, t, \ell). \quad (2)$$

S_{f} and S_{c} require access to the target and LU, in addition to $\mathbf{x}$, but S_{d} does not. For clarity, we omit the dependence on the input sentence, target, and lexical unit, whenever the context is clear. Below we describe how each of the scores is computed based on the individual **parts** that make up the candidate parses.

Frame SRL score. The score of a frame-semantic parse consists of

- the score for a predicate part, $s_f(p)$ where each predicate is defined as a combination of a target t, the associated LU, ℓ, and the frame evoked by the LU, $f \in \mathcal{F}_\ell$;
- the score for argument parts, $s_f(a)$, each associated with a token span and semantic role from $\mathcal{R}_f$.

Together, this results in a set of frame-semantic parts of size $O(n^2 \, |\mathcal{F}_\ell| \, |\mathcal{R}_f|)$.[5] The score for a frame semantic structure $\mathbf{y}$ is the sum of local scores of parts in $\mathbf{y}$:

$$S_{\text{f}}(\mathbf{y}) = \sum_{y_i \in \mathbf{y}} s_{\text{f}}(y_i). \quad (3)$$

The computation of s_{f} is described in §4.2.

[3] In FrameNet, phrase types of arguments and their grammatical function in relation to their target have been annotated. But in order to apply head rules, the *internal* structure of arguments (or at least their semantic heads) would also require syntactic annotations.

[4] For simplicity, we consider only a single target here; handling of multiple targets is discussed in §6.

[5] With pruning (described in §6) we reduce this to a number of parts linear in n. Also, $|\mathcal{F}_\ell|$ is usually small (averaging 1.9), as is $|\mathcal{R}_f|$ (averaging 9.5).

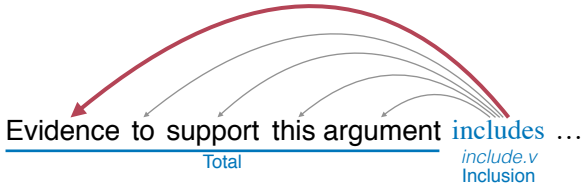


Figure 2: An example of cross-task parts from the FrameNet 1.5 development set. We enumerate all unlabeled semantic dependencies from the first word of the target (*includes*) to any token inside the span. The red bolded arc indicates the prediction of our model.

Semantic dependency score. Following Martins and Almeida (2014), we consider three types of parts in a semantic dependency graph: semantic heads, unlabeled semantic arcs, and labeled semantic arcs. Analogous to Equation 3, the score for a dependency graph $\mathbf{z}$ is the sum of local scores:

$$S_{\mathrm{d}}(\mathbf{z}) = \sum_{z_j \in \mathbf{z}} s_{\mathrm{d}}(z_j), \tag{4}$$

The computation of s_{d} is described in §4.3.

Cross task score. In addition to task-specific parts, we introduce a set $\mathcal{C}$ of cross-task parts. Each cross-task part relates an argument part from $\mathbf{y}$ to an *unlabeled* dependency arc from $\mathbf{z}$. Based on the head-rules described in §2.3, we consider unlabeled arcs from the target to any token inside the span.[6] Intuitively, an argument in FrameNet *would* be converted into a dependency from its target to the semantic head of its span. Since we do not know the semantic head of the span, we consider all tokens in the span as potential modifiers of the target. Figure 2 shows examples of cross-task parts. The cross-task score is given by

$$S_{\mathrm{c}}(\mathbf{y}, \mathbf{z}) = \sum_{(y_i, z_j) \in (\mathbf{y} \times \mathbf{z}) \cap \mathcal{C}} s_{\mathrm{c}}(y_i, z_j). \tag{5}$$

The computation of s_{c} is described in §4.4.

In contrast to previous work (Lluís et al., 2013; Peng et al., 2017), where there are parallel annotations for all formalisms, our input sentences contain only one of the two—either the span-based frame SRL annotations, or semantic dependency graphs from DM. To handle missing annotations, we treat semantic dependencies $\mathbf{z}$ as latent when decoding frame-semantic structures.[7] Because the DM dataset we use does not have target annotations, we do not use latent variables for frame semantic structures when predicting semantic dependency graphs. The parsing problem here reduces to

$$\hat{\mathbf{z}} = \arg\max_{\mathbf{z} \in \mathcal{Z}} S_{\mathrm{d}}(\mathbf{z}), \tag{6}$$

in contrast with Equation 1 .

4 Parameterizations of Scores

This section describes the parametrization of the scoring functions from §3. At a very high level: we learn contextualized token and span vectors using a bidirectional LSTM (biLSTM; Graves, 2012) and multilayer perceptrons (MLPs) (§4.1); we learn lookup embeddings for LUs, frames, roles, and arc labels; and to score a part, we combine the relevant representations into a single scalar score using a (learned) low-rank multilinear mapping. Scoring frames and arguments is detailed in §4.2, that of dependency structures in §4.3, and §4.4 shows how to capture interactions between arguments and dependencies. All parameters are learned jointly, through the optimization of a multitask objective (§5).

Tensor notation. The **order** of a tensor is the number of its dimensions—an order-2 tensor is a matrix and an order-1 tensor is a vector. Let $\otimes$ denote tensor product; the tensor product of two order-2 tensors $\mathcal{A}$ and $\mathcal{B}$ yields an order-4 tensor where $(\mathcal{A} \otimes \mathcal{B})_{i,j,k,l} = \mathcal{A}_{i,j}\mathcal{B}_{k,l}$. We use $\langle \cdot, \cdot \rangle$ to denote inner products.

4.1 Token and Span Representations

The representations of tokens and spans are formed using biLSTMs followed by MLPs.

Contextualized token representations. Each token in the input sentence $\mathbf{x}$ is mapped to an embedding vector. Two LSTMs (Hochreiter and Schmidhuber, 1997) are run in opposite directions over the input vector sequence. We use the concatenation of the two hidden representations at each position i as a contextualized word embedding for each token:

$$\mathbf{h}_i = \left[\overrightarrow{\mathbf{h}}_i; \overleftarrow{\mathbf{h}}_i\right]. \tag{7}$$

[6] Most targets are single-words (§2.1). For multi-token targets, we consider only the first token, which is usually content-bearing.

[7] Semantic dependency parses over a sentence are not constrained to be identical for different frame-semantic targets.

Span representations. Following Lee et al. (2017), span representations are computed based on boundary word representations and discrete length and distance features. Concretely, given a target t and its associated argument $a = (i, j, r)$ with boundary indices i and j, we compute three features $\boldsymbol{\phi}_t(a)$ based on the length of a, and the distances from i and j to the start of t. We concatenate the token representations at a's boundary with the discrete features $\boldsymbol{\phi}_t(a)$. We then use a two-layer $\tanh$-MLP to compute the span representation:

$$\mathbf{g}^{\text{span}}(i, j) = \text{MLP}^{\text{span}}\left([\mathbf{h}_i; \mathbf{h}_j; \boldsymbol{\phi}_t(a)]\right). \quad (8)$$

The target representation $\mathbf{g}^{\text{tgt}}(t)$ is similarly computed using a separate MLP^{tgt}, with a length feature but no distance features.

4.2 Frame and Argument Scoring

As defined in §3, the representation for a predicate part incorporates representations of a target span, the associated LU and the frame evoked by the LU. The score for a predicate part is given by a multilinear mapping:

$$\mathbf{g}^{\text{pred}}(f) = \mathbf{g}^{\text{fr}}(f) \otimes \mathbf{g}^{\text{tgt}}(t) \otimes \mathbf{g}^{\text{lu}}(\ell) \quad (9a)$$

$$s_{\text{f}}(p) = \left\langle \boldsymbol{\mathcal{W}}, \mathbf{g}^{\text{pred}}(f) \right\rangle, \quad (9b)$$

where $\boldsymbol{\mathcal{W}}$ is a low-rank order-3 tensor of learned parameters, and $\mathbf{g}^{\text{fr}}(f)$ and $\mathbf{g}^{\text{lu}}(\ell)$ are learned lookup embeddings for the frame and LU.

A candidate argument consists of a span and its role label, which in turn depends on the frame, target and LU. Hence the score for argument part, $a = (i, j, r)$ is given by extending definitions from Equation 9:

$$\mathbf{g}^{\text{arg}}(a) = \mathbf{g}^{\text{span}}(i, j) \otimes \mathbf{g}^{\text{role}}(r), \quad (10a)$$

$$s_{\text{f}}(a) = \left\langle \boldsymbol{\mathcal{W}} \otimes \boldsymbol{\mathcal{U}}, \mathbf{g}^{\text{pred}}(f) \otimes \mathbf{g}^{\text{arg}}(a) \right\rangle, \quad (10b)$$

where $\boldsymbol{\mathcal{U}}$ is a low-rank order-2 tensor of learned parameters and $\mathbf{g}^{\text{role}}(r)$ is a learned lookup embedding of the role label.

4.3 Dependency Scoring

Local scores for dependencies are implemented with two-layer $\tanh$-MLPs, followed by a final linear layer reducing the represenation to a single scalar score. For example, let $u = i \rightarrow j$ denote an unlabeled arc (ua). Its score is:

$$\mathbf{g}^{\text{ua}}(u) = \text{MLP}^{\text{ua}}\left([\mathbf{h}_i; \mathbf{h}_j]\right) \quad (11a)$$

$$s_{\text{d}}(u) = \mathbf{w}^{\text{ua}} \cdot \mathbf{g}^{\text{ua}}(u), \quad (11b)$$

where $\mathbf{w}^{\text{ua}}$ is a vector of learned weights. The scores for other types of parts are computed similarly, but with separate MLPs and weights.

4.4 Cross-Task Part Scoring

As shown in Figure 2, each cross-task part c consists of two first-order parts: a frame argument part a, and an unlabeled dependency part, u. The score for a cross-task part incorporates both:

$$s_{\text{c}}(c) = \left\langle \boldsymbol{\mathcal{W}} \otimes \boldsymbol{\mathcal{U}} \otimes \boldsymbol{\mathcal{V}}, \mathbf{g}^{\text{pred}}(f) \otimes \mathbf{g}^{\text{arg}}(a) \otimes \mathbf{w}^{\text{ua}} \otimes \mathbf{g}^{\text{ua}}(u) \right\rangle, \quad (12)$$

where $\boldsymbol{\mathcal{V}}$ is a low-rank order-2 tensor of parameters. Following previous work (Lei et al., 2014; Peng et al., 2017), we construct the parameter tensors $\boldsymbol{\mathcal{W}}$, $\boldsymbol{\mathcal{U}}$, and $\boldsymbol{\mathcal{V}}$ so as to upper-bound their ranks.

5 Training and Inference

All parameters from the previous sections are trained using a max-margin training objective (§5.1). For inference, we use a linear programming procedure, and a sparsity-promoting penalty term for speeding it up (§5.2).

5.1 Max-Margin Training

Let $\mathbf{y}^*$ denote the gold frame-semantic parse, and let $\delta(\mathbf{y}, \mathbf{y}^*)$ denote the cost of predicting $\mathbf{y}$ with respect to $\mathbf{y}^*$. We optimize the latent structured hinge loss (Yu and Joachims, 2009), which gives a subdifferentiable upper-bound on δ:

$$\mathcal{L}(\mathbf{y}^*) = \max_{(\mathbf{y}, \mathbf{z}) \in \mathcal{Y} \times \mathcal{Z}} \{S(\mathbf{y}, \mathbf{z}) + \delta(\mathbf{y}, \mathbf{y}^*)\} - \max_{\mathbf{z} \in \mathcal{Z}} \{S(\mathbf{y}^*, \mathbf{z})\}. \quad (13)$$

Following Martins and Almeida (2014), we use a weighted Hamming distance as the cost function, where, to encourage recall, we use costs 0.6 for false negative predictions and 0.4 for false positives. Equation 13 can be evaluated by applying the same max-decoding algorithm twice—once with cost-augmented inference (Crammer et al., 2006), and once more keeping $\mathbf{y}^*$ fixed. Training then aims to minimize the average loss over all training instances.[8]

Another potential approach to training a model on disjoint data would be to marginalize out the

[8] We do not use latent frame structures when decoding semantic dependency graphs (§3). Hence, the loss reduces to structured hinge (Tsochantaridis et al., 2004) when training on semantic dependencies.

latent structures and optimize the conditional log-likelihood (Naradowsky et al., 2012). Although max-decoding and computing marginals are both NP-hard in general graphical models, there are more efficient off-the-shelf implementations for approximate max-decoding, hence, we adopt a max-margin formulation.

5.2 Inference

We formulate the maximizations in Equation 13 as 0–1 integer linear programs and use AD^3 to solve them (Martins et al., 2011). We only enforce a non-overlapping constraint when decoding FrameNet structures, so that the argument identification subproblem can be efficiently solved by a dynamic program (Kong et al., 2016; Swayamdipta et al., 2017). When decoding semantic dependency graphs, we enforce the determinism constraint (Flanigan et al., 2014), where certain labels may appear on at most one arc outgoing from the same token.

Inference speedup by promoting sparsity. As discussed in §3, even after pruning, the number of within-task parts is linear in the length of the input sentence, so the number of cross-task parts is quadratic. This leads to potentially very slow inference. We address this problem by imposing an ℓ_1 penalty on the cross-task part scores:

$$\mathcal{L}(\mathbf{y}^*) + \lambda \sum_{(y_i, z_j) \in \mathcal{C}} \left| s_c(y_i, z_j) \right|, \qquad (14)$$

where λ is a hyperparameter, set to 0.01 as a practical tradeoff between efficiency and development set performance. Whenever the score for a cross-task part is driven to zero, that part's score no longer needs to be considered during inference. It is important to note that by promoting sparsity this way, we do not prune out any candidate solutions. We are instead encouraging fewer terms in the scoring function, which leads to smaller, faster inference problems even though the space of feasible parses is unchanged.

The above technique is closely related to a line of work in estimating the structure of sparse graphical models (Yuan and Lin, 2007; Friedman et al., 2008), where an ℓ_1 penalty is applied to the inverse covariance matrix in order to induce a smaller number of conditional dependencies between variables. To the best of our knowledge, we are the first to apply this technique to the output of neural scoring functions. Here, we are interested in learning sparse graphical models only because they result in faster inference, not because we have any *a priori* belief about sparsity. This results in roughly a $14\times$ speedup in our experiments, without any significant drop in performance.

	Train	Exemplars	Dev.	Test
FN 1.5	17,143	153,952	2,333	4,457
FN 1.7	19,875	192,460	2,308	6,722
DM id	33,961	-	1,692	1,410
DM ood	-	-	-	1,849

Table 1: Number of instances in datasets.

6 Experiments

Datasets. Our model is evaluated on two different releases of FrameNet: FN 1.5 and FN 1.7,[9] using splits from Swayamdipta et al. (2017). Following Swayamdipta et al. (2017) and Yang and Mitchell (2017), each target annotation is treated as a separate training instance. We also include as training data the exemplar sentences, each annotated for a single target, as they have been reported to improve performance (Kshirsagar et al., 2015; Yang and Mitchell, 2017). For semantic dependencies, we use the English DM dataset from the SemEval 2015 Task 18 closed track (Oepen et al., 2015).[10] DM contains instances from the WSJ corpus for training and both in-domain (id) and out-of-domain (ood) test sets, the latter from the Brown corpus.[11] Table 1 summarizes the sizes of the datasets.

Baselines. We compare FN performance of our joint learning model (FULL) to two baselines:

BASIC: A single-task frame SRL model, trained using a structured hinge objective.

NOCTP: A joint model without cross-task parts. It demonstrates the effect of sharing parameters in word embeddings and LSTMs (like in FULL). It does not use latent semantic dependency structures, and aims to minimize the sum of training losses from both tasks.

We also compare semantic dependency parsing performance against the single task model by Peng

[9] https://FN.icsi.berkeley.edu/fndrupal/

[10] http://sdp.delph-in.net/. The closed track does not have access to any syntactic analyses. The impact of syntactic features on SDP performance is extensively studied in Ribeyre et al. (2015).

[11] Our FN training data does not overlap with the DM test set. We remove the 3 training sentences from DM which appear in FN test data.

Model	Prec.	Rec.	F_1
Roth	72.2	68.0	70.0
Täckström	75.4	65.8	70.3
FitzGerald	74.8	65.5	69.9
FitzGerald (10×)	75.0	67.3	70.9
open-SESAME	71.0	67.8	69.4
open-SESAME (5×)	71.2	70.5	70.9
Yang and Mitchell (REL)	77.1	68.7	72.7
†*Yang and Mitchell (ALL)	78.8	74.5	76.6
†This work (FULL)	80.4	73.5	76.8
†This work (FULL, 2×)	**80.4**	**74.7**	**77.4**
†This work (BASIC)	79.2	71.7	75.3
†This work (NOCTP)	76.9	74.8	75.8

Table 2: FN 1.5 full structure extraction test performance. † denotes the models jointly predicting frames and arguments, and other systems implement two-stage pipelines and use the algorithm by Hermann et al. (2014) to predict frames. $K\times$ denotes a product-of-experts ensemble of K models. *Ensembles a sequential tagging CRF and a relational model. Bold font indicates best performance among all systems.

et al. (2017), denoted as NeurboParser (BASIC). To ensure fair comparison with our FULL model, we made several modifications to their implementation (§6.3). We observed performance improvements from our reimplementation, which can be seen in Table 5.

Pruning strategies. For frame SRL, we discard argument spans longer than 20 tokens (Swayamdipta et al., 2017). We further pretrain an unlabeled model and prune spans with posteriors lower than $1/n^2$, with n being the input sentence length. For semantic dependencies, we generally follow Martins and Almeida (2014), replacing their feature-rich pruner with neural networks. We observe that $O(n)$ spans/arcs remain after pruning, with around 96% FN development recall, and more than 99% for DM.[12]

6.1 Empirical Results

FN parsing results. Table 2 compares our full frame-semantic parsing results to previous systems. Among them, Täckström et al. (2015) and Roth (2016) implement a two-stage pipeline and use the method from Hermann et al. (2014) to predict frames. FitzGerald et al. (2015) uses the same pipeline formulation, but improves the frame identification of Hermann et al. (2014) with better syntactic features. open-SESAME (Swayamdipta et al., 2017) uses predicted frames from FitzGerald et al. (2015), and improves argument identification using a softmax-margin segmental RNN. They observe further improvements from product of experts ensembles (Hinton, 2002).

The best published FN 1.5 results are due to Yang and Mitchell (2017). Their relational model (REL) formulates argument identification as a sequence of local classifications. They additionally introduce an ensemble method (denoted as ALL) to integrate the predictions of a sequential CRF. They use a linear program to jointly predict frames and arguments at test time. As shown in Table 2, our single-model performance outperforms their REL model, and is on par with their ALL model. For a fair comparison, we build an ensemble (FULL, 2×) by separately training two models, differing only in random seeds, and averaging their part scores. Our ensembled model outperforms previous best results by 0.8% absolute.

Model	All	Ambiguous
Hartmann	87.6	-
Yang and Mitchell	88.2	-
Hermann	88.4	73.1
†This work (BASIC)	89.2	76.3
†This work (NOCTP)	89.2	76.4
†This work (FULL)	89.9	77.7
†This work (FULL, 2×)	**90.0**	**78.0**

Table 3: Frame identification accuracy on the FN 1.5 test set. *Ambiguous* evaluates only on lexical units having more than one possible frames. † denotes joint frame and argument identification, and bold font indicates best performance.[13]

Table 3 compares our frame identification results with previous approaches. Hermann et al. (2014) and Hartmann et al. (2017) use distributed word representations and syntax features. We follow the FULL LEXICON setting (Hermann et al., 2014) and extract candidate frames from the offi-

[12]On average, around $0.8n$ argument spans, and $5.7n$ unlabeled dependency arcs remain after pruning.

[13]Our comparison to Hermann et al. (2014) is based on their updated version: http://www.aclweb.org/anthology/P/P14/P14-1136v2.pdf. Ambiguous frame identification results by Yang and Mitchell (2017) and Hartmann et al. (2017) are 75.7 and 73.8. Their ambiguous lexical unit sets are different from the one extracted from the official frame directory, and thus the results are not comparable to those in Table 3.

	Full Structure			Frame Id.	
Model	**Prec.**	**Rec.**	F_1	**All**	**Amb.**
BASIC	78.0	72.1	75.0	88.6	76.6
NOCTP	79.8	72.4	75.9	88.5	76.3
FULL	**80.2**	**72.9**	**76.4**	**89.1**	**77.5**

Table 4: FN 1.7 full structure extraction and frame identification test results. Bold font indicates best performance. FN 1.7 test set is an extension of FN 1.5 test, hence the results here are not comparable to those reported in Table 2.

Model	**id F_1**	**ood F_1**
NeurboParser (BASIC)	89.4	84.5
NeurboParser (FREDA3)	90.4	85.3
NeurboParser (BASIC, reimpl.)	90.0	84.6
This work (NOCTP)	89.9	85.2
This work (FULL)	90.5	85.9
This work (FULL, 2×)	**91.2**	**86.6**

Table 5: Labeled parsing performance in F_1 score for DM semantic dependencies. *id* denotes in-domain WSJ test data, and *ood* denotes out-of-domain brown corpus test data. Bold font indicates best performance.

cial directories. The *Ambiguous* setting compares lexical units with more than one possible frames. Our approach improves over all previous models under both settings, demonstrating a clear benefit from joint learning.

We observe similar trends on FN 1.7 for both full structure extraction and for frame identification only (Table 4). FN 1.7 extends FN 1.5 with more consistent annotations. Its test set is different from that of FN 1.5, so the results are not directly comparable to Table 2. We are the first to report frame-semantic parsing results on FN 1.7, and we encourage future efforts to do so as well.

Semantic dependency parsing results. Table 5 compares our semantic dependency parsing performance on DM with the baselines. Our reimplementation of the BASIC model slightly improves performance on in-domain test data. The NOCTP model ties parameters from word embeddings and LSTMs when training on FrameNet and DM, but does not use cross-task parts or joint prediction. NOCTP achieves similar in-domain test performance, and improves over BASIC on out-of-domain data. By jointly predicting FrameNet structures and semantic dependency graphs, the FULL model outperforms the baselines by more than 0.6% absolute F_1 scores under both settings.

Previous state-of-the-art results on DM are due to the joint learning model of Peng et al. (2017), denoted as NeurboParser (FREDA3). They adopted a multitask learning approach, jointly predicting three different parallel semantic dependency annotations. Our FULL model's in-domain test performance is on par with FREDA3, and improves over it by 0.6% absolute F_1 on out-of-domain test data. Our ensemble of two FULL models achieves a new state-of-the-art in both in-domain and out-of-domain test performance.

		Rel. Err. (%)	
Operation	**Description**	BASIC	FULL
Frame error	Frame misprediction.	11.3	11.1
Role error	Matching span with incorrect role.	12.6 (5.2)	13.4 (5.9)
Span error	Matching role with incorrect span.	11.4	12.3
Arg. error	Predicted argument does not overlap with any gold span.	18.6	22.4
Missing arg.	Gold argument does not overlap with any predicted span.	43.5	38.0

Table 6: Percentage of errors made by BASIC and FULL models on the FN 1.5 development set. Parenthesized numbers show the percentage of role errors when frame predictions are correct.

6.2 Analysis

Error type breakdown. Similarly to He et al. (2017), we categorize prediction errors made by the BASIC and FULL models in Table 6. Entirely missing an argument accounts for most of the errors for both models, but we observe fewer errors by FULL compared to BASIC in this category. FULL tends to predict more arguments in general, including more incorrect arguments.

Since candidate roles are determined by frames, frame and role errors are highly correlated. Therefore, we also show the role errors when frames are correctly predicted (parenthesized numbers in the second row). When a predicted argument span matches a gold span, predicting the semantic role is less challenging. Role errors account for only around 13% of all errors, and half of them are due to mispredictions of frames.

Performance by argument length. Figure 3 plots dev. precision and recall of both BASIC and FULL against binned argument lengths. We ob-

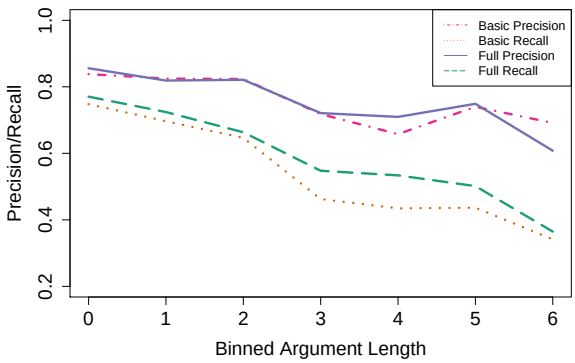


Figure 3: FN 1.5 development precision and recall of BASIC and FULL by different argument lengths. Length ℓ is binned to $\lfloor \log_{1.6} \ell \rfloor$, and precision/recall values are smoothed with `loess`, with a smoothing parameter of 0.1.

serve two trends: (a) FULL tends to predict longer arguments (averaging 3.2) compared to BASIC (averaging 2.9), while keeping similar precision;[14] (b) recall improvement in FULL mainly comes from arguments longer than 4.

6.3 Implementation Details

Our implementation is based on DyNet (Neubig et al., 2017).[15] We use predicted part-of-speech tags and lemmas using NLTK (Bird et al., 2009).[16]

Parameters are optimized with stochastic subgradient descent for up to 30 epochs, with ℓ_2 norms of gradients clipped to 1. We use 0.33 as initial learning rate, and anneal it at a rate of 0.5 every 10 epochs. Early stopping is applied based on FN development F_1. We apply logarithm with base 2 to all discrete features, e.g., $\log_2(d+1)$ for distance feature valuing d. To speed up training, we randomly sample a 35% subset from the FN exemplar instances each epoch.

Hyperparameters. Each input token is represented as the concatenation a word embedding vector, a learned lemma vector, and a learned vector for part-of speech, all updated during training. We use 100-dimensional `GloVe` (Pennington et al., 2014) to initialize word embeddings. We apply word dropout (Iyyer et al., 2015) and randomly replace a word w with a special `UNK` symbol with probability $\frac{\alpha}{1+\#(w)}$, with $\#(w)$ being the count of w in the training set. We follow the default parameters initialization procedure by DyNet, and an ℓ_2 penalty of 10^{-6} is applied to all weights. See Table 7 for other hyperparameters.

Hyperparameter	Value
Word embedding dimension	100 (32)
Lemma embedding dimension	50 (16)
POS tag embedding dimension	50 (16)
MLP dimension	100 (32)
Tensor rank r	100 (32)
BiLSTM layers	2 (1)
BiLSTM dimensions	200 (64)
α for word dropout	1.0 (1.0)

Table 7: Hyperparameters used in the experiments. Parenthesized numbers indicate those used by the pretrained pruners.

Modifications to Peng et al. (2017). To ensure fair comparisons, we note two implementation modifications to Peng et al.'s basic model. We use a more recent version (2.0) of the DyNet toolkit, and we use 50-dimensional lemma embeddings instead of their 25-dimensional randomly-initialized learned word embeddings.

7 Conclusion

We presented a novel multitask approach to learning semantic parsers from disjoint corpora with structurally divergent formalisms. We showed how joint learning and prediction can be done with scoring functions that explicitly relate spans and dependencies, even when they are never observed together in the data. We handled the resulting inference challenges with a novel adaptation of graphical model structure learning to the deep learning setting. We raised the state-of-the-art on DM and FrameNet parsing by learning from both, despite their structural differences and non-overlapping data. While our selection of factors is specific to spans and dependencies, our general techniques could be adapted to work with more combinations of structured prediction tasks. We have released our implementation at `https://github.com/Noahs-ARK/NeurboParser`.

Acknowledgments

We thank Kenton Lee, Luheng He, and Rowan Zellers for their helpful comments, and the anonymous reviewers for their valuable feedback. This work was supported in part by NSF grant IIS-1562364.

[14] Average gold span length is 3.4 after discarding those longer than 20.

[15] `https://github.com/clab/dynet`

[16] `http://www.nltk.org/`

References

Apoorv Agarwal, Sriramkumar Balasubramanian, Anup Kotalwar, Jiehan Zheng, and Owen Rambow. 2014. Frame semantic tree kernels for social network extraction from text. In *Proc. of EACL*.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proc. ACL*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proc. LAW-ID*.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. " O'Reilly Media, Inc.".

Yun-Nung Chen, William Yang Wang, and Alexander I Rudnicky. 2013. Unsupervised induction and filling of semantic slots for spoken dialogue systems using frame-semantic parsing. In *Proc. of ASRU-IEEE*.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics* 29(4):589–637.

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

Ann Copestake, Dan Flickinger, Ivan A. Sag, and Carl Pollard. 2005. Minimal recursion semantics: An introduction. *Research on Language & Computation* 3(4):281–332.

Bob Coyne, Alex Klapheke, Masoud Rouhizadeh, Richard Sproat, and Daniel Bauer. 2012. Annotation tools and knowledge representation for a text-to-scene system. In *Proc. of COLING*.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *JMLR* 7:551–585.

Dipanjan Das, Nathan Schneider, Desai Chen, and Noah A. Smith. 2010. Probabilistic frame-semantic parsing. In *Proc. of NAACL*.

Charles Fillmore. 1982. Frame semantics. *Linguistics in the morning calm* pages 111–137.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proc. of EMNLP*.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proc. ACL*.

Daniel Flickinger, Yi Zhang, and Valia Kordoni. 2012. DeepBank: A dynamically annotated treebank of the Wall Street Journal. In *Proc. of TLT*. pages 85–96.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2008. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics* 9(3):432–441.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics* 28(3):245–288.

Alex Graves. 2012. *Supervised Sequence Labelling with Recurrent Neural Networks*, volume 385 of *Studies in Computational Intelligence*. Springer.

Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Ondřej Bojar, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semecký, Jana Šindlerová, Jan Štěpánek, Josef Toman, Zdeňka Urešová, and Zdeněk Žabokrtský. 2012. Announcing Prague Czech-English dependency treebank 2.0. In *Proc. of LREC*.

Silvana Hartmann, Ilia Kuznetsov, Teresa Martin, and Iryna Gurevych. 2017. Out-of-domain FrameNet semantic role labeling. In *Proc. of EACL*.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and whats next. In *Proc. of ACL*.

Luheng He, Mike Lewis, and Luke S. Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *Proc. of EMNLP*.

Karl Moritz Hermann, Dipanjan Das, Jason Weston, and Kuzman Ganchev. 2014. Semantic frame identification with distributed word representations. In *Proc. of ACL*.

Geoffrey E. Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation* 14(8):1771–1800.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9(8):1735–1780.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proc. of ACL*.

Richard Johansson and Pierre Nugues. 2007. LTH: Semantic structure extraction using nonprojective dependency trees. In *Proc. of SemEval*.

Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2016. Segmental recurrent neural networks. In *Proc. of ICLR*.

Meghana Kshirsagar, Sam Thomson, Nathan Schneider, Jaime Carbonell, Noah A. Smith, and Chris Dyer. 2015. Frame-semantic role labeling with heterogeneous annotations. In *Proc. ACL*.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proc. of EMNLP*.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *Proc. ACL*.

Xavier Lluís, Xavier Carreras, and Lluís Màrquez. 2013. Joint arc-factored parsing of syntactic and semantic dependencies. *TACL* 1:219–230.

André F. T. Martins and Mariana S. C. Almeida. 2014. Priberam: A turbo semantic parser with second order features. In *Proc. of SemEval*.

André F. T. Martins, Noah A. Smith, Pedro M. Q. Aguiar, and Mário A. T. Figueiredo. 2011. Dual decomposition with many overlapping components. In *Proc. of EMNLP*.

Jason Naradowsky, Sebastian Riedel, and David A. Smith. 2012. Improving NLP through marginalization of hidden syntactic structure. In *Proc. EMNLP*.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. DyNet: The dynamic neural network toolkit. arXiv:1701.03980.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinkova, Dan Flickinger, Jan Hajic, and Zdenka Uresova. 2015. SemEval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proc. of SemEval*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, Angelina Ivanova, and Zdeňka Urešová. 2016. Towards comparability of linguistic graph banks for semantic parsing. In *Proc. of LREC*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Yi Zhang. 2014. SemEval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proc. of SemEval*.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computional Linguistics* 31(1):71–106.

Hao Peng, Sam Thomson, and Noah A. Smith. 2017. Deep multitask learning for semantic dependency parsing. In *Proc. of ACL*.

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

Corentin Ribeyre, Éric Villemonte De La Clergerie, and Djamé Seddah. 2015. Because Syntax does Matter: Improving Predicate-Argument Structures Parsing Using Syntactic Features. In *Proc. of NAACL*.

Michael Roth. 2016. Improving frame semantic parsing via dependency path embeddings. In *Book of Abstracts of the 9th International Conference on Construction Grammar*.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *Proc. of EMNLP-CoNLL*.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proc. of ACL*.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proc. of CoNLL*.

Swabha Swayamdipta, Sam Thomson, Chris Dyer, and Noah A. Smith. 2017. Frame-semantic parsing with softmax-margin segmental RNNs and a syntactic scaffold. arXiv:1706.09528.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient inference and structured learning for semantic role labeling. *TACL* 3:29–41.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2008. A global joint model for semantic role labeling. *Computational Linguistics* 34(2):161–191.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proc. of ICML*.

Bishan Yang and Tom Mitchell. 2017. A joint sequential and relational model for frame-semantic parsing. In *Proc. of EMNLP*.

Chun-Nam John Yu and Thorsten Joachims. 2009. Learning structural SVMs with latent variables. In *Proc. of ICML*.

Ming Yuan and Yi Lin. 2007. Model selection and estimation in the gaussian graphical model. *Biometrika* 94(1):19–35.

Identifying Semantic Divergences in Parallel Text without Annotations

Yogarshi Vyas and **Xing Niu** and **Marine Carpuat**
Department of Computer Science
University of Maryland
College Park, MD 20742, USA
yogarshi@cs.umd.edu, xingniu@cs.umd.edu, marine@cs.umd.edu

Abstract

Recognizing that even correct translations are not always semantically equivalent, we automatically detect meaning divergences in parallel sentence pairs with a deep neural model of bilingual semantic similarity which can be trained for any parallel corpus without any manual annotation. We show that our semantic model detects divergences more accurately than models based on surface features derived from word alignments, and that these divergences matter for neural machine translation.

1 Introduction

Parallel sentence pairs are sentences that are translations of each other, and are therefore often assumed to convey the same meaning in the source and target language. Occasional mismatches between source and target have been primarily viewed as alignment noise (Goutte et al., 2012) due to imperfect sentence alignment tools in parallel corpora drawn from translated texts (Tiedemann, 2011; Xu and Yvon, 2016), or the noisy process of extracting parallel segments from non-parallel corpora (Fung and Cheung, 2004; Munteanu and Marcu, 2005).

In contrast, we view translation as a process that inherently introduces meaning mismatches, so that even correctly aligned sentence pairs are not necessarily semantically equivalent. This can happen for many reasons: translation lexical choice often involves selecting between near synonyms that introduce language-specific nuances (Hirst, 1995), typological divergences lead to structural mismatches (Dorr, 1994), differences in discourse organization can make it impossible to find one-to-one sentence alignments (Li et al., 2014). Cross-linguistic variations in other discourse phenomena such as coreference, discourse relation and modality (Lapshinova-Koltunski, 2015) compounded with translation effects that distinguish "translationese" from original text (Koppel and Ordan, 2011) might also lead to meaning mismatches between source and target.

In this paper, we aim to provide empirical evidence that semantic divergences exist in parallel corpora and matter for downstream applications. This requires an automatic method to distinguish semantically equivalent sentence pairs from semantically divergent pairs, so that parallel corpora can be used more judiciously in downstream cross-lingual NLP applications. We propose a semantic model to automatically detect whether a sentence pair is semantically divergent (Section 3). While prior work relied on surface cues to detect mis-aligments, our approach focuses on comparing the meaning of words and overlapping text spans using bilingual word embeddings (Luong et al., 2015) and a deep convolutional neural network (He and Lin, 2016). Crucially, training this model requires no manual annotation. Noisy supervision is obtained automatically borrowing techniques developed for parallel sentence extraction (Munteanu and Marcu, 2005). Our model can thus easily be trained to detect semantic divergences in any parallel corpus.

We extensively evaluate our semantically-motivated models on intrinsic and extrinsic tasks: detection of divergent examples in two parallel English-French data sets (Section 5), and data selection for English-French and Vietnamese-English machine translation (MT) (Section 6).The semantic models significantly outperform other methods on the intrinsic task, and help select data to train neural machine translation faster with no loss in translation quality. Taken together, these results provide empirical evidence that sentence-alignment does not necessarily imply semantic equivalence, and that this distinction matters in practice for a downstream NLP application.

Proceedings of NAACL-HLT 2018, pages 1503–1515
New Orleans, Louisiana, June 1 - 6, 2018.

2 Background

Translation Divergences We use the term semantic divergences to refer to bilingual sentence pairs, including translations, that do not have the same meaning. These differ from *typological divergences*, which have been defined as structural differences between sentences that convey the same meaning (Dorr, 1994; Habash and Dorr, 2002), and reflect the fact that languages do not encode the same information in the same way.

Non-Parallel Corpora Mismatches in bilingual sentence pairs have previously been studied to extract parallel segments from non-parallel corpora, and augment MT training data (Fung and Cheung, 2004; Munteanu and Marcu, 2005, 2006; Abdul-Rauf and Schwenk, 2009; Smith et al., 2010; Riesa and Marcu, 2012, *inter alia)*. Methods for parallel sentence extraction rely primarily on surface features based on translation lexicons and word alignment patterns (Munteanu and Marcu, 2005, 2006). Similar features have proved to be useful for the related task of translation quality estimation (Specia et al., 2010, 2016), which aims to detect divergences introduced by MT errors, rather than human translation. Recently, sentence embeddings have also been used to detect parallelism (España-Bonet et al., 2017; Schwenk and Douze, 2017). Although embeddings capture semantic generalizations, these models are trained with neural MT objectives, which do not distinguish semantically equivalent segments from divergent parallel segments.

Cross-Lingual Sentence Semantics Cross-lingual semantic textual similarity (Agirre et al., 2016) and cross-lingual textual entailment (Negri and Mehdad, 2010; Negri et al., 2012, 2013) seek to characterize semantic relations between sentences in two different languages beyond translation equivalence, and are therefore directly relevant to our goal. While the human judgments obtained for each task differ, they all take inputs of the same form (two segments in different languages) and output a prediction that can be interpreted as indicating whether they are equivalent in meaning or not. Models share core intuitions, relying either on MT to transfer the cross-lingual task into its monolingual equivalent (Jimenez et al., 2012; Zhao et al., 2013), or on features derived from MT components such as translation dictionaries and word alignments (Turchi and Negri, 2013; Lo et al., 2016). Training requires manually annotated examples, either bilingual, or monolingual when using MT for language transfer.

Impact of mismatched sentence pairs on MT Prior MT work has focused on noise in sentence alignment rather than semantic divergence. Goutte et al. (2012) show that phrase-based systems are remarkably robust to noise in parallel segments. When introducing noise by permuting the target side of parallel pairs, as many as 30% of training examples had to be permuted to degrade BLEU significantly. While such artificial noise does not necessarily capture naturally occurring divergences, there is evidence that data cleaning to remove real noise can benefit MT in low-resource settings (Matthews et al., 2014).

Neural MT quality appears to be more sensitive to the nature of training examples than phrase-based systems. Chen et al. (2016) suggest that neural MT systems are sensitive to sentence pair permutations in domain adaptation settings. Belinkov and Bisk (2018) demonstrate the brittleness of character-level neural MT when exposed to synthetic noise (random permutations of words and characters) as well as natural human errors. Concurrently with our work, Hassan et al. (2018) claim that even small amounts of noise can have adverse effects on neural MT models, as they tend to assign high probabilities to rare events. They filter out noise and select relevant in-domain examples jointly, using similarities between sentence embeddings obtained from the encoder of a bidirectional neural MT system trained on clean in-domain data. In contrast, we detect semantic divergence with dedicated models that require only 5000 parallel examples (see Section 5).

This work builds on our initial study of semantic divergences (Carpuat et al., 2017), where we provide a framework for evaluating the impact of meaning mismatches in parallel segments on MT via data selection: we show that filtering out the most divergent segments in a training corpus improves translation quality. However, we previously detect mismatches using a cross-lingual entailment classifier, which is based on surface features only, and requires manually annotated training examples (Negri et al., 2012, 2013). In this paper, we detect divergences using a semantically-motivated model that can be trained given any existing parallel corpus without manual intervention.

3 Approach

We introduce our approach to detecting divergence in parallel sentences, with the goal of (1) detecting differences ranging from large mismatches to subtle nuances, (2) without manual annotation.

Cross-Lingual Semantic Similarity Model We address the first requirement using a neural model that compares the meaning of sentences using a range of granularities. We repurpose the Very Deep Pairwise Interaction (VDPWI) model, which has been previously been used to detect semantic textual similarity (STS) between English sentence pairs (He and Lin, 2016). It achieved competitive performance on data from the STS 2014 shared task (Agirre et al., 2014), and outperformed previous approaches on sentence classification tasks (He et al., 2015; Tai et al., 2015), with fewer parameters, faster training, and without requiring expensive external resources such as WordNet.

The VDPWI model was designed for comparing the meaning of sentences in the same language, based not only on word-to-word similarity comparisons, but also on comparisons between overlapping spans of the two sentences, as learned by a deep convolutional neural network. We adapt the model to our cross-lingual task by initializing it with bilingual embeddings. To the best of our knowledge, this is the first time this model has been used for cross-lingual tasks in such a way. We give a brief overview of the resulting model here and refer the reader to the original paper for details. Given sentences e and f , VDPWI models the semantic similarity between them using a pipeline consisting of five components:

1. **Bilingual word embeddings**: Each word in e and f is represented as a vector using pretrained, bilingual embeddings.
2. **BiLSTM for contextualizing words**: Contextualized representations for words in e and f are obtained by choosing the output vectors at each time step obtained by running a bidirectional LSTM (Schuster and Paliwal, 1997) on each sentence.
3. **Word similarity cube**: The contextualized representations are used to calculate various similarity scores between each word in *e* with each word in *f*. Each score thus forms a matrix and all such matrices are stacked to form a *similarity cube* tensor.
4. **Similarity focus layer**: The similarity cube is fed to a similarity focus layer that reweights the similarities in the cube to focus on highly similar word pairs, by decreasing the weights of pairs which are less similar. This output is called the *focus cube*.
5. **Deep convolutional network**: The focus cube is treated as an "image" and passed to a deep neural network, the likes of which have been used to detect patterns in images. The network consists of repeating convolution and pooling layers. Each repetition consists of a spatial convolutional layer, a Rectified Linear Unit (Nair and Hinton, 2010), and a max pooling layer, followed by fully connected layers, and a softmax to obtain the final output.

The entire architecture is trained end-to-end to minimize the Kullback-Leibler divergence (Kullback, 1959) between the output similarity score and gold similarity score.

Noisy Synthetic Supervision How can we obtain gold similarity scores as supervision for our task? We automatically construct examples of semantically divergent and equivalent sentences as follows. Since a large number of parallel sentence pairs are semantically equivalent, we use parallel sentences as positive examples. Synthetic negative examples are generated following the protocol introduced by Munteanu and Marcu (2005) to distinguish parallel from non-parallel segments. Specifically, candidate negative examples are generated starting from the positive examples $\{(e_i, f_i)\ \forall i\}$ and taking the Cartesian product of the two sides of the positive examples$\{(e_i, f_j) \forall i, j$ s.t. $i \neq j\}$. This candidate set is filtered to ensure that negative examples are not too easy to identify: we only retain pairs that are close to each other in length (a length ratio of at most 1:2), and have enough words (at least half) which have a translation in the other sentence according to a bilingual dictionary derived from automatic word alignments.

This process yields positive and negative examples that are a noisy source of supervision for our task, as some of the positive examples might not be fully equivalent in meaning. However, experiments will show that, in aggregate, they provide a useful signal for the VDPWI model to learn to detect semantic distinctions (Section 5).

		Equivalent with High Agreement ($n = 5$)
subs	en	the epidemic took my wife, my stepson.
	fr	l'épidémie a touché ma femme, mon beau-fils.
	gl	the epidemic touched my wife, my stepson.
		Equivalent with Low Agreement ($n = 3$)
cc	en	cancellation policy: if cancelled up to 28 days before date of arrival, no fee will be charged.
	fr	conditions d'annulation : en cas d'annulation jusqu'à 28 jours avant la date d'arrivée, l'hôtel ne prélève pas de frais sur la carte de crédit fournie.
	gl	cancellation conditions: in case of cancellation up to 28 days before arrival date, the hotel does not charge fees from the credit card given.
		Divergent with Low Agreement ($n = 3$)
cc	en	what does it mean when food is "low in ash" or "low in magnesium"?
	fr	quels sont les avantages dune nourriture "réduite en cendres" et "faible en magnésium" ?
	gl	what are the advantages of a food "low in ash" or "low in magnesium"?
		Divergent with High Agreement ($n = 5$)
subs	en	rabbit? if i told you it was a chicken, you wouldn't know the difference.
	fr	vous croirez manger du poulet.
	gl	you think eat chicken

Table 1: Randomly selected sentence pairs (English (en), French (fr) and gloss of French (gl)) annotated as divergent or equivalent, with high and low degrees of agreement between the 5 annotators. Examples are taken either from the OpenSubtitles (subs) or Common Crawl (cc) corpus.

4 Crowdsourcing Divergence Judgments

We crowdsource annotations of English-French sentence pairs to construct test beds for evaluating our models, and also to assess how frequent semantic divergences are in parallel corpora.

Data Selection We draw examples for annotation randomly from two English-French corpora, using a resource-rich and well-studied language pair, and for which bilingual annotators can easily be found. The **OpenSubtitles corpus** contains 33M sentence pairs based on translations of movie subtitles. The sentence pairs are expected to not be completely parallel given the many constraints imposed on translations that should fit on a screen and be synchronized with a movie (Tiedemann, 2007; Lison and Tiedemann, 2016), and the use of more informal registers which might require frequent non-literal translations of figurative language. The **Common Crawl corpus** contains sentence-aligned parallel documents automatically mined from the Internet. Parallel documents are discovered using e.g., URL containing language code patterns, and sentences are automatically aligned after structural cleaning of HTML. The resulting corpus of 3M sentence pairs is noisy, yet extremely useful to improve translation quality for multiple language pairs and domains (Smith et al., 2013).

Annotation Protocol Divergence annotations are obtained via Crowdflower.[1] Since this task requires good command of both French and English, we rely on a combination of strategies to obtain good quality annotations, including Crowdflower's internal worker proficiency ratings, geo-restriction, reference annotations by a bilingual speaker in our lab, and instructions that alternate between the two languages (Agirre et al., 2016).

Annotators are shown an English-French sentence pair, and asked whether they agree or disagree with the statement "the French and English text convey the same information." We do not use the term "divergent", and let the annotators' intuitions about what constitutes the same take precedence. We set up two distinct annotation tasks, one for each corpus, so that workers only see examples sampled from a given corpus in a given job. Each example is shown to five distinct annotators.

Annotation Analysis Forcing an assignment of divergent or equivalent labels by majority vote yields 43.6% divergent examples in OpenSubtitles, and 38.4% in Common Crawl. Fleiss' Kappa indicates moderate agreement between annotators (0.41 for OpenSubtitles and 0.49 for Common Crawl). This suggests that the annotation protocol can be improved, perhaps by using graded judgments as in Semantic Textual Similarity tasks (Agirre et al., 2016), or for sentence alignment

[1] http://crowdflower.com

confidence evaluation (Xu and Yvon, 2016). Current annotations are nevertheless useful, and different degrees of agreement reveal nuances in the nature of divergences (Table 1). Examples labeled as divergent with high confidence (lowest block of the table) are either unrelated or one language misses significant information that is present in the other. Examples labeled divergent with lower confidence contain more subtle differences (e.g., "what does it mean" in English vs. "what are the advantages" in French).

5 Divergence Detection Evaluation

Using the two test sets obtained above, we can evaluate the accuracy of our cross-lingual semantic divergence detector, and compare it against a diverse set of baselines in controlled settings. We test our hypothesis that semantic divergences are more than alignment mismatches by comparing the semantic divergence detector with models that capture mis-alignment (Section 5.2) or translation (Section 5.3). Then, we compare the deep convolutional architecture of the semantic divergence model, with a much simpler model that directly compares bilingual sentence embeddings (Section 5.4). Finally, we compare our model trained on synthetic examples with a supervised classifier used in prior work to predict finer-grained textual entailment categories based on manually created training examples (Section 5.5). Except for the entailment classifier which uses external resources, all models are trained on the exact same parallel corpora (OpenSubtitles or CommonCrawl for evaluating on the corresponding test bed.)

5.1 Neural Semantic Divergence Detection

Model and Training Settings We use the publicly available implementation of the VDPWI model.[2] We initialize with 200 dimensional BiVec French-English word embeddings (Luong et al., 2015), trained on the parallel corpus from which our test set is drawn. We use the default setting for all other VDPWI parameters. The model is trained for 25 epochs and the model that achieves the best Pearson correlation coefficient on the validation set is chosen. At test time, VDPWI outputs a score $\in [0, 1]$, where a higher value indicates less divergence. We tune a threshold on development data to convert the real-valued score to binary predictions.

Synthetic Data Generation The synthetic training data is constructed using a random sample of 5000 sentences from the training parallel corpus as positive examples. We generate negative examples automatically as described in Section 3, and sample a subset to maintain a 1:5 ratio of positive to negative examples.[3]

5.2 Parallel vs. Non-parallel Classifier

Are divergences observed in parallel corpora more than alignment errors? To answer this question, we reimplement the model proposed by Munteanu and Marcu (2005). It discriminates parallel pairs from non-parallel pairs in comparable corpora using a supervised linear classifier with the following features for each sentence pair (e, f):

- Length features: $|f|$, $|e|$, $\frac{|f|}{|e|}$, and $\frac{|e|}{|f|}$
- Alignment features (for each of e and f):[4]
  - Count and ratio of unaligned words
  - Top three largest fertilities
  - Longest contiguous unaligned and aligned sequence lengths
- Dictionary features:[5] fraction of words in e that have a translation in f and vice-versa.

Training uses the exact same synthetic examples as the semantic divergence model (Section 3).

5.3 Neural MT

If divergent examples are nothing more than bad translations, a neural MT system should assign lower scores to divergent segments pairs than to those that are equivalent in meaning. We test this empirically using neural MT systems trained for a single epoch, and use the system to score each of the sentence pairs in the test sets. We tune a threshold on the development set to convert scores to binary predictions. The system architecture and training settings are described in details in the later MT section (Section 6.2). Preliminary experiments showed that training for more than one epoch does not help divergence detection.

[2] https://github.com/castorini/VDPWI-NN-Torch

[3] We experimented with other ratios and found that the results only slightly degraded while using a more balanced ratio (1:1, 1:2), but severely degraded with a skewed ratio (1:9).

[4] Alignments are obtained using IBM Model 2 trained in each direction, combined with `union`, `intersection`, and `grow-diag-final-and` heuristics.

[5] The bilingual dictionary used to extract features is constructed using word alignments from a random sample of a million sentences from the training parallel corpus.

5.4 Bilingual Sentence Embeddings

Our semantic divergence model introduces a large number of parameters to combine the pairwise word comparisons into a single sentence-level prediction. This baseline tests whether a simpler model would suffice. We detect semantic divergence by computing the cosine similarity between sentence embeddings in a bilingual space. The sentence embeddings are bag-of-word representations, build by taking the mean of bilingual word embeddings for each word in the sentence. This approach has been shown to be effective, despite ignoring fundamental linguistic information such as word order and syntax (Mitchell and Lapata, 2010). We use the same 200 dimensional BiVec word embeddings (Luong et al., 2015), trained on OpenSubtitles and CommonCrawl respectively.

5.5 Textual Entailment Classifier

Our final baseline replicates our previous system (Carpuat et al., 2017) where we repurposed annotations and models designed for the task of Cross-Lingual Textual Entailment (CLTE) to detect semantic divergences. This baseline also helps us understand how the synthetic training data compares to training examples generated manually, for a related cross-lingual task. Using CLTE datasets from SemEval (Negri et al., 2012, 2013), we train a supervised linear classifier that can distinguish sentence pairs that entail each other, from pairs where entailment does not hold in at least one direction. The features of the classifier are based on word alignments and sentence lengths.[6]

5.6 Intrinsic Evaluation Results

Table 2 shows that the semantic similarity model is most successful at distinguishing equivalent from divergent examples. The break down per class shows that both equivalent and divergent examples are better detected. The improvement is larger for divergent examples with gains of about 10 points for F-score for the divergent class, when compared to the next-best scores.

Among the baseline methods, the non-entailment model is the weakest. While it uses manually constructed training examples, these examples are drawn from distant domains, and the categories annotated do not exactly match the task at hand. In contrast, the other models benefit from training on examples drawn from the same corpus as each test set.

Next, the machine translation based model and the sentence embedding model, both of which are unsupervised, are significantly weaker than the two supervised models trained on synthetic data, highlighting the benefits of the automatically constructed divergence examples. The strength of the semantic similarity model compared to the sentence embeddings model highlights the benefits of the fine-grained representation of bilingual sentence pairs as a similarity cube, as opposed to the bag-of-words sentence embedding representation.

Finally, despite training on the same noisy synthetic data as the parallel v/s non-parallel system, the semantic similarity model is better able to detect meaning divergences. This highlights the benefits of (1) meaning comparison between words in a shared embedding space, over the discrete translation dictionary used by the baseline, and of (2) the deep convolutional neural network which enables the semantic comparison of overlapping spans in sentence pairs, as opposed to more local word alignment features.

5.7 Analysis

We manually examine the 13-15% of examples in each test set that are correctly detected as divergent by semantic similarity and mis-classified by the non-parallel detector.

On OpenSubtitles, most of these examples are true divergences rather than noisy alignments (i.e. sentences that are not translation of each other.) The non-parallel detector weighs length features highly, and is fooled by sentence pairs of similar length that share little content and therefore have very sparse word alignments. The remaining sentence pairs are plausible translations in some context that still contain inherent divergences, such as details missing or added in one language. The non-parallel detector views these pairs as non-divergent since most words can be aligned. The semantic similarity model can identify subtle meaning differences, and correctly classify them as divergent. As a result, the non-parallel detector has a higher false positive rate (22%) than the semantic similarity classifier (14%), while having similar false negative rates (11% v/s 12%).

On the CommonCrawl test set, the examples with disagreement are more diverse, ranging from

[6] As in the prior work, alignments are trained on 2M sentence pairs from Europarl (Koehn, 2005) using the Berkeley aligner (Liang et al., 2006). The classifier is the linear SVM from Scikit-Learn.

Divergence Detection Approach	OpenSubtitles +P	+R	+F	-P	-R	-F	Overall F	Common Crawl +P	+R	+F	-P	-R	-F	Overall F
Sentence Embeddings	65	60	62	56	61	58	60	78	58	66	52	**74**	61	64
MT Scores (1 epoch)	67	53	59	54	68	60	60	54	65	59	17	11	14	42
Non-entailment	58	78	66	53	30	38	54	73	49	58	48	72	57	58
Non-parallel	70	83	76	61	42	50	66	70	83	76	61	42	49	67
Semantic Dissimilarity	**76**	**80**	**78**	**75**	**70**	**72**	**77**	**82**	**88**	**85**	**78**	69	**73**	**80**

Table 2: Intrinsic evaluation on crowdsourced semantic equivalence vs. divergence testsets. We report overall F-score, as well as precision (P), recall (R) and F-score (F) for the equivalent (+) and divergent (-) classes separately. Semantic similarity yields better results across the board, with larger improvements on the divergent class.

noisy segments that should not be aligned to sentences with subtle divergences.

6 Machine Translation Evaluation

Having established the effectiveness of the semantic divergence detector, we now measure the impact of divergences on a downstream task, machine translation. As in our prior work (Carpuat et al., 2017), we take a data selection approach, selecting the least divergent examples in a parallel corpus based on a range of divergence detectors, and comparing the translation quality of the resulting neural MT systems.

6.1 Translation Tasks

English-French We evaluate on 4867 sentences from the Microsoft Spoken Language Translation dataset (Federmann and Lewis, 2016) as well as on 1300 sentences from TED talks (Cettolo et al., 2012), as in past work (Carpuat et al., 2017). Training examples are drawn from OpenSubtitles, which contains ~28M examples after deduplication. We discard 50% examples for data selection.

Vietnamese-English Since the SEMANTIC SIMILARITY model was designed to be easily portable to new language pairs, we also test its impact on the IWSLT Vietnamese-English TED task, which comes with ~120,000 and 1268 in-domain sentences for training and testing respectively (Farajian et al., 2016). This is a more challenging translation task as Vietnamese and English are more distant languages, there is little training data, and the sentence pairs are expected to be cleaner and more parallel than those from OpenSubtitles. In these lower resource settings, we discard 10% of examples for data selection.

6.2 Neural MT System

We use the attentional encoder-decoder model (Bahdanau et al., 2015) implemented in the Sock-Eye toolkit (Hieber et al., 2017). Encoders and decoders are single-layer GRUs (Cho et al., 2014) with 1000 hidden units. Source and target word embeddings have size 512. Using byte-pair encoding (Sennrich et al., 2016), the vocabulary size is 50000. Maximum sequence length is set to 50.

We optimize the standard cross-entropy loss using Adam (Kingma and Ba, 2014), until validation perplexity does not decrease for 8 checkpoints. The learning rate is set to 0.0003 and is halved when the validation perplexity does not decrease for 3 checkpoints. The batch size is set to 80. At decoding time, we construct a new model by averaging the parameters for the 8 checkpoints with best validation perplexity, and decode with a beam of 5. All experiments are run thrice with distinct random seeds.

Note that the baseline in this work is much stronger than in our prior work (>5 BLEU). This is due to multiple factors that have been recommended as best practices for neural MT and have been incorporated in the present baseline – deduplication of the training data, ensemble decoding using multiple random runs, use of Adam as the optimizer instead of AdaDelta (Bahar et al., 2017; Denkowski and Neubig, 2017), and checkpoint averaging (Bahar et al., 2017) – as well as a more recent neural modeling toolkit.

6.3 English-French Results

We train English-French neural MT systems by selecting the least divergent half of the training corpus with the following criteria:

- SEMANTIC SIMILARITY (Section 3)
- PARALLEL: the non-parallel sentence detector (Section 5.2)
- ENTAILMENT: the entailment classifier (Section 5.5), as in Carpuat et al. (2017)
- RANDOM: Randomly downsampling the training corpus

Learning curves (Figure 1) show that data selected using SEMANTIC SIMILARITY yields better

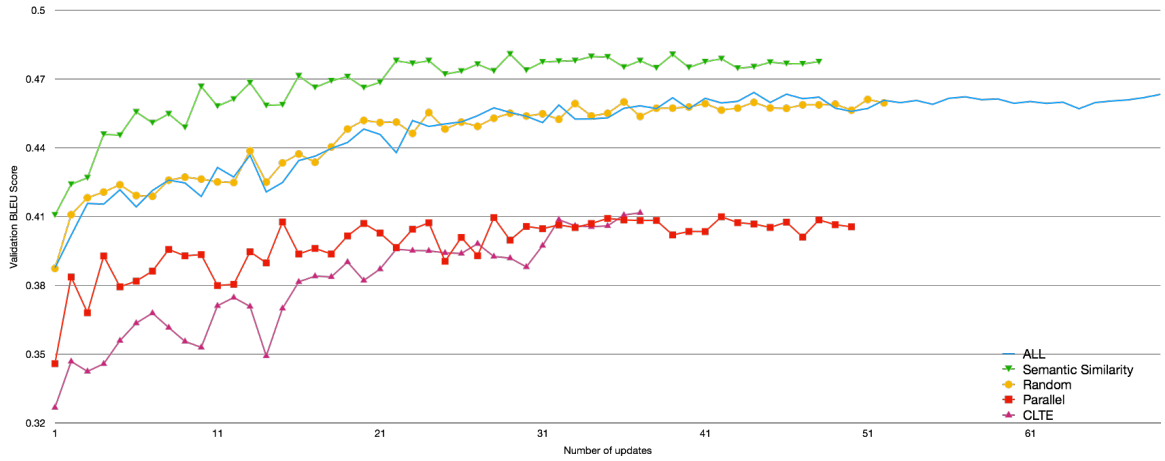


Figure 1: Learning curves on the validation set for English-French models (mean of 3 runs/model). The SEMANTIC SIMILARITY model outperforms other models throughout training, including the one trained on all data.

Model	MSLT BLEU		TED BLEU	
	Avg.	Ensemble	Avg.	Ensemble
RANDOM	43.49	45.64	36.05	38.20
PARALLEL	40.65	42.12	35.99	37.86
ENTAILMENT	39.64	41.86	33.30	35.40
SEMANTIC SIM.	**45.53**	**47.23***	**36.98**	**38.87**
ALL	44.64	46.26	36.98	38.59

Table 3: English-French decoding results. BLEU scores are either averaged across 3 runs ("Avg.") or obtained via ensemble decoding ("Ensemble"). SEMANTIC SIMILARITY reaches BLEU scores on par with ALL with only half of the training data. SEMANTIC SIMILARITY scores marked with * are significanly better ($p < 0.05$) than the corresponding ALL scores.

validation BLEU throughout training compared to all other models. SEMANTIC SIMILARITY selects more useful examples for MT than PARALLEL, even though both selection models are trained on the same synthetic examples. This highlights the benefits of semantic modeling over surface mis-alignment features. Furthermore, SEMANTIC SIMILARITY achieves the final validation BLEU of the model that uses ALL data with only 30% of the updates. This suggests that semantically divergent examples are pervasive in the training corpus, confirming the findings from manual annotation (Section 4), and that the presence of such examples slows down neural MT training.

Decoding results on the blind test sets (Table 3) show that SEMANTIC SIMILARITY outperforms all other data selection criteria (with differences being statistically significant ($p < 0.05$) (Koehn, 2004)), and performs as well or better than the ALL model which has access to twice as many training examples.

Model	Avg. Test Set BLEU
RANDOM (90%)	22.71
SEMANTIC SIM. (90%)	**23.38**
ALL	23.30

Table 4: Vietnamese-English decoding results: dropping 10% of the data based on SEMANTIC SIMILARITY does not hurt BLEU, and performs significantly ($p < 0.05$) better than RANDOM selection.

The SEMANTIC SIMILARITY model also achieves significantly better translation quality than the ENTAILMENT model used in our prior work. Surprisingly, the ENTAILMENT model performs worse than the ALL baseline, unlike in our prior work. We attribute this different behavior to several factors: the strength of the new baseline (Section 6.2), the use of Adam instead of AdaDelta, which results in stronger BLEU scores at the beginning of the learning curves for all models, and finally the deduplication of the training data. In our prior systems, the training corpus was not deduplicated. Data selection had a side-effect of reducing the ratio of duplicated examples. When the ENTAILMENT model was used, longer sentence pairs with more balanced length were selected, yielding longer translations with a better BLEU brevity penalty than the baseline. With the new systems, these advantages vanish. We further analyze output lengths in Section 6.5.

6.4 Vietnamese-English Results

Trends from English-French carry over to Vietnamese English, as the SEMANTIC SIMILARITY model compares favorably to ALL while reducing the number of training updates by 10%. SEMAN-

TIC SIMILARITY also yields better BLEU than RANDOM with the differences being statistically significant. While differences in score here are smaller, these result are encouraging since they demonstrate that our semantic divergence models port to more distant low-resource language pairs.

6.5 Analysis

We break down the results seen in Figure 1 and Table 3, with a focus on the behavior of the ENTAILMENT and ALL models. We start by analyzing the BLEU brevity penalty trends observed on the validation set during training (Figure 2).

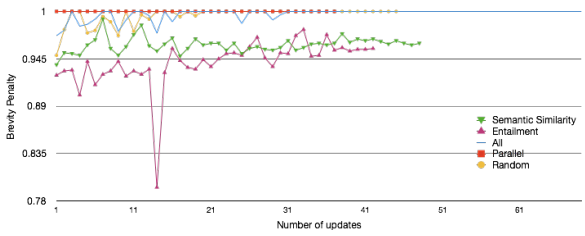


Figure 2: Brevity penalties on the validation set for English-French models (single run).

We observe that both the ENTAILMENT and SEMANTIC SIMILARITY based models have similar brevity penalties despite having performances that are at opposite ends of the spectrum in terms of BLEU. This implies that translations generated by the SEMANTIC SIMILARITY model have better n-gram overlap with the reference, but are much shorter. Manual examination of the translations suggests that the ENTAILMENT model often fails by under-translating sentences, either dropping segments from the beginning or the end of source sentences (Table 5).

The PARALLEL model consistently produces translations that are longer than the reference. [7] This is partially due to the model's propensity to generate a sequence of garbage tokens in the beginning of a sentence, especially while translating shorter sentences. In our test set, almost 12% of the translated sentences were found to begin with the garbage text shown in Table 5. Only a small fraction ($< 0.02\%$) of the French sentences in our training data begin with these tokens, but the tendency of PARALLEL to promote divergent examples above non-divergent ones, seems to exaggerate the generation of this sequence.

[7] The brevity penalty does not penalize translations that are longer than the reference.

ENTAILMENT is inadequate due to under-translation	
Source	he's a very impressive man **and still goes out to do digs.**
Reference	c'est un homme très impressionnant **et il fait encore des fouilles.**
ENTAILMENT	c'est un homme très impressionnant.
Source	when the Heat **first** won.
Reference	lorsque les Heat ont gagné **pour la première fois**.
ENTAILMENT	quand le Heat a gagné.
PARALLEL produces garbage tokens	
Source	alright.
Reference	d'accord.
ENTAILMENT	{ \ **pos (192,210)**} d'accord.

Table 5: Selected translation examples from the ensemble systems of the various models.

7 Conclusion

We conducted an extensive empirical study of semantic divergences in parallel corpora. Our crowdsourced annotations confirms that correctly aligned sentences are not necessarily meaning equivalent. We introduced an approach based on neural semantic similarity that detects such divergences much more accurately than shallower translation or alignment based models. Importantly, our model does not require manual annotation, and can be trained for any language pair and domain with a parallel corpus. Finally, we show that filtering out divergent examples helps speed up the convergence of neural machine translation training without loss in translation quality, for two language pairs and data conditions. New datasets and models introduced in this work are available at `http://github.com/yogarshi/SemDiverge`.

These findings open several avenues for future work: How can we improve divergence detection further? Can we characterize the nature of the divergences beyond binary predictions? How do divergent examples impact other applications, including cross-lingual NLP applications and semantic models induced from parallel corpora, as well as tools for human translators and second language learners?

Acknowledgments

We thank the CLIP lab at the University of Maryland and the anonymous reviewers from NAACL 2018 and WMT 2017 for their constructive feedback. This work was supported in part by research awards from Amazon, Google, and the Clare Boothe Luce Foundation.

References

Sadaf AbduI-Rauf and Holger Schwenk. 2009. On the Use of Comparable Corpora to Improve SMT Performance. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Athens, Greece, EACL '09, pages 16–23.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. Semeval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Association for Computational Linguistics Dublin, Ireland, pages 81–91.

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. SemEval-2016 Task 1: Semantic Textual Similarity, Monolingual and Cross-Lingual Evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. Association for Computational Linguistics, San Diego, California, pages 497–511.

Parnia Bahar, Tamer Alkhouli, Jan-Thorsten Peter, Christopher Jan-Steffen Brix, and Hermann Ney. 2017. Empirical investigation of optimization algorithms in neural machine translation. *The Prague Bulletin of Mathematical Linguistics* 108(1):13–25.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Learning Representations (ICLR)*.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *Proceedings of the 6th International Conference on Learning Representations*. Vancouver, Canada.

Marine Carpuat, Yogarshi Vyas, and Xing Niu. 2017. Detecting cross-lingual semantic divergence for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*. Association for Computational Linguistics, pages 69–79. http://aclweb.org/anthology/W17-3209.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*. volume 261, page 268.

Boxing Chen, Roland Kuhn, George Foster, Colin Cherry, and Fei Huang. 2016. Bilingual Methods for Adaptive Training Data Selection for Machine Translation. *Proceedings of the 12th Conference of the Association for Machine Translation in the Americas (AMTA)* page 93.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of EMNLP 2014*. ArXiv: 1406.1078. http://arxiv.org/abs/1406.1078.

Michael Denkowski and Graham Neubig. 2017. Stronger baselines for trustable results in neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*. Association for Computational Linguistics, pages 18–27. http://aclweb.org/anthology/W17-3203.

Bonnie J. Dorr. 1994. Machine Translation Divergences: A Formal Description and Proposed Solution. *Computional Linguistics* 20(4):597–633.

Cristina España-Bonet, Ádám Csaba Varga, Alberto Barrón-Cedeño, and Josef van Genabith. 2017. An Empirical Analysis of NMT-Derived Interlingual Embeddings and their Use in Parallel Sentence Identification. *arXiv:1704.05415 [cs]* .

M. Amin Farajian, Rajen Chatterjee, Costanza Conforti, Shahab Jalalvand, Vevake Balaraman, Mattia A. Di Gangi, Duygu Ataman, Marco Turchi, Matteo Negri, and Marcello Federico. 2016. FBK's Neural Machine Translation Systems for IWSLT 2016. In *Proceedings of the Ninth International Workshop on Spoken Language Translation (IWSLT), Seattle, WA*.

Christian Federmann and William D. Lewis. 2016. Microsoft speech language translation (MSLT) corpus: The IWSLT 2016 release for English, French and German. In *International Workshop on Spoken Language Translation (IWSLT)*. Seattle, USA.

Pascale Fung and Percy Cheung. 2004. Multi-level Bootstrapping for Extracting Parallel Sentences from a Quasi-comparable Corpus. In *Proceedings of the 20th International Conference on Computational Linguistics*. Association for Computational Linguistics, Geneva, Switzerland, COLING '04. https://doi.org/10.3115/1220355.1220506.

Cyril Goutte, Marine Carpuat, and George Foster. 2012. The Impact of Sentence Alignment Errors on Phrase-Based Machine Translation Performance. In *Proceedings of AMTA-2012: The Tenth Biennial Conference of the Association for Machine Translation in the Americas*.

Nizar Habash and Bonnie Dorr. 2002. Handling translation divergences: Combining statistical and symbolic techniques in generation-heavy machine translation. In *Conference of the Association for Machine Translation in the Americas*. Springer, pages 84–93.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt,

William Lewis, Mu Li, Shujie Liu, Tie-Yan Liu, Renqian Luo, Arul Menezes, Tao Qin, Frank Seide, Xu Tan, Fei Tian, Lijun Wu, Shuangzhi Wu, Yingce Xia, Dongdong Zhang, Zhirui Zhang, and Ming Zhou. 2018. Achieving Human Parity on Automatic Chinese to English News Translation. *arXiv:1803.05567 [cs]* https://arxiv.org/pdf/1803.05567.pdf.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-Perspective Sentence Similarity Modeling with Convolutional Neural Networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1576–1586.

Hua He and Jimmy Lin. 2016. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *Proceedings of NAACL-HLT*. pages 937–948.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2017. Sockeye: A Toolkit for Neural Machine Translation. *ArXiv e-prints* https://arxiv.org/abs/1712.05690.

Graeme Hirst. 1995. Near-synonymy and the structure of lexical knowledge. In *AAAI Symposium on Representation and Acquisition of Lexical Knowledge: Polysemy, Ambiguity, and Generativity*. pages 51–56.

Sergio Jimenez, Claudia Becerra, and Alexander Gelbukh. 2012. Soft Cardinality + ML: Learning Adaptive Similarity Functions for Cross-lingual Textual Entailment. In *Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval)*. Association for Computational Linguistics, Montreal, Canada, SemEval '12, pages 684–688.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. ArXiv: 1412.6980. http://arxiv.org/abs/1412.6980.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP 2004*. http://www.aclweb.org/anthology/W04-3250.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Machine Translation Summit X*. Phuket, Thailand.

Moshe Koppel and Noam Ordan. 2011. Translationese and its dialects. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, HLT '11, pages 1318–1326. http://dl.acm.org/citation.cfm?id=2002472.2002636.

Solomon Kullback. 1959. *Information theory and statistics*. Courier Corporation.

Ekaterina Lapshinova-Koltunski. 2015. Exploration of inter-and intralingual variation of discourse phenomena. In *Proceedings of the Second Workshop on Discourse in Machine Translation*. pages 158–167.

Junyi Jessy Li, Marine Carpuat, and Ani Nenkova. 2014. Assessing the Discourse Factors that Influence the Quality of Machine Translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 283–288.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by Agreement. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, HLT-NAACL '06, pages 104–111. https://doi.org/10.3115/1220835.1220849.

Pierre Lison and Jörg Tiedemann. 2016. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. In *Proceedings of the 10th International Conference on Language Resources and Evaluation*.

Chi-kiu Lo, Cyril Goutte, and Michel Simard. 2016. CNRC at SemEval-2016 Task 1: Experiments in Crosslingual Semantic Textual Similarity. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. Association for Computational Linguistics, San Diego, California, pages 668–673.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*. pages 151–159.

Austin Matthews, Waleed Ammar, Archna Bhatia, Weston Feely, Greg Hanneman, Eva Schlinger, Swabha Swayamdipta, Yulia Tsvetkov, Alon Lavie, and Chris Dyer. 2014. The CMU Machine Translation Systems at WMT 2014. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Baltimore, Maryland, USA, pages 142–149.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive science* 34(8):1388–1429.

Dragos Stefan Munteanu and Daniel Marcu. 2005. Improving machine translation performance by exploiting non-parallel corpora. *Computational Linguistics* 31(4):477–504.

Dragos Stefan Munteanu and Daniel Marcu. 2006. Extracting Parallel Sub-sentential Fragments from

Non-parallel Corpora. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Sydney, Australia, ACL-44, pages 81–88. https://doi.org/10.3115/1220175.1220186.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*. Omnipress, USA, ICML'10, pages 807–814.

Matteo Negri, Alessandro Marchetti, Yashar Mehdad, Luisa Bentivogli, and Danilo Giampiccolo. 2012. Semeval-2012 Task 8: Cross-lingual Textual Entailment for Content Synchronization. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*. Association for Computational Linguistics, Montr\?al, Canada, SemEval '12, pages 399–407.

Matteo Negri, Alessandro Marchetti, Yashar Mehdad, Luisa Bentivogli, and Danilo Giampiccolo. 2013. Semeval-2013 Task 8: Cross-lingual Textual Entailment for Content Synchronization. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. Association for Computational Linguistics, Atlanta, Georgia, USA, pages 25–33.

Matteo Negri and Yashar Mehdad. 2010. Creating a Bilingual Entailment Corpus Through Translations with Mechanical Turk: $100 for a 10-day Rush. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*. Association for Computational Linguistics, Los Angeles, California, CSLDAMT '10, pages 212–216.

Jason Riesa and Daniel Marcu. 2012. Automatic Parallel Fragment Extraction from Noisy Data. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Montréal, Canada, pages 538–542.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45(11):2673–2681.

Holger Schwenk and Matthijs Douze. 2017. Learning joint multilingual sentence representations with neural machine translation. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*. Association for Computational Linguistics, pages 157–167. http://aclweb.org/anthology/W17-2619.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. *Proceedings of the Meeting of the Association for Computational Linguistics (ACL)* .

Jason R. Smith, Chris Quirk, and Kristina Toutanova. 2010. Extracting Parallel Sentences from Comparable Corpora Using Document Level Alignment. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Los Angeles, California, HLT '10, pages 403–411.

Jason R. Smith, Herve Saint-Amand, Magdalena Plamada, Philipp Koehn, Chris Callison-Burch, and Adam Lopez. 2013. Dirt Cheap Web-Scale Parallel Text from the Common Crawl. In *Proceedings of 51st Meeting of the Association for Computational Linguistics*. pages 1374–1383.

Lucia Specia, Varvara Logacheva, and Carolina Scarton. 2016. WMT16 Quality Estimation Shared Task Training and Development Data. LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague.

Lucia Specia, Dhwaj Raj, and Marco Turchi. 2010. Machine translation evaluation versus quality estimation. *Machine Translation* 24(1):39–50. https://doi.org/10.1007/s10590-010-9077-2.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of ACL 2015*. Beijing, China.

Jörg Tiedemann. 2007. Improved sentence alignment for movie subtitles. In *Recent Advances in Natural Language Processing (RANLP)*. volume 7.

Jörg Tiedemann. 2011. *Bitext Alignment*, volume 4.

Marco Turchi and Matteo Negri. 2013. ALTN: Word Alignment Features for Cross-lingual Textual Entailment. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Seventh International Workshop on Semantic Evaluation (SemEval 2013)*.

Yong Xu and François Yvon. 2016. Novel elicitation and annotation schemes for sentential and sub-sentential alignments of bitexts. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth Language Resources and Evaluation Conference (LREC 2016)*. European Language Resources Association (ELRA), Portorož, Slovenia, page 10.

Jiang Zhao, Man Lan, and Zheng-yu Niu. 2013. ECNUCS: Recognizing cross-lingual textual entailment using multiple text similarity and text difference measures. In *Proceedings of the International Workshop on Semantic Evaluation (SemEval)*.

Bootstrapping Generators from Noisy Data

Laura Perez-Beltrachini and **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
{lperez,mlap}@inf.ed.ac.uk

Abstract

A core step in statistical data-to-text generation concerns learning correspondences between structured data representations (e.g., facts in a database) and associated texts. In this paper we aim to bootstrap generators from large scale datasets where the data (e.g., DBPedia facts) and related texts (e.g., Wikipedia abstracts) are loosely aligned. We tackle this challenging task by introducing a special-purpose content selection mechanism.[1] We use multi-instance learning to automatically discover correspondences between data and text pairs and show how these can be used to enhance the content signal while training an encoder-decoder architecture. Experimental results demonstrate that models trained with content-specific objectives improve upon a vanilla encoder-decoder which solely relies on soft attention.

1 Introduction

A core step in statistical data-to-text generation concerns learning correspondences between structured data representations (e.g., facts in a database) and paired texts (Barzilay and Lapata, 2005; Kim and Mooney, 2010; Liang et al., 2009). These correspondences describe how data representations are expressed in natural language (*content realisation*) but also indicate which subset of the data is verbalised in the text (*content selection*).

Although content selection is traditionally performed by domain experts, recent advances in generation using neural networks (Bahdanau et al., 2015; Ranzato et al., 2016) have led to the use of large scale datasets containing loosely related data and text pairs. A prime example are online data sources like DBPedia (Auer et al., 2007) and Wikipedia and their associated texts which are often independently edited. Another example are sports databases and related textual resources. Wiseman et al. (2017) recently define a generation task relating statistics of basketball games with commentaries and a blog written by fans.

In this paper, we focus on short text generation from such loosely aligned data-text resources. We work with the biographical subset of the DBPedia and Wikipedia resources where the data corresponds to DBPedia facts and texts are Wikipedia abstracts about people. Figure 1 shows an example for the film-maker *Robert Flaherty*, the Wikipedia infobox, and the corresponding abstract. We wish to bootstrap a data-to-text generator that learns to verbalise properties about an entity from a loosely related example text. Given the set of properties in Figure (1a) and the related text in Figure (1b), we want to learn verbalisations for those properties that are mentioned in the text and produce a short description like the one in Figure (1c).

In common with previous work (Mei et al., 2016; Lebret et al., 2016; Wiseman et al., 2017) our model draws on insights from neural machine translation (Bahdanau et al., 2015; Sutskever et al., 2014) using an encoder-decoder architecture as its backbone. Lebret et al. (2016) introduce the task of generating biographies from Wikipedia data, however they focus on single sentence generation. We generalize the task to multi-sentence text, and highlight the limitations of the standard attention mechanism which is often used as a proxy for content selection. When exposed to sub-sequences that do not correspond to any facts in the input, the soft attention mechanism will still try to justify the sequence and somehow distribute the attention weights over the input representation (Ghader and Monz, 2017). The decoder will still memorise high frequency sub-sequences in spite of these not being supported by any facts in the input.

We propose to alleviate these shortcom-

[1] Our code and data are available at https://github.com/EdinburghNLP/wikigen.

Proceedings of NAACL-HLT 2018, pages 1516–1527
New Orleans, Louisiana, June 1 - 6, 2018.

(a)

Born	Robert Joseph Flaherty February 16, 1884 Iron Mountain, Michigan, U.S.
Died	July 23, 1951 (aged 67) Dummerston, Vermont, U.S.
Cause of death	Cerebral thrombosis
Occupation	Filmmaker
Spouse(s)	Frances Johnson Hubbard

(b) Robert Joseph Flaherty, (February 16, 1884 July 23, 1951) was an American film-maker who directed and produced the first commercially successful feature-length documentary film, Nanook of the North (1922). The film made his reputation and nothing in his later life fully equalled its success, although he continued the development of this new genre of narrative documentary, e.g., with Moana (1926), set in the South Seas, and Man of Aran (1934), filmed in Ireland's Aran Islands. He is considered the "father" of both the documentary and the ethnographic film. Flaherty was married to writer Frances H. Flaherty from 1914 until his death in 1951. Frances worked on several of her husband's films, and received an Academy Award nomination for Best Original Story for Louisiana Story (1948).

(c) *Robert Joseph Flaherty, (February 16, 1884 July 23, 1951) was an American film-maker. Flaherty was married to Frances H. Flaherty until his death in 1951.*

Figure 1: Property-value pairs (a), related biographic abstract (b) for the Wikipedia entity *Robert Flaherty*, and model verbalisation in italics (c).

ings via a specific content selection mechanism based on multi-instance learning (MIL; Keeler and Rumelhart, 1992) which automatically discovers correspondences, namely alignments, between data and text pairs. These alignments are then used to modify the generation function during training. We experiment with two frameworks that allow to incorporate alignment information, namely multi-task learning (MTL; Caruana, 1993) and reinforcement learning (RL; Williams, 1992). In both cases we define novel objective functions using the learnt alignments. Experimental results using automatic and human-based evaluation show that models trained with content-specific objectives improve upon vanilla encoder-decoder architectures which rely solely on soft attention.

The remainder of this paper is organised as follows. We discuss related work in Section 2 and describe the MIL-based content selection approach in Section 3. We explain how the generator is trained in Section 4 and present evaluation experiments in Section 5. Section 7 concludes the paper.

2 Related Work

Previous attempts to exploit loosely aligned data and text corpora have mostly focused on extracting verbalisation spans for data units. Most approaches work in two stages: initially, data units are aligned with sentences from related corpora using some heuristics and subsequently extra content is discarded in order to retain only text spans verbalising the data. Belz and Kow (2010) obtain verbalisation spans using a measure of strength of association between data units and words, Walter et al. (2013) extract textual patterns from paths in dependency trees while Mrabet et al. (2016) rely on crowd-sourcing. Perez-Beltrachini and Gardent (2016) learn shared representations for data units and sentences reduced to subject-predicate-object triples with the aim of extracting verbalisations for knowledge base properties. Our work takes a step further, we not only induce data-to-text alignments but also learn generators that produce short texts verbalising a set of facts.

Our work is closest to recent neural network models which learn generators from independently edited data and text resources. Most previous work (Lebret et al., 2016; Chisholm et al., 2017; Sha et al., 2017; Liu et al., 2017) targets the generation of single sentence biographies from Wikipedia infoboxes, while Wiseman et al. (2017) generate game summary documents from a database of basketball games where the input is always the same set of table fields. In contrast, in our scenario, the input data varies from one entity (e.g., athlete) to another (e.g., scientist) and properties might be present or not due to data incompleteness. Moreover, our generator is enhanced with a content selection mechanism based on multi-instance learning. MIL-based techniques have been previously applied to a variety of problems including image retrieval (Maron and Ratan, 1998; Zhang et al., 2002), object detection (Carbonetto et al., 2008; Cour et al., 2011), text classification (Andrews and Hofmann, 2004), image captioning (Wu et al., 2015; Karpathy and Fei-Fei, 2015), paraphrase detection (Xu et al., 2014), and information extraction (Hoffmann et al., 2011). The application of MIL to content selection is novel to our knowledge.

We show how to incorporate content selection into encoder-decoder architectures following training regimes based on multi-task learning and reinforcement learning. Multi-task learning aims to improve a main task by incorporating joint learning of one or more related auxiliary tasks. It has been applied with success to a variety of sequence-prediction tasks focus-

ing mostly on morphosyntax. Examples include chunking, tagging (Collobert et al., 2011; Søgaard and Goldberg, 2016; Bjerva et al., 2016; Plank, 2016), name error detection (Cheng et al., 2015), and machine translation (Luong et al., 2016). Reinforcement learning (Williams, 1992) has also seen popularity as a means of training neural networks to directly optimize a task-specific metric (Ranzato et al., 2016) or to inject task-specific knowledge (Zhang and Lapata, 2017). We are not aware of any work that compares the two training methods directly. Furthermore, our reinforcement learning-based algorithm differs from previous text generation approaches (Ranzato et al., 2016; Zhang and Lapata, 2017) in that it is applied to documents rather than individual sentences.

3 Bidirectional Content Selection

We consider loosely coupled data and text pairs where the data component is a set $\mathcal{P}$ of property-values $\{p_1 : v_1, \cdots, p_{|\mathcal{P}|} : v_{|\mathcal{P}|}\}$ and the related text $\mathcal{T}$ is a sequence of sentences $(s_1, \cdots, s_{|\mathcal{T}|})$. We define a *mention span* τ as a (possibly discontinuous) subsequence of $\mathcal{T}$ containing one or several words that verbalise one or more property-value from $\mathcal{P}$. For instance, in Figure 1, the mention span *"married to Frances H. Flaherty"* verbalises the property-value $\{Spouse(s) : Frances\ Johnson\ Hubbard\}$.

In traditional supervised data to text generation tasks, data units (e.g., $p_i : v_i$ in our particular setting) are either covered by some mention span τ_j or do not have any mention span at all in $\mathcal{T}$. The latter is a case of content selection where the generator will learn which properties to ignore when generating text from such data. In this work, we consider text components which are independently edited, and will unavoidably contain *unaligned spans*, i.e., text segments which do not correspond to any property-value in $\mathcal{P}$. The phrase *"from 1914"* in the text in Figure (1b) is such an example. Similarly, the last sentence, talks about Frances' awards and nominations and this information is not supported by the properties either.

Our model checks content in both directions; it identifies which properties have a corresponding text span (data selection) and also foregrounds (un)aligned text spans (text selection). This knowledge is then used to discourage the generator from producing text not supported by facts in the property set $\mathcal{P}$. We view a property set $\mathcal{P}$ and its loosely coupled text $\mathcal{T}$ as a coarse level, imperfect alignment. From this alignment signal, we want to discover a set of finer grained alignments indicating which mention spans in $\mathcal{T}$ align to which properties in $\mathcal{P}$. For each pair $(\mathcal{P}, \mathcal{T})$, we learn an alignment set $\mathcal{A}(\mathcal{P}, \mathcal{T})$ which contains property-value word pairs. For example, for the properties *spouse* and *died* in Figure 1, we would like to derive the alignments in Table 1.

married	$spouse : FrancesJohnsonFlaherty$
to	$spouse : FrancesJohnsonFlaherty$
Frances	$spouse : FrancesJohnsonFlaherty$
Flaherty	$spouse : FrancesJohnsonFlaherty$
death	$died : july23, 1951$
in	$died : july23, 1951$
1951	$died : july23, 1951$

Table 1: Example of word-property alignments for the Wikipedia abstract and facts in Figure 1.

We formulate the task of discovering finer-grained word alignments as a multi-instance learning problem (Keeler and Rumelhart, 1992). We assume that words from the text are positive labels for some property-values but we do not know which ones. For each data-text pair $(\mathcal{P}, \mathcal{T})$, we derive $|\mathcal{T}|$ pairs of the form $(\mathcal{P}, s)$ where $|\mathcal{T}|$ is the number of sentences in $\mathcal{T}$. We encode property sets $\mathcal{P}$ and sentences s into a common multimodal h-dimensional embedding space. While doing this, we discover finer grained alignments between words and property-values. The intuition is that by learning a high similarity score for a property set $\mathcal{P}$ and sentence pair s, we will also learn the contribution of individual elements (i.e., words and property-values) to the overall similarity score. We will then use this individual contribution as a measure of word and property-value alignment. More concretely, we assume the pair is aligned (or unaligned) if this individual score is above (or below) a given threshold. Across examples like the one shown in Figure (1a-b), we expect the model to learn an alignment between the text span *"married to Frances H. Flaherty"* and the property-value $\{spouse : Frances\ Johnson\ Hubbard\}$.

In what follows we describe how we encode $(\mathcal{P}, s)$ pairs and define the similarity function.

Property Set Encoder As there is no fixed order among the property-value pairs $p : v$ in $\mathcal{P}$, we individually encode each one of them. Furthermore, both properties p and values v may consist of short phrases. For instance, the property $cause\ of\ death$ and value *cerebral thrombosis* in Figure 1. We

therefore consider property-value pairs as concatenated sequences pv and use a bidirectional Long Short-Term Memory Network (LSTM; Hochreiter and Schmidhuber, 1997) network for their encoding. Note that the same network is used for all pairs. Each property-value pair is encoded into a vector representation:

$$\mathbf{p}_i = \text{biLSTM}_{denc}(pv_i) \quad (1)$$

which is the output of the recurrent network at the final time step. We use addition to combine the forward and backward outputs and generate encoding $\{\mathbf{p}_1, \cdots, \mathbf{p}_{|\mathcal{P}|}\}$ for $\mathcal{P}$.

Sentence Encoder We also use a biLSTM to obtain a representation for the sentence $s = w_1, \cdots, w_{|s|}$. Each word w_t is represented by the output of the forward and backward networks at time step t. A word at position t is represented by the concatenation of the forward and backward outputs of the networks at time step t :

$$\mathbf{w}_t = \text{biLSTM}_{senc}(w_t) \quad (2)$$

and each sentence is encoded as a sequence of vectors $(\mathbf{w}_1, \cdots, \mathbf{w}_{|s|})$.

Alignment Objective Our learning objective seeks to maximise the similarity score between property set $\mathcal{P}$ and a sentence s (Karpathy and Fei-Fei, 2015). This similarity score is in turn defined on top of the similarity scores among property-values in $\mathcal{P}$ and words in s. Equation (3) defines this similarity function using the dot product. The function seeks to align each word to the best scoring property-value:

$$S_{\mathcal{P}s} = \sum_{t=1}^{|s|} max_{i \in \{1,\ldots,|\mathcal{P}|\}} \mathbf{p}_i \cdot \mathbf{w}_t \quad (3)$$

Equation (4) defines our objective which encourages related properties $\mathcal{P}$ and sentences s to have higher similarity than other $\mathcal{P}' \neq \mathcal{P}$ and $s' \neq s$:

$$\begin{aligned} \mathcal{L}_{CA} = & max(0, S_{\mathcal{P}s} - S_{\mathcal{P}s'} + 1) \\ & + max(0, S_{\mathcal{P}s} - S_{\mathcal{P}'s} + 1) \end{aligned} \quad (4)$$

4 Generator Training

In this section we describe the base generation architecture and explain two alternative ways of using the alignments to guide the training of the model. One approach follows multi-task training where the generator learns to output a sequence of words but also to predict alignment labels for each word. The second approach relies on reinforcement learning for adjusting the probability distribution of word sequences learnt by a standard word prediction training algorithm.

4.1 Encoder-Decoder Base Generator

We follow a standard attention based encoder-decoder architecture for our generator (Bahdanau et al., 2015; Luong et al., 2015). Given a set of properties X as input, the model learns to predict an output word sequence Y which is a verbalisation of (part of) the input. More precisely, the generation of sequence Y is conditioned on input X:

$$P(Y|X) = \prod_{t=1}^{|Y|} P(y_t|y_{1:t-1}, X) \quad (5)$$

The encoder module constitutes an intermediate representation of the input. For this, we use the property-set encoder described in Section 3 which outputs vector representations $\{\mathbf{p}_1, \cdots, \mathbf{p}_{|X|}\}$ for a set of property-value pairs. The decoder uses an LSTM and a soft attention mechanism (Luong et al., 2015) to generate one word y_t at a time conditioned on the previous output words and a context vector c_t dynamically created:

$$P(y_{t+1}|y_{1:t}, X) = softmax(g(\mathbf{h}_t, c_t)) \quad (6)$$

where $g(\cdot)$ is a neural network with one hidden layer parametrised by $\mathbf{W}_o \in \mathbb{R}^{|V| \times d}$, $|V|$ is the output vocabulary size and d the hidden unit dimension, over $\mathbf{h}_t$ and c_t composed as follows:

$$g(\mathbf{h}_t, c_t) = \mathbf{W}_o \, tanh(\mathbf{W}_c[c_t; \mathbf{h}_t]) \quad (7)$$

where $\mathbf{W}_c \in \mathbb{R}^{d \times 2d}$. $\mathbf{h}_t$ is the hidden state of the LSTM decoder which summarises $y_{1:t}$:

$$\mathbf{h}_t = \text{LSTM}(y_t, \mathbf{h}_{t-1}) \quad (8)$$

The dynamic context vector c_t is the weighted sum of the hidden states of the input property set (Equation (9)); and the weights α_{ti} are determined by a dot product attention mechanism:

$$c_t = \sum_{i=1}^{|X|} \alpha_{ti} \mathbf{p}_i \quad (9)$$

$$\alpha_{ti} = \frac{\exp(\mathbf{h}_t \cdot \mathbf{p}_i)}{\sum_{i'} \exp(\mathbf{h}_t \cdot \mathbf{p}_{i'})} \quad (10)$$

We initialise the decoder with the averaged sum of the encoded input representations (Vinyals et al., 2016). The model is trained to optimize negative log likelihood:

$$\mathcal{L}_{wNLL} = -\sum_{t=1}^{|Y|} log\, P(y_t|y_{1:t-1}, X) \quad (11)$$

We extend this architecture to multi-sentence texts in a way similar to Wiseman et al. (2017). We view the abstract as a single sequence, i.e., all sentences are concatenated. When training, we cut the abstracts in blocks of equal size and perform forward backward iterations for each block (this includes the back-propagation through the encoder). From one block iteration to the next, we initialise the decoder with the last state of the previous block. The block size is a hyperparameter tuned experimentally on the development set.

4.2 Predicting Alignment Labels

The generation of the output sequence is conditioned on the previous words and the input. However, when certain sequences are very common, the language modelling conditional probability will prevail over the input conditioning. For instance, the phrase *from 1914* in our running example is very common in contexts that talk about periods of marriage or club membership, and as a result, the language model will output this phrase often, even in cases where there are no supporting facts in the input. The intuition behind multi-task training (Caruana, 1993) is that it will smooth the probabilities of frequent sequences when trying to simultaneously predict alignment labels.

Using the set of alignments obtained by our content selection model, we associate each word in the training data with a binary label a_t indicating whether it aligns with some property in the input set. Our auxiliary task is to predict a_t given the sequence of previously predicted words and input X:

$$P(a_{t+1}|y_{1:t},X) = sigmoid(g'(\mathbf{h}_t,c_t)) \quad (12)$$

$$g'(\mathbf{h}_t,c_t) = \mathbf{v}_a \cdot tanh(\mathbf{W}_c[c_t;\mathbf{h}_t]) \quad (13)$$

where $\mathbf{v}_a \in \mathbb{R}^d$ and the other operands are as defined in Equation (7). We optimise the following auxiliary objective function:

$$\mathcal{L}_{aln} = -\sum_{t=1}^{|Y|} log\, P(a_t|y_{1:t-1},X) \quad (14)$$

and the combined multi-task objective is the weighted sum of both word prediction and alignment prediction losses:

$$\mathcal{L}_{MTL} = \lambda\, \mathcal{L}_{wNLL} + (1-\lambda)\, \mathcal{L}_{aln} \quad (15)$$

where λ controls how much model training will focus on each task. As we will explain in Section 5, we can anneal this value during training in favour of one objective or the other.

4.3 Reinforcement Learning Training

Although the multi-task approach aims to smooth the target distribution, the training process is still driven by the imperfect target text. In other words, at each time step t the algorithm feeds the previous word w_{t-1} of the target text and evaluates the prediction against the target w_t.

Alternatively, we propose a training approach based on reinforcement learning (Williams 1992) which allows us to define an objective function that does not fully rely on the target text but rather on a revised version of it. In our case, the set of alignments obtained by our content selection model provides a revision for the target text. The advantages of reinforcement learning are twofold: (a) it allows to exploit additional task-specific knowledge (Zhang and Lapata, 2017) during training, and (b) enables the exploration of other word sequences through sampling. Our setting differs from previous applications of RL (Ranzato et al., 2016; Zhang and Lapata, 2017) in that the reward function is not computed on the target text but rather on its alignments with the input.

The encoder-decoder model is viewed as an agent whose action space is defined by the set of words in the target vocabulary. At each time step, the encoder-decoder takes action $\hat{y}_t$ with policy $P_\pi(\hat{y}_t|\hat{y}_{1:t-1},X)$ defined by the probability in Equation (6). The agent terminates when it emits the End Of Sequence (EOS) token, at which point the sequence of all actions taken yields the output sequence $\hat{Y} = (\hat{y}_1,\cdots,\hat{y}_{|\hat{Y}|})$. This sequence in our task is a short text describing the properties of a given entity. After producing the sequence of actions $\hat{Y}$, the agent receives a reward $r(\hat{Y})$ and the policy is updated according to this reward.

Reward Function We define the reward function $r(\hat{Y})$ on the alignment set $\mathcal{A}(X,Y)$. If the output action sequence $\hat{Y}$ is precise with respect to the set of alignments $\mathcal{A}(X,Y)$, the agent will receive a high reward. Concretely, we define $r(\hat{Y})$ as follows:

$$r(\hat{Y}) = \gamma^{pr}\, r^{pr}(\hat{Y}) \quad (16)$$

where γ^{pr} adjusts the reward value r^{pr} which is the unigram precision of the predicted sequence $\hat{Y}$ and the set of words in $\mathcal{A}(X,Y)$.

Training Algorithm We use the REINFORCE algorithm (Williams, 1992) to learn an agent that maximises the reward function. As this is a gradient descent method, the training loss of a sequence

is defined as the negative expected reward:

$$\mathcal{L}_{RL} = -\mathbb{E}_{(\hat{y}_1,\cdots,\hat{y}_{|\hat{Y}|})} \sim P_\pi(\cdot|X)[r(\hat{y}_1,\cdots,\hat{y}_{|\hat{Y}|})]$$

where P_π is the agent's policy, i.e., the word distribution produced by the encoder-decoder model (Equation (6)) and $r(\cdot)$ is the reward function as defined in Equation (16). The gradient of $\mathcal{L}_{RL}$ is given by:

$$\nabla \mathcal{L}_{RL} \approx \sum_{t=1}^{|\hat{Y}|} \nabla \log P_\pi(\hat{y}_t|\hat{y}_{1:t-1}, X)[r(\hat{y}_{1:|\hat{Y}|}) - b_t]$$

where b_t is a baseline linear regression model used to reduce the variance of the gradients during training. b_t predicts the future reward and is trained by minimizing mean squared error. The input to this predictor is the agent hidden state $\mathbf{h}_t$, however we do not back-propagate the error to $\mathbf{h}_t$. We refer the interested reader to Williams (1992) and Ranzato et al. (2016) for more details.

Document Level Curriculum Learning Rather than starting from a state given by a random policy, we initialise the agent with a policy learnt by pre-training with the negative log-likelihood objective (Ranzato et al., 2016; Zhang and Lapata, 2017). The reinforcement learning objective is applied gradually in combination with the log-likelihood objective on each target block subsequence. Recall from Section 4.1 that our document is segmented into blocks of equal size during training which we denote as MAXBLOCK. When training begins, only the last $\mho$ tokens are predicted by the agent while for the first $(\text{MAXBLOCK} - \mho)$ we still use the negative log-likelihood objective. The number of tokens $\mho$ predicted by the agent is incremented by $\mho$ units every 2 epochs. We set $\mho = 3$ and the training ends when $(\text{MAXBLOCK} - \mho) = 0$. Since we evaluate the model's predictions at the block level, the reward function is also evaluated at the block level.

5 Experimental Setup

Data We evaluated our model on a dataset collated from WIKIBIO (Lebret et al., 2016), a corpus of 728,321 biography articles (their first paragraph) and their infoboxes sampled from the English Wikipedia. We adapted the original dataset in three ways. Firstly, we make use of the entire abstract rather than first sentence. Secondly, we reduced the dataset to examples with a rich set of properties and multi-sentential text. We eliminated examples with less than six property-value pairs and abstracts consisting of one sentence. We also placed a minimum restriction of 23 words in the length of the abstract. We considered abstracts up to a maximum of 12 sentences and property sets with a maximum of 50 property-value pairs. Finally, we associated each abstract with the set of DBPedia properties $p:v$ corresponding to the abstract's main entity. As entity classification is available in DBPedia for most entities, we concatenate class information c (whenever available) with the property value, i.e., $p:vc$. In Figure 1, the property value $spouse:FrancesH.Flaherty$ is extended with class information from the DBPedia ontology to $spouse:FrancesH.FlahertyPerson$.

Pre-processing Numeric date formats were converted to a surface form with month names. Numerical expressions were delexicalised using different tokens created with the property name and position of the delexicalised token on the value sequence. For instance, given the property-value for *birth date* in Figure (1a), the first sentence in the abstract (Figure (1b)) becomes " *Robert Joseph Flaherty, (February DLX_birth_date_2, DLX_birth_date_4 – July ...* ". Years and numbers in the text not found in the values of the property set were replaced with tokens YEAR and NUMERIC.[2] In a second phase, when creating the input and output vocabularies, $\mathcal{V}^I$ and $\mathcal{V}^O$ respectively, we delexicalised words w which were absent from the output vocabulary but were attested in the input vocabulary. Again, we created tokens based on the property name and the position of the word in the value sequence. Words not in $\mathcal{V}^O$ or $\mathcal{V}^I$ were replaced with the symbol UNK. Vocabulary sizes were limited to $|\mathcal{V}^I| = 50k$ and $|\mathcal{V}^O| = 50k$ for the alignment model and $|\mathcal{V}^O| = 20k$ for the generator. We discarded examples where the text contained more than three UNKs (for the content aligner) and five UNKs (for the generator); or more than two UNKs in the property-value (for generation). Finally, we added the empty relation to the property sets.

Table 2 summarises the dataset statistics for the generator. We report the number of abstracts in the dataset (size), the average number of sentences and tokens in the abstracts, and the average number of properties and sentence length in tokens

[2] We exploit these tokens to further adjust the score of the reward function given by Equation (16). Each time the predicted output contains some of these symbols we decrease the reward score by κ which we empirically set to 0.025 .

generation	train	dev	test
size	165,324	25,399	23,162
sentences	3.51±1.99	3.46±1.94	3.22±1.72
tokens	74.13±43.72	72.85±42.54	66.81±38.16
properties	14.97±8.82	14.96±8.85	21.6±9.97
sent.len	21.06±8.87	21.03±8.85	20.77±8.74

Table 2: Dataset statistics.

(sent.len). For the content aligner (cf. Section 3), each sentence constitutes a training instance, and as a result the sizes of the train and development sets are 796,446 and 153,096, respectively.

Training Configuration We adjusted all models' hyperparameters according to their performance on the development set. The encoders for both content selection and generation models were initialised with GloVe (Pennington et al., 2014) pre-trained vectors. The input and hidden unit dimension was set to 200 for content selection and 100 for generation. In all models, we used encoder biLSTMs and decoder LSTM (regularised with a dropout rate of 0.3 (Zaremba et al., 2014)) with one layer. Content selection and generation models (base encoder-decoder and MTL) were trained for 20 epochs with the ADAM optimiser (Kingma and Ba, 2014) using a learning rate of 0.001. The reinforcement learning model was initialised with the base encoder-decoder model and trained for 35 additional epochs with stochastic gradient descent and a fixed learning rate of 0.001. Block sizes were set to 40 (base), 60 (MTL) and 50 (RL). Weights for the MTL objective were also tuned experimentally; we set $\lambda = 0.1$ for the first four epochs (training focuses on alignment prediction) and switched to $\lambda = 0.9$ for the remaining epochs.

Content Alignment We optimized content alignment on the development set against manual alignments. Specifically, two annotators aligned 132 sentences to their infoboxes. We used the Yawat annotation tool (Germann, 2008) and followed the alignment guidelines (and evaluation metrics) used in Cohn et al. (2008). The inter-annotator agreement using macro-averaged f-score was 0.72 (we treated one annotator as the reference and the other one as hypothetical system output).

Alignment sets were extracted from the model's output (cf. Section 3) by optimizing the threshold $avg(sim) + a * std(sim)$ where sim denotes the similarity between the set of property values and words, and a is empirically set to 0.75; avg and std are the mean and standard deviation of sim scores across the development set. Each word was aligned to a property-value if their similarity exceeded a threshold of 0.22. Our best content alignment model (Content-Aligner) obtained an f-score of 0.36 on the development set.

We also compared our Content-Aligner against a baseline based on pre-trained word embeddings (EmbeddingsBL). For each pair $(\mathcal{P}, s)$ we computed the dot product between words in s and properties in $\mathcal{P}$ (properties were represented by the the averaged sum of their words' vectors). Words were aligned to property-values if their similarity exceeded a threshold of 0.4. EmbeddingsBL obtained an f-score of 0.057 against the manual alignments. Finally, we compared the performance of the Content-Aligner at the level of property set $\mathcal{P}$ and sentence s similarity by comparing the average ranking position of correct pairs among 14 distractors, namely rank@15. The Content-Aligner obtained a rank of 1.31, while the EmbeddingsBL model had a rank of 7.99 (lower is better).

6 Results

We compared the performance of an encoder-decoder model trained with the standard negative log-likelihood method (ED), against a model trained with multi-task learning (ED_{MTL}) and reinforcement learning (ED_{RL}). We also included a template baseline system (Templ) in our evaluation experiments.

The template generator used hand-written rules to realise property-value pairs. As an approximation for content selection, we obtained the 50 more frequent property names from the training set and manually defined content ordering rules with the following criteria. We ordered personal life properties (e.g., *birth_date* or *occupation*) based on their most common order of mention in the Wikipedia abstracts. Profession dependent properties (e.g., *position* or *genre*), were assigned an equal ordering but posterior to the personal properties. We manually lexicalised properties into single sentence templates to be concatenated to produce the final text. The template for the property *position* and example verbalisation for the property-value $position : defender$ of the entity *zanetti* are "[NAME] *played as* [POSITION]." and "*Zanetti played as defender.*" respectively.

Automatic Evaluation Table 3 shows the results of automatic evaluation using BLEU-4

Model	Abstract	RevAbs
Templ	5.47	6.43
ED	13.46	35.89
ED_{MTL}	**13.57**	**37.18**
ED_{RL}	12.97	35.74

Table 3: BLEU-4 results using the original Wikipedia abstract (Abstract) as reference and crowd-sourced revised abstracts (RevAbs) for template baseline (Templ), standard encoder-decoder model (ED), and our content-based models trained with multi-task learning (ED_{MTL}) and reinforcement learning (ED_{RL}).

(Papineni et al., 2002) against the noisy Wikipedia abstracts. Considering these as a gold standard is, however, not entirely satisfactory for two reasons. Firstly, our models generate considerably shorter text and will be penalized for not generating text they were not supposed to generate in the first place. Secondly, the model might try to reproduce what is in the imperfect reference but not supported by the input properties and as a result will be rewarded when it should not. To alleviate this, we crowd-sourced using AMT a revised version of 200 randomly selected abstracts from the test set.

Crowdworkers were shown a Wikipedia infobox with the accompanying abstract and were asked to adjust the text to the content present in the infobox. Annotators were instructed to delete spans which did not have supporting facts and rewrite the remaining parts into a well-formed text. We collected three revised versions for each abstract. Inter-annotator agreement was 81.64 measured as the mean pairwise BLEU-4 amongst AMT workers.

Automatic evaluation results against the revised abstracts are also shown in Table 3. As can be seen, all encoder-decoder based models have a significant advantage over Templ when evaluating against both types of abstracts. The model enabled with the multi-task learning content selection mechanism brings an improvement of 1.29 BLEU-4 over a vanilla encoder-decoder model. Performance of the RL trained model is inferior and close to the ED model. We discuss the reasons for this discrepancy shortly.

To provide a rough comparison with the results reported in Lebret et al. (2016), we also computed BLEU-4 on the first sentence of the text generated by our system.[3] Recall that their model generates the first sentence of the abstract, whereas we output multi-sentence text. Using the first sentence in the Wikipedia abstract as reference, we obtained a score of 37.29% (ED), 38.42% (ED_{MTL}) and 38.1% (ED_{RL}) which compare favourably with their best performing model (34.7%±0.36).

[3]We post-processed system output with Stanford CoreNLP (Manning et al., 2014) to extract the first sentence.

System	1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}	Rank
Templ	12.17	14.33	10.17	15.50	**47.83**	3.72
ED	12.83	24.17	24.67	**25.17**	13.17	3.02
ED_{MTL}	14.83	**26.17**	**26.17**	19.17	13.67	2.90
ED_{RL}	14.67	**25.00**	**25.50**	24.00	10.83	2.91
RevAbs	**47.00**	14.00	12.67	16.17	9.17	2.27

Table 4: Rankings shown as proportions and mean ranks given to systems by human subjects.

Human-Based Evaluation We further examined differences among systems in a human-based evaluation study. Using AMT, we elicited 3 judgements for the same 200 infobox-abstract pairs we used in the abstract revision study. We compared the output of the templates, the three neural generators and also included one of the human edited abstracts as a gold standard (reference). For each test case, we showed crowdworkers the Wikipedia infobox and five short texts in random order. The annotators were asked to rank each of the texts according to the following criteria: (1) Is the text faithful to the content of the table? and (2) Is the text overall comprehensible and fluent? Ties were allowed only when texts were identical strings. Table 5 presents examples of the texts (and properties) crowdworkers saw.

Table 4 shows, proportionally, how often crowdworkers ranked each system, first, second, and so on. Unsurprisingly, the human authored gold text is considered best (and ranked first 47% of the time). ED_{MTL} is mostly ranked second and third best, followed closely by ED_{RL}. The vanilla encoder-decoder system ED is mostly forth and Templ is fifth. As shown in the last column of the table (Rank), the ranking of ED_{MTL} is overall slightly better than ED_{RL}. We further converted the ranks to ratings on a scale of 1 to 5 (assigning ratings 5…1 to rank placements 1…5). This allowed us to perform Analysis of Variance (ANOVA) which revealed a reliable effect of system type. Post-hoc Tukey tests showed that all systems were significantly worse than RevAbs and significantly better than Templ ($p < 0.05$). ED_{MTL} is not significantly better than ED_{RL} but is significantly ($p < 0.05$) different from ED.

Discussion The texts generated by ED_{RL} are shorter compared to the other two neural systems

property-set	**name**= dorsey burnette, **date**= may 2012, **bot**= blevintron bot, **background**= solo singer, **birth**= december 28 , 1932, **birth place**= memphis, tennessee, **death place**= {los angeles; canoga park, california}, **death**= august 19 , 1979, **associated acts**= the rock and roll trio, **hometown**= memphis, tennessee, **genre**= {rock and roll; rockabilly; country music}, **occupation**= {composer; singer}, **instruments**= {rockabilly bass; vocals; acoustic guitar}, **record labels**= {era records; coral records; smash records; imperial records; capitol records; dot records; reprise records}
RevAbs	*Dorsey Burnette (December 28 , 1932 – August 19 , 1979) was an american early Rockabilly singer. He was a member of the Rock and Roll Trio.*
Templ	*Dorsey Burnette (DB) was born in December 28 , 1932. DB was born in Memphis, Tennessee. DB died in August 19 , 1979. DB died in August 19 , 1979. DB died in Canoga Park, California. DB died in los angeles. DB was a composer. DB was a singer. DB 's genre was Rock and Roll. The background of DB was solo singer. DB 's genre was Rockabilly. DB worked with the Rock and Roll Trio. DB 's genre was Country music. DB worked with the Rock and Roll Trio.*
ED	*Dorsey Burnette (December 28 , 1932 – August 19 , 1979) was an american singer and songwriter. He was a member of the Rock band the band from YEAR to YEAR.*
ED_{MTL}	*Dorothy Burnette (December 28 , 1932 – August 19 , 1979) was an american country music singer and songwriter. He was a member of the Rock band Roll.*
ED_{RL}	*Burnette Burnette (December 28 , 1932 – August 19 , 1979) was an american singer and songwriter. He was born in memphis , Tennessee.*
property-set	**name**= indrani bose, **doctoral advisor**= chanchal kumar majumdar, **alma mater**= university of calcutta, **birth**= 1951-0-0, **birth place**= kolkata, **field**= theoretical physics, **work institution**= bose institute, **birth**= august 15 , 1951, **honours**= fna sc, **nationality**= india, **known for**= first recipient of stree sakthi science samman award
RevAbs	*Indrani Bose (born 1951) is an indian physicist at the Bose institute. Professor Bose obtained her ph.d. from University of Calcutta*
Templ	*Indrani Bose (IB) was born in year-0-0. IB was born in August 15 , 1951. IB was born in kolkata. IB was a india. IB studied at University of Calcutta. IB was known for First recipient of Stree Sakthi Science Samman Award.*
ED	*Indrani UNK (born 15 August 1951) is an indian Theoretical physicist and Theoretical physicist. She is the founder and ceo of UNK UNK.*
ED_{MTL}	*Indrani Bose (born 15 August 1951) is an indian Theoretical physicist. She is a member of the UNK Institute of Science and technology.*
ED_{RL}	*Indrani UNK (born 15 August 1951) is an indian Theoretical physicist. She is a member of the Institute of technology (UNK).*
property-set	**name**= aaron moores, **coach**= sarah paton, **club**= trowbridge asc, **birth**= may 16 , 1994, **birth place**= trowbridge, **sport**= swimming, **paralympics**= 2012
RevAbs	*Aaron Moores (born 16 May 1994) is a british ParalyMpic swiMMer coMpeting in the s14 category , Mainly in the backstroke and breaststroke and after qualifying for the 2012 SuMMer ParalyMpics he won a Silver Medal in the 100 M backstroke.*
Templ	*Aaron Moores (AM) was born in May 16 , 1994. AM was born in May 16 , 1994. AM was born in Trowbridge.*
ED	*Donald Moores (born 16 May 1994) is a Paralympic swimmer from the United states. He has competed in the Paralympic Games.*
ED_{MTL}	*Donald Moores (born 16 May 1994) is an english swimmer. He competed at the 2012 Summer Paralympics.*
ED_{RL}	*Donald Moores (born 16 May 1994) is a Paralympic swimmer from the United states. He competed at the dlx_updated_3 Summer Paralympics.*
property-set	**name**= kirill moryganov, **height**= 183.0, **birth**= february 7 , 1991, **position**= defender, **height**= 1.83, **goals**= {0; 1}, **clubs**= fc torpedo moscow, **pcupdate**= may 28 , 2016, **years**= {2013; 2012; 2015; 2016; 2010; 2014; 2008; 2009}, **team**= {fc neftekhimik nizhnekamsk; fc znamya truda orekhovo- zuyevo; fc irtysh omsk; fc vologda; fc torpedo-zil moscow; fc tekstilshchik ivanovo; fc khimki; fc oktan perm, fc ryazan, fc amkar perm}, **matches**= {16; 10; 3; 4; 9; 0; 30; 7; 15}
RevAbs	*Kirill Andreyevich Moryganov (; born 7 February 1991) is a russian professional football player. He plays for fc Irtysh Omsk. He is a Central defender.*
Templ	*Kirill Moryganov (KM) was born in February 7 , 1991. KM was born in February 7 , 1991. The years of KM was 2013. The years of KM was 2013. KM played for fc Neftekhimik Nizhnekamsk. KM played for fc Znamya Truda Orekhovo- zuyevo. KM scored 1 goals. The years of KM was 2013. KM played for fc Irtysh Omsk. The years of KM was 2013. KM played as Defender. KM played for fc Vologda. KM played for fc Torpedo-zil Moscow. KM played for fc Tekstilshchik Ivanovo. KM scored 1 goals. KM 's Club was fc Torpedo Moscow. KM played for fc Khimki. The years of KM was 2013. The years of KM was 2013. The years of KM was 2013. KM played for fc Amkar Perm. The years of KM was 2013. KM played for fc Ryazan. KM played for fc Oktan Perm.*
ED	*Kirill mikhailovich Moryganov (; born February 7 , 1991) is a russian professional football player. He last played for fc Torpedo armavir.*
ED_{MTL}	*Kirill Moryganov (; born 7 February 1991) is an english professional footballer who plays as a Defender. He plays for fc Neftekhimik Nizhnekamsk.*
ED_{RL}	*Kirill viktorovich Moryganov (; born February 7 , 1991) is a russian professional football player. He last played for fc Tekstilshchik Ivanovo.*

Table 5: Examples of system output.

which might affect BLEU-4 scores and also the ratings provided by the annotators. As shown in Table 5 (entity *dorsey burnette*), ED_{RL} drops information pertaining to dates or chooses to just verbalise birth place information. In some cases, this is preferable to hallucinating incorrect facts; however, in other cases outputs with more information are rated more favourably. Overall, ED_{MTL} seems to be more detail oriented and faithful to the facts included in the infobox (see *dorsey burnette*, *aaron moores*, or *kirill moryganov*). The template system manages in some specific configurations to verbalise appropriate facts (*indrani bose*), however, it often fails to verbalise infrequent properties (*aaron moores*) or focuses on properties which are very frequent in the knowledge base but are rarely found in the abstracts (*kirill moryganov*).

7 Conclusions

In this paper we focused on the task of bootstrapping generators from large-scale datasets consisting of DBPedia facts and related Wikipedia biography abstracts. We proposed to equip standard encoder-decoder models with an additional content selection mechanism based on multi-instance learning and developed two training regimes, one based on multi-task learning and the other on reinforcement learning. Overall, we find that the proposed content selection mechanism improves the accuracy and fluency of the generated texts. In the future, it would be interesting to investigate a more sophisticated representation of the input (Vinyals et al., 2016). It would also make sense for the model to decode hierarchically, taking sequences of words *and* sentences into account (Zhang and Lapata, 2014; Lebret et al., 2015).

Acknowledgments

We thank the NAACL reviewers for their constructive feedback. We also thank Xingxing Zhang, Li Dong and Stefanos Angelidis for useful discussions about implementation details. We gratefully acknowledge the financial support of the European Research Council (award number 681760).

References

Stuart Andrews and Thomas Hofmann. 2004. Multiple instance learning via disjunctive programming boosting. In *Advances in Neural Information Processing Systems 16*, Curran Associates, Inc., pages 65–72.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBPedia: A nucleus for a web of open data. *The Semantic Web* pages 722–735.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Represetnations*. San Diego, CA.

Regina Barzilay and Mirella Lapata. 2005. Collective content selection for concept-to-text generation. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. Vancouver, British Columbia, Canada, pages 331–338.

Anja Belz and Eric Kow. 2010. Extracting parallel fragments from comparable corpora for data-to-text generation. In *Proceedings of the 6th International Natural Language Generation Conference*. Association for Computational Linguistics, Ireland, pages 167–171.

Johannes Bjerva, Barbara Plank, and Johan Bos. 2016. Semantic tagging with deep residual networks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan, pages 3531–3541.

Peter Carbonetto, Gyuri Dorkó, Cordelia Schmid, Hendrik Kück, and Nando De Freitas. 2008. Learning to recognize objects with little supervision. *International Journal of Computer Vision* 77(1):219–237.

Richard Caruana. 1993. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the 10th International Conference on Machine Learning*. Morgan Kaufmann, pages 41–48.

Hao Cheng, Hao Fang, and Mari Ostendorf. 2015. Open-domain name error detection using a multitask RNN. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal, pages 737–746.

Andrew Chisholm, Will Radford, and Ben Hachey. 2017. Learning to generate one-sentence biographies from Wikidata. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain, pages 633–642.

Trevor Cohn, Chris Callison-Burch, and Mirella Lapata. 2008. Constructing corpora for the development and evaluation of paraphrase systems. *Computational Linguistics* 34(4):597–614.

Ronan Ronan Collobert, Jason Weston, Michael Karlen Léon Bottou, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(Aug):2493–2537.

Timothee Cour, Ben Sapp, and Ben Taskar. 2011. Learning from partial labels. *Journal of Machine Learning Research* 12(May):1501–1536.

Ulrich Germann. 2008. Yawat: yet another word alignment tool. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Demo Session*. Columbus, Ohio, pages 20–23.

Hamidreza Ghader and Christof Monz. 2017. What does attention in neural machine translation pay attention to? In *Proceedings of the 8th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Asian Federation of Natural Language Processing, Taipei, Taiwan, pages 30–39.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Portland, Oregon, USA, pages 541–550.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 3128–3137.

Jim Keeler and David E Rumelhart. 1992. A self-organizing integrated segmentation and recognition neural net. In *Advances in Neural Information Processing Systems 5*. Curran Associates, Inc., pages 496–503.

Joohyun Kim and Raymond J. Mooney. 2010. Generative alignment and semantic parsing for learning from ambiguous supervision. In *Proceedings of the 23rd International Conference on Computational Linguistics*. Beijing, China, pages 543–551.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas, pages 1203–1213.

Rémi Lebret, Pedro O Pinheiro, and Ronan Collobert. 2015. Phrase-based image captioning. *arXiv preprint arXiv:1502.03671* .

Percy Liang, Michael Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Suntec, Singapore, pages 91–99.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2017. Table-to-text generation by structure-aware seq2seq learning. *arXiv preprint arXiv:1711.09724* .

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *Proceedings of the International Conference on Learning Representations*. San Juan, Puerto Rico.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal, pages 1412–1421.

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

Oded Maron and Aparna Lakshmi Ratan. 1998. Multiple-instance learning for natural scene classification. In *Proceedings of the 15th International Conference on Machine Learning*. San Francisco, California, USA, volume 98, pages 341–349.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using LSTMs with coarse-to-fine alignment. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California, pages 720–730.

Yassine Mrabet, Pavlos Vougiouklis, Halil Kilicoglu, Claire Gardent, Dina Demner-Fushman, Jonathon Hare, and Elena Simperl. 2016. *Proceedings of the 2nd International Workshop on Natural Language Generation and the Semantic Web*, Association for Computational Linguistics, chapter Aligning Texts and Knowledge Bases with Semantic Sentence Simplification, pages 29–36.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA, pages 311–318.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Doha, Qatar, pages 1532–1543.

Laura Perez-Beltrachini and Claire Gardent. 2016. Learning embeddings to lexicalise RDF properties. In *Proceedings of the 5th Joint Conference on Lexical and Computational Semantics*. Berlin, Germany, pages 219–228.

Barbara Plank. 2016. Keystroke dynamics as signal for shallow syntactic parsing. In *Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan, pages 609–619.

Marc'Aurelio Ranzato, Summit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *Proceedings of the International Conference on Learning Representations*. San Juan, Puerto Rico.

Lei Sha, Lili Mou, Tianyu Liu, Pascal Poupart, Sujian Li, Baobao Chang, and Zhifang Sui. 2017. Order-planning neural text generation from structured data. *arXiv preprint arXiv:1709.00155* .

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

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., pages 3104–3112.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2016. Order matters: Sequence to sequence for sets. In *Proceedings of the International Conference on Learning Representations*. San Juan, Puerto Rico.

Sebastian Walter, Christina Unger, and Philipp Cimiano. 2013. A corpus-based approach for the induction of ontology lexica. In *International Conference on Application of Natural Language to Information Systems*. Springer, pages 102–113.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark, pages 2243–2253.

Jiajun Wu, Yinan Yu, Chang Huang, and Kai Yu. 2015. Deep multiple instance learning for image classification and auto-annotation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern*

Recognition. Boston, Massachusetts, USA, pages 3460–3469.

Wei Xu, Alan Ritter, Chris Callison-Burch, William B Dolan, and Yangfeng Ji. 2014. Extracting lexically divergent paraphrases from Twitter. *Transactions of the Association for Computational Linguistics* 2:435–448.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *CoRR* abs/1409.2329.

Qi Zhang, Sally A Goldman, Wei Yu, and Jason E Fritts. 2002. Content-based image retrieval using multiple-instance learning. In *Proceedings of the 19th International Conference on Machine Learning*. Sydney, Australia, volume 2, pages 682–689.

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Doha, Qatar, pages 670–680.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark, pages 595–605.

SHAPED: Shared-Private Encoder-Decoder for Text Style Adaptation

Ye Zhang*
UT Austin
yezhang@utexas.edu

Nan Ding **Radu Soricut**
Google AI
{dingnan,rsoricut}@google.com

Abstract

Supervised training of abstractive language generation models results in learning conditional probabilities over language sequences based on the supervised training signal. When the training signal contains a variety of writing styles, such models may end up learning an 'average' style that is directly influenced by the training data make-up and cannot be controlled by the needs of an application. We describe a family of model architectures capable of capturing both generic language characteristics via shared model parameters, as well as particular style characteristics via private model parameters. Such models are able to generate language according to a specific learned style, while still taking advantage of their power to model generic language phenomena. Furthermore, we describe an extension that uses a mixture of output distributions from all learned styles to perform on-the-fly style adaptation based on the textual input alone. Experimentally, we find that the proposed models consistently outperform models that encapsulate single-style or average-style language generation capabilities.

1 Introduction

Encoder-decoder models have recently pushed forward the state-of-the-art performance on a variety of language generation tasks, including machine translation (Bahdanau et al., 2015; Wu et al., 2016; Vaswani et al., 2017), text summarization (Rush et al., 2015; Nallapati et al., 2016; See et al., 2017), dialog systems (Li et al., 2016; Asghar et al., 2017), and image captioning (Xu et al., 2015; Ranzato et al., 2015; Liu et al., 2017). This framework consists of an encoder that reads the input data and encodes it as a sequence of vectors, which is in turn used by a decoder to generate another sequence of vectors used to produce output symbols step by step.

The prevalent approach to training such a model is to update all the model parameters using all the examples in the training data (over multiple epochs). This is a reasonable approach, under the assumption that we are modeling a single underlying distribution in the data. However, in many applications and for many natural language datasets, there exist multiple underlying distributions, characterizing a variety of language styles. For instance, the widely-used Gigaword dataset (Graff and Cieri, 2003) consists of a collection of articles written by various publishers (The New York Times, Agence France Presse, Xinhua News, etc.), each with its own style characteristics. Training a model's parameters on all the training examples results in an averaging effect across style characteristics, which may lower the quality of the outputs; additionally, this averaging effect may be completely undesirable for applications that require a level of control over the output style. At the opposite end of the spectrum, one can choose to train one independent model per each underlying distribution (assuming we have the appropriate signals for identifying them at training time). This approach misses the opportunity to exploit common properties shared by these distributions (e.g., generic characteristics of a language, such as noun-adjective position), and leads to models that are under-trained due to limited data availability per distribution.

In order to address these issues, we propose a novel neural architecture called SHAPED (shared-private encoder-decoder). This architecture has both shared encoder/decoder parameters that are updated based on all the training examples, as well as private encoder/decoder parameters that are updated using only examples from their corresponding underlying training distributions. In addition

*Work done as an intern at Google AI.

Proceedings of NAACL-HLT 2018, pages 1528–1538
New Orleans, Louisiana, June 1 - 6, 2018.

to learning different parametrization between the shared model and the private models, we jointly learn a classifier to estimate the probability of each example belonging to each of the underlying training distributions. In such a setting, the shared parameters ('shared model') are expected to learn characteristics shared by the entire set of training examples (i.e., language generic), whereas each private parameter set ('private model') learns particular characteristics (i.e., style specific) of their corresponding training distribution. At the same time, the classifier is expected to learn a probability distribution over the labels used to identify the underlying distributions present in the input data. At test time, there are two possible scenarios. In the first one, the input signal explicitly contains information about the underlying distribution (e.g., the publisher's identity). In this case, we feed the data into the shared model and also the corresponding private model, and perform sequence generation based on a concatenation of their vector outputs; we refer to this model as the SHAPED model. In a second scenario, the information about the underlying distribution is either not available, or it refers to a distribution that was not seen during training. In this case, we feed the data into the shared model and all the private models; the output distribution of the symbols of the decoding sequence is estimated using a mixture of distributions from all the decoders, weighted according to the classifier's estimates for that particular example; we refer to this model as the Mix-SHAPED model.

We test our models on the headline-generation task based on the aforementioned Gigaword dataset. When the publisher's identity is presented as part of the input, we show that the SHAPED model significantly surpasses the performance of the shared encoder-decoder baseline, as well as the performance of private models (where one individual, per-publisher model is trained for each in-domain style). When the publisher's identity is not presented as part of the input (i.e., not presented at run-time but revealed at evaluation-time for measurement purposes), we show that the Mix-SHAPED model exhibits a high level of classification accuracy based on textual inputs alone (accuracy percentage in the 80s overall, varying by individual publisher), while its generation accuracy still surpasses the performance of the baseline models. Finally, when the publisher's identity is unknown to the model (i.e., a publisher that was not part of the training dataset), we show that the Mix-SHAPED model performance far surpasses the shared model performance, due to the ability of the Mix-SHAPED model to perform on-the-fly adaptation of output style. This feat comes from our model's ability to perform two distinct tasks: match the incoming, previously-unseen input style to existing styles learned at training time, and use the correlations learned at training time between input and output style characteristics to generate style-appropriate token sequences.

2 Related Work

Encoder-Decoder Models for Structured Output Prediction

Encoder-decoder architectures have been successfully applied to a variety of structure prediction tasks recently. Tasks for which such architectures have achieved state-of-the-art results include machine translation (Bahdanau et al., 2015; Wu et al., 2016; Vaswani et al., 2017), automatic text summarization (Rush et al., 2015; Chopra et al., 2016; Nallapati et al., 2016; Paulus et al., 2017; Nema et al., 2017), sentence simplification (Filippova et al., 2015; Zhang and Lapata, 2017), dialog systems (Li et al., 2016, 2017; Asghar et al., 2017), image captioning (Vinyals et al., 2015; Xu et al., 2015; Ranzato et al., 2015; Liu et al., 2017), etc. By far the most used implementation of such architectures is based on the original sequence-to-sequence model (Sutskever et al., 2014), augmented with its attention-based extension (Bahdanau et al., 2015). Although our SHAPED and Mix-SHAPED model formulations do not depend on a particular architecture implementation, we do make use of the (Bahdanau et al., 2015) model to instantiate our models.

Domain Adaptation for Neural Network Models

One general approach to domain adaptation for natural language tasks is to perform data/feature augmentation that represents inputs as both general and domain-dependent data, as originally proposed in (Daumé III, 2009), and ported to neural models in (Kim et al., 2016). For computer vision tasks, a line of work related to our approach has been proposed by Bousmalis et al. (2016) using what they call domain separation networks. As a tool for studying unsupervised domain adaptation

for image recognition tasks, their proposal uses CNNs for encoding an image into a feature representation, and also for reconstructing the input sample. It also makes use of a private encoder for each domain, and a shared encoder for both the source and the target domain. The approach we take in this paper shares this idea of model parametrization according to the domain/style, but goes further with the Mix-SHAPED model, performing on-the-fly adaptation of the model outputs. Other CNN-based domain adaptation methods for object recognition tasks are presented in (Long et al., 2016; Chopra et al., 2013; Tzeng et al., 2015; Sener et al., 2016).

For NLP tasks, Peng and Dredze (2017) take a multi-task approach to domain adaptation and sequence tagging. They use a shared encoder to represent instances from all of the domains, and use a domain projection layer to project the shared layer into a domain-specific space. They only consider the supervised domain-adaptation case, in which labeled training data exists for the target domain. Glorot et al. (2011) use auto-encoders for learning a high-level feature extraction across domains for sentiment analysis, while Zhou et al. (2016) employ auto-encoders to directly transfer the examples across different domains also for the same sentiment analysis task. Hua and Wang (2017) perform an experimental analysis on domain adaptation for neural abstractive summarization.

An important requirement of all the methods in the related work described above is that they require access to the (unlabeled) target domain data, in order to learn a domain-invariant representation across source and target domains. In contrast, our Mix-SHAPED model does not need access to a target domain or style at training time, and instead performs the adaptation on-the-fly, according to the specifics of the input data and the correlations learned at training time between available input and output style characteristics. As such, it is a more general approach, which allows adaptation for a much larger set of target styles, under the weaker assumption that there exists one or more styles present in the training data that can act as representative underlying distributions.

3 Model Architecture

Generally speaking, a standard encoder-decoder model has two components: an encoder that takes as input a sequence of symbols $\mathbf{x} = (x_1, x_2, ..., x_{T_x})$ and encodes them into a set of vectors $\mathbf{H} = (h_1, h_2, ..., h_{T_x})$,

$$\mathbf{H} = f_{\text{enc}}(\mathbf{x}), \tag{1}$$

where f_{enc} is the computation unit in the encoder; and, a decoder that generates output symbols at each time stamp t, conditioned on $\mathbf{H}$ as well as the decoder inputs $\mathbf{y}_{1:t-1}$,

$$s_t = f_{\text{dec}}(\mathbf{y}_{1:t-1}, \mathbf{H}), \tag{2}$$

where f_{dec} is the computation unit in the decoder. Instantiations of this framework include the widely-used attention-based sequence-to-sequence model (Bahdanau et al., 2015), in which f_{enc} and f_{dec} are implemented by an RNN architecture using LSTM (Hochreiter and Schmidhuber, 1997) or GRU (Chung et al., 2014) units. A more recent instantiation of this architecture is the Transformer model (Vaswani et al., 2017), built using self-attention layers.

3.1 SHAPED: Shared-private encoder-decoder

The abstract encoder-decoder model described above is usually trained over all examples in the training data. We call such a model a *shared* encoder-decoder model, because the model parameters are shared across all training and test instances. Formally, the shared encoder-decoder consists of the computation units f^s_{enc} and f^s_{dec}. Given an instance $\mathbf{x}$, it generates a sequence of vectors $\mathbf{S}^s = (s^s_1, ...s^s_T)$ by:

$$\mathbf{H}^s = f^s_{\text{enc}}(\mathbf{x}), s^s_t = f^s_{\text{dec}}(\mathbf{y}_{1:t-1}, \mathbf{H}^s). \tag{3}$$

The drawback of the shared encoder-decoder is that it fails to account for particular properties of each style that may be present in the data. In order to capture such particular style characteristics, a straightforward solution is to train a *private* model for each style. Assuming a style set $\mathbb{D} = \{D_1, D_2..., D_{|\mathbb{D}|}\}$, such a solution implies that each style has its own private encoder computation unit and decoder computation unit. At both training and testing time, each private encoder and decoder only process instances that belong to their own style. Given an instance along with its style $(\mathbf{x}, z)$ where $z \in \{1, \ldots, |\mathbb{D}|\}$, the private encoder-decoder generates a sequence of vectors $\mathbf{S}^z = (s^z_1, ...s^z_T)$ by:

$$\mathbf{H}^z = f^z_{\text{enc}}(\mathbf{x}), s^z_t = f^z_{\text{dec}}(\mathbf{y}_{1:t-1}, \mathbf{H}^z). \tag{4}$$

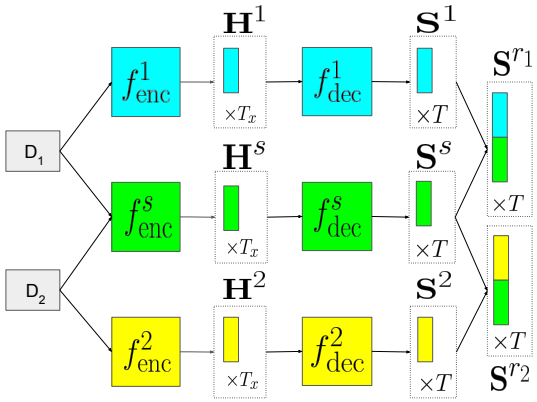


Figure 1: Illustration of the SHAPED model using two styles D_1 and D_2. D_1 articles pass through the private encoder f^1_{enc} and decoder f^1_{dec}. D_2 articles pass through the private encoder f^2_{enc} and decoder f^2_{dec}. Both of them also go through the shared encoder f^s_{dec} and decoder f^s_{dec}.

Although the private encoder/decoder models do preserve style characteristics, they fail to take into account the common language features shared across styles. Furthermore, since each style is represented by a subset of the entire training set, such private models may end up as under-trained, due to limited number of available data examples.

In order to efficiently capture both common and unique features of data with different styles, we propose the SHAPED model. In the SHAPED model, each data-point goes through both the shared encoder-decoder and its corresponding private encoder-decoder. At each step of the decoder, the output from private and shared ones are concatenated to form a new vector:

$$s^{r_z}_t = [s^z_t, s^s_t], \tag{5}$$

that contains both private features for style z and shared features induced from other styles, as illustrated in Fig 1. The output symbol distribution over tokens $o_t \in V$ (where V is the output vocabulary) at step t is given by:

$$p(o_t|\mathbf{x}, y_{1:t-1}, z) = \text{Softmax}(g(s^{r_z}_t)), \tag{6}$$

where g is a multi-layer feed-forward network that maps $s^{r_z}_t$ to a vector of size $|V|$. Given N training examples $(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}, z^{(1)}), \ldots, (\mathbf{x}^{(N)}, \mathbf{y}^{(N)}, z^{(N)})$, the conditional probability of the output $\mathbf{y}^{(i)}$ given article $\mathbf{x}^{(i)}$ and its style $z^{(i)} \in \{1, \ldots, |\mathbb{D}|\}$ is:

$$p(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}, z^{(i)}) = \prod_t p(o_t = y^{(i)}_t|\mathbf{x}^{(i)}, \mathbf{y}^{(i)}_{1:t-1}, z^{(i)}). \tag{7}$$

At inference time, given an article $\mathbf{x}$ with style z, we feed $\mathbf{x}$ into $f^s_{\text{enc}}, f^s_{\text{dec}}, f^z_{\text{enc}}, f^z_{\text{dec}}$ (Eq. 3-4) and obtain symbol distributions at each step t using Eq. 6. We sample from the distribution and obtain a symbol o_t which will be used as the estimated y_t and fed to the next steps.

3.2 The Mix-SHAPED Model

One limitation of the above model is that it can only handle test data containing an explicit style label from $\mathbb{D} = \{D_1, D_2..., D_{|\mathbb{D}|}\}$. However, there is frequently the case that, at test time, the style label is not present as part of the input, or that the input style is not part of the modeled set $\mathbb{D}$.

We treat both of these cases similarly, as a case of modeling an unknown style. We first describe our treatment of such a case at run-time. We use a latent random variable $z \in \{1, \ldots, |\mathbb{D}|\}$ to denote the underlying style of a given input. When generating a token at step t, the output token distribution takes the form of a mixture of SHAPED (Mix-SHAPED) model outputs:

$$p(o_t|\mathbf{x}, \mathbf{y}_{1:t-1}) = \sum_{d=1}^{|\mathbb{D}|} p(o_t|\mathbf{x}, \mathbf{y}_{1:t-1}, z = d)p(z = d|\mathbf{x}), \tag{8}$$

where $p(o_t|\mathbf{x}, \mathbf{y}_{1:t-1}, z = d)$ is the output symbol distribution of SHAPED decoder d, evaluated as in Eq. 6. Fig. 2 contains an illustration of such a model. In this formulation, $p(z|\mathbf{x})$ denotes the style conditional probability distribution from a trainable style classifier.

The joint data likelihood of target sequence $\mathbf{y}$ and target domain label z for input sequence $\mathbf{x}$ is:

$$p(\mathbf{y}, z|\mathbf{x}) = p(\mathbf{y}|z, \mathbf{x}) \cdot p(z|\mathbf{x}) \tag{9}$$

Training the Mix-SHAPED model involves minimizing a loss function that combines the negative log-likelihood of the style labels and the negative log-likelihood of the symbol sequences (see the model in Fig 3):

$$\begin{aligned}\text{Loss}_{\text{Mix-SHAPED}} = &-\sum_{i=1}^{N} \log p(z^{(i)}|\mathbf{x}^{(i)}) \\ &-\sum_{i=1}^{N} \log p(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}, z^{(i)}).\end{aligned} \tag{10}$$

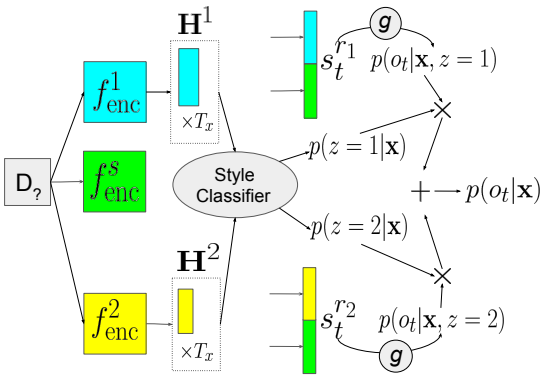


Figure 2: Decoding data with unknown style using a Mix-SHAPED model. The data is run through all encoders and decoders. The output of private encoders is fed into a classifier that estimates style distribution. The output symbol distribution is a mixture over all decoder outputs.

At run-time, if the style d of the input is available and $d \in \mathbb{D}$, we decode the sequence using Eq. 6. This also corresponds to the case $p(z = d|\mathbf{x}) = 1$ and 0 for all other styles, and reduces Eq. 8 to Eq. 6. If the style of the input is unknown (or known, but with $d' \notin \mathbb{D}$), we decode the sequence using Eq. 8, in which case the mixture over SHAPED models given by $p(z|\mathbf{x})$ is approximating the desired output style.

4 Model Instantiation

As an implementation of the encoder-decoder model, we use the attention-based sequence-to-sequence model from (Bahdanau et al., 2015), with an RNN architecture using GRU units (Chung et al., 2014). The input token sequences are first projected into an embedding space via an embedding matrix $\mathbf{E}$, resulting in a sequence of vectors as input representations.

The private and shared RNN cells generate a sequence of hidden state vectors $\mathbf{H}^z = \{h^z_j\}$, $z \in \{1, ..., |\mathbb{D}|\}$ and $\mathbf{H}^s = \{h^s_j\}$, for $j \in \{1, ..., T_x\}$. At each step in the encoder, h^z_j and h^s_j are concatenated to form a new output vector $h^{r_z}_j = [h^z_j, h^s_j]$. The final state of each encoder is used as the initial state of the corresponding decoder. At time step t in the decoder, the private and shared RNN cell first generate hidden state vectors $\{s^z_t\}, z \in \{1, ..., |\mathbb{D}|\}$ and s^s_t, then s^s_t is concatenated with each s^z_t to form new vectors $\{s^{r_z}_t\}$ ($z \in \{1, ..., |\mathbb{D}|\}$).

We apply the attention mechanism on $s^{r_z}_t$, using

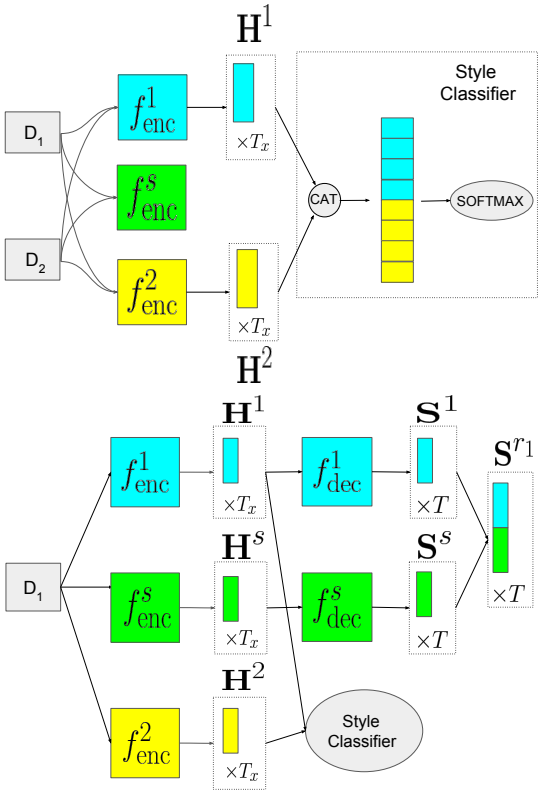


Figure 3: Training a Mix-SHAPED model. (a) Each example is fed to all private encoders f^1_{enc}, f^2_{enc}, whose outputs are concatenated and fed to a style classifier. (b) The D_1 examples only use $f^1_{enc}, f^1_{dec}, f^s_{enc}, f^s_{dec}$ to decode texts. Private encoder-decoders of other styles are not used.

attention weights calculated as:

$$q^{r_z}_{tj} = v_a \tanh(W_a h^{r_z}_j + U_a s^{r_z}_t), \quad (11)$$

which are normalized to a probability distribution:

$$\alpha^{r_z}_{tj} = \frac{\exp(q^{r_z}_{tj})}{\sum_{i=1}^{T_x} \exp(q^{r_z}_{ti})} \quad (12)$$

Context vectors are computed using normalized attention weights:

$$c^{r_z}_t = \sum_{j=1}^{T_x} \alpha^{r_z}_{tj} h^{r_z}_j \quad (13)$$

Given the context vector and the hidden state vectors, the symbol distribution at step t is:

$$p(o_t|\mathbf{x}, \mathbf{y}_{1:t}, z) = \text{softmax}(g([c^{r_z}_t, s^{r_z}_t])) \quad (14)$$

The attention weights in W_a, U_a, and v_a, as well as the embedding matrix $\mathbf{E}$ and vocabulary V are shared by all encoders and decoders. We use Eq. 14 to calculate the symbol loss in Eq. 10.

5 Quantitative Experiments

We perform a battery of quantitative experiments, designed to answer several main questions: 1) Do the proposed model improve generation performance over alternative approaches? 2) Can a style classifier built using an auxiliary loss provide a reliable estimate on text style? 3) In the case of unknown style, does the Mix-SHAPED model improve generation performance over alternative approaches? 4) To what extent do our models capture style characteristics as opposed to, say, content characteristics?

We perform our experiments using text summarization as the main task. More precisely, we train and evaluate headline generation models using the publicly-available Gigaword dataset (Graff and Cieri, 2003; Napoles et al., 2012).

5.1 Headline-generation Setup

The Gigaword dataset contains news articles from seven publishers: Agence France-Presse (AFP), Associated Press Worldstream (APW), Central News Agency of Taiwan (CNA), Los Angeles Times/Washington Post Newswire Service (LTW), New York Times (NYT), Xinhua News Agency (XIN), and Washington Post/Bloomberg Newswire Service (WPB). We pre-process this dataset in the same way as in (Rush et al., 2015), which results in articles with average length 31.4 words, and headlines with average length 8.5 words.

We consider the publisher identity as a proxy for style, and choose to model as in-domain styles the set $\mathbb{D} = \{\text{AFP, APW, NYT, XIN}\}$, while holding out CNA and LTW for out-of-domain style testing. This results in a training set containing the following number of (article, headline) instances: 993,584 AFP, 1,493,758 APW, 578,259 NYT, and 946,322 XIN. For the test set, we sample a total number of 10,000 in-domain examples from the original Gigawords test dataset, which include 2,886 AFP, 2,832 APW, 1,610 NYT, and 2,012 XIN. For out-of-domain testing, we randomly sample 10,000 LTW and 10,000 CNA test data examples. We remove the WPB articles due to their small number of instances.

5.1.1 Experimental Setup

We compare the following models:

- A Shared encoder-decoder model (S) trained on all styles in $\mathbb{D}$;
- A suite of Private encoder-decoder models (P), each one trained on a particular style from $\mathbb{D} = \{\text{AFP, APW, NYT, XIN}\}$;[1]
- A SHAPED model (SP) trained on all styles in $\mathbb{D}$; at test time, the style of test data is provided to the model; the article is only run through its style-specific private network and shared network (style classifier is not needed);
- A Mix-SHAPED model (M-SP) trained on all styles in $\mathbb{D}$; at test time, the style of article is not provided to the model; the output is computed using the mixture model, with the estimated style probabilities from the style classifier used as weights.

When testing on the out-of-domain styles CNA/LTW, we only compare the Shared (S) model with the Mix-SHAPED (M-SP) model, as the others cannot properly handle this scenario.

As hyper-parameters for the model instantiation, we used 500-dimension word embeddings, and a three-layer, 500-dimension GRU-cell RNN architecture; the encoder was instantiated as a bi-directional RNN. The lengths of the input and output sequences were truncated to 40 and 20 tokens, respectively. All the models were optimized using Adagrad (Duchi et al., 2011), with an initial learning rate of 0.01. The training procedure was done over mini-batches of size 128, and the updates were done asynchronously across 40 workers for 5M steps. The encoder/decoder word embedding and the output projection matrices were tied to minimize the number of parameters. To avoid the slowness from the softmax operator over large vocabulary sizes, and also mitigate the impact of out-of-vocabulary tokens, we applied a subtokenization method (Wu et al., 2016), which invertibly transforms a native token into a sequence of subtokens from a limited vocabulary (here set to 32K).

Comparison with Previous Work In the next section, we report our main results using the in-domain and out-of-domain (w.r.t. the selected publisher styles) test sets described above, since these test sets have a balanced publisher style frequency that allows us to measure the impact of our style-adaptation models. However, we also report

[1] We also tried to warm-start a private model using the best checkpoint of the shared model, but found that it cannot improve over the shared model.

	AFP/APW/XIN/NYT Test		
	Rouge-1	Rouge-2	Rouge-L
P	39.14±0.47	19.74±0.48	36.42±0.46
S	39.32±0.26	19.63±0.24	36.51±0.26
SP	**40.34**±0.26	**20.38**±0.25	**37.52**±0.25
M-SP	40.10±0.25	20.21±0.26	37.30±0.26

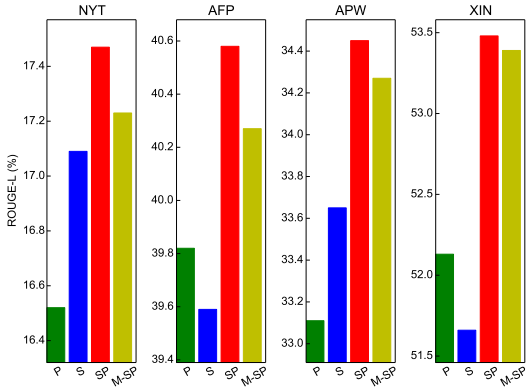
Table 1: ROUGE F1 scores on the combined AFP/APW/XIN/NYT in-domain test set.

here the performance of our Shared (S) baseline model (with the above hyper-parameters) on the original 2K test set used in (Rush et al., 2015). On that test set, our S model obtains 30.13 F1 ROUGE-L score, compared to 28.34 ROUGE-L obtained by the ABS+ model (Rush et al., 2015), and 30.64 ROUGE-L obtained by the words-lvt2k-1sent model (Nallapati et al., 2016). This comparison indicates that our S model is a competitive baseline, making the comparisons against the SP and M-SP models meaningful when using our in-domain and out-of-domain test sets.

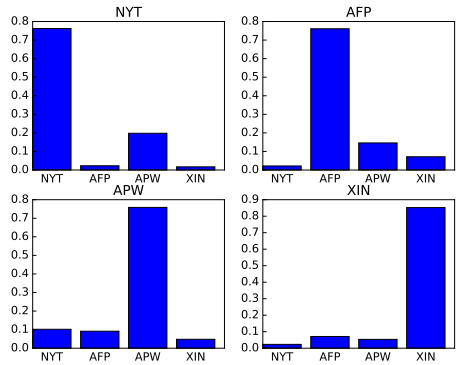
5.1.2 Main Results

The Rouge scores for the in-domain testing data are reported in Table 1 (over the combined AFP/APW/XIN/NYT testset) and Fig. 4a (over individual-style test sets). The numbers indicate that the SP and M-SP models consistently outperform the S and P model, supporting the conclusion that the S model loses important characteristics due to averaging effects, while the P models miss the opportunity to efficiently exploit the training data. Additionally, the performance of SP is consistently better than M-SP in this setting, which indicates that the style label is helpful. As shown in Fig. 4b, the style classifier achieves around 80% accuracy overall in predicting the style under the M-SP model, with some styles (e.g., XIN) being easier to predict than others. The performance of the classifier is directly reflected in the quantitative difference between the SP and M-SP models on individual-style test sets (see Fig. 4a, where the XIN style has the smallest difference between the two models).

The evaluation results for the out-of-domain scenario are reported in Table 2. The numbers indicate that the M-SP model significantly outperforms the S model, supporting the conclusion that the M-SP model is capable of performing on-the-fly adaptation of output style. This conclusion is further strengthened by the style probability distributions shown in Fig 5: they indicate that, for



(a) Rouge-L scores on headline generation, shown separately on four in-domain styles.



(b) Average estimated probability distribution by the M-SP model over the four styles, for each in-domain target style in the test set.

Figure 4: Experimental results on the headline generation task, for in-domain styles.

the out-of-domain CNA style, the output mixture is heavily weighted towards the XIN style (0.6 of the probability mass), while for the LTW style, the output mixture weights heavily the NYT style (0.72 of the probability mass). This result is likely to reflect true style characteristics shared by these publishers, since both CNA and XIN are produced by Chinese news agencies (from Taiwan and mainland China, respectively), while both LTW and NYT are U.S. news agencies owned by the same media corporation.

5.1.3 Experiment Variants

Model capacity In order to remove the possibility that the improved performance of the SP model is due simply to an increased model size compared to the S model, we perform an experiment in which we triple the size of the GRU cell dimensions for the S model. However, we find no significant performance difference compared to the

	CNA Test			LTW Test		
	Rouge-1	Rouge-2	Rouge-L	Rouge-1	Rouge-2	Rouge-L
S	40.73±0.21	17.75±0.18	37.70±0.20	27.08±0.19	8.97±0.15	25.01±0.17
M-SP	**42.00**±0.20	**19.48**±0.21	**39.24**±0.22	**27.79**±0.19	**9.31**±0.18	**25.60**±0.17

Table 2: ROUGE F1 scores on out-of-domain style test sets CNA and LTW.

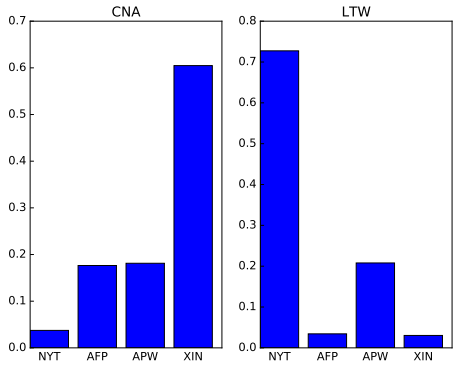


Figure 5: Estimated style probabilities over the four in-domain styles AFP/APW/XIN/NYT, for out-of-domain styles CNA and LTW.

original dimensions (the ROUGE-L score of the triple-size S model is 36.61, compared to 36.51 obtained of the original S model).

Style embedding A competitive approach to modeling different styles is to directly encode the style information into the embedding space. In (Johnson et al., 2016), the style label is converted into a one-hot vector and is concatenated with the word embedding at each time step in the S model. The outputs of this model are at 36.68 ROUGE-L, slightly higher than the baseline S model, but significantly lower than the SP model performance (37.52 ROUGE-L).

Another style embedding approach is to augment the S model with continuous trainable style embeddings for each predefined style label, similar to (Ammar et al., 2016). The resulting outputs achieve 37.2 ROUGE-L, which is better than the S model with one-hot style embedding, but still worse than the SP method (statistically significant at p-value=0.025 using paired t-test). However, neither of these approaches apply to the cases when the style is out-of-domain or unknown during testing. In contrast, such cases are handled naturally by the proposed M-SP model.

Ensemble model Another question is whether the SP model simply benefits from ensembling multiple models rather than style adaptation. To answer this question, we apply a uniform mixture over the private model output along with the shared model output, rather than using the learnt probability distribution from the style classifier. The ROUGE-1/2/L scores are 39.9/19.7/37.0. They are higher than the S model but significantly lower than the SP model and the M-SP model (p-value 0.016). This result confirms that the information that the style classifier encodes is beneficiary, and leads to improved performance.

Style vs. Content Previous experiments indicate that the SP and M-SP models have superior generation accuracy, but it is unclear to what extent the difference comes from improved modeling of style versus modeling of content. To clarify this issue, we performed an experiment in which we replace the named entities appearing in both article and headline with corresponding entity tags, in effect suppressing almost completely any content signal. For instance, given an input such as "China called Thursday on the parties involved in talks on North Korea's nuclear program to show flexibility as a deadline for implementing the first steps of a breakthrough deal approached.", paired with goldtruth output "China urges flexibility as NKorea deadline approaches", we replaced the named entities with their types, and obtained: "LOC_0 called Thursday on the ORG_0 involved in NON_2 on LOC_1 's NON_3 to show NON_0 as a NON_1 for implementing the first NON_4 of a NON_5 approached .", paired with "LOC_0 urges NON_0 as LOC_1 NON_1 approaches."

Under this experimental conditions, both the SP and M-SP models still achieve significantly better performance compared to the S baseline. On the combined AFP/APW/XIN/NYT in-domain test set, the SP model achieves 61.70 ROUGE-L and M-SP achieves 61.52 ROUGE-L, compared to 60.20 ROUGE-L obtained by the S model. On the CNA/LTW out-of-domain test set, M-SP achieves 60.75 ROUGE-L, compared to 59.47 ROUGE-L by the S model.

In Table 3, we show an example which indi-

article	the org_2 is to forge non_1 with the org_3 located in loc_2 , loc_1 , the per_0 of the loc_0 org_4 said tuesday .
title	loc_0 org_0 to forge non_0 with loc_1 org_1
output by S	org_0 to org_1 in non_0
output by M-SP	loc_0 org_0 to forge non_0 with loc_1 org_1
article	loc_0 - born per_0 per_0 will pay non_1 here next month to per_1 , the org_2 (org_1) per_1 who per_1 perished in an non_2 in february , the org_3 said thursday .
title	per_0 to pay non_0 to late org_1 org_0
output by S	per_0 to visit org_0 in non_0
output by M-SP	per_0 to pay non_0 to org_1 org_0

Table 3: Examples of input article (and groundtruth title) and output generated by S and M-SP. Named entities in the training instances (both article and title) are replaced the entity type.

cates the ability of style adaptation benifiting summarization. For instance, we find that both CNA and XIN make more frequent use of the style pattern "xxx will/to [verb] yyy ..., zzz said ???day" (about 15% of CNA articles contain this pattern, while only 2% of LTW articles have it). From Table 3, we can see that the S model sometimes misses or misuses the verb in its output, while the M-SP model does a much better job at capturing both the verb/action as well as other relations (via prepositions, etc.)

Fig. 6 shows the estimated style probabilities over the four styles AFP/APW/XIN/NYT for CNA and LTW, under this experiment condition. We observe that, in this version as well, CNA is closely matching the style of XIN, while LTW is matching that of NYT. The distribution is similar to the one in Fig. 5, albeit a bit flatter as a result of content removal. As such, it supports the conclusion that the classifier indeed learns style (in addition to content) characteristics.

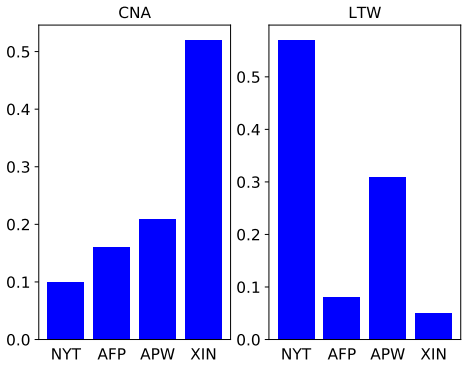


Figure 6: Estimated style probabilities over the four in-domain styles AFP/APW/XIN/NYT, for out-of-domain styles CNA and LTW, after named entities in the article and summary are replaced with entity tags.

6 Conclusion

In this paper, we describe two new style-adaptation model architectures for text sequence generation tasks, SHAPED and Mix-SHAPED. Both versions are shown to significantly outperform models that are either trained in a manner that ignores style characteristics (and hence exhibit a style-averaging effect in their outputs), or models that are trained single-style.

The latter is a particularly interesting result, as a model that is trained (with enough data) on a single-style and evaluated on the same style would be expected to exhibit the highest performance. Our results show that, even for single-style models trained on over 1M examples, their performance is inferior to the performance of SHAPED models on that particular style.

Our conclusion is that the proposed architectures are both efficient and effective in modeling both generic language phenomena, as well as particular style characteristics, and are capable of producing higher-quality abstractive outputs that take into account style characteristics.

References

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

Nabiha Asghar, Pascal Poupart, Xin Jiang, and Hang Li. 2017. Deep active learning for dialogue generation. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*. pages 78–83.

D. Bahdanau, K. Cho, and Y. Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

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

Sumit Chopra, Michael Auli, Alexander M Rush, and SEAS Harvard. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *HLT-NAACL*. pages 93–98.

Sumit Chopra, Suhrid Balakrishnan, and Raghuraman Gopalan. 2013. Dlid: Deep learning for domain adaptation by interpolating between domains. In *ICML workshop on challenges in representation learning*. volume 2.

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

Hal Daumé III. 2009. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815* .

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

Katja Filippova, Enrique Alfonseca, Carlos Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP'15)*.

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

David Graff and C Cieri. 2003. English gigaword corpus. *Linguistic Data Consortium* .

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

Xinyu Hua and Lu Wang. 2017. A pilot study of domain adaptation effect for neural abstractive summarization. *arXiv preprint arXiv:1707.07062* .

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation. *CoRR* abs/1611.04558. http://arxiv.org/abs/1611.04558.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2016. Frustratingly easy neural domain adaptation. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*.

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

Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547* .

Siqi Liu, Zhenhai Zhu, Ning Ye, Sergio Guadarrama, and Kevin Murphy. 2017. Optimization of image description metrics using policy gradient methods. In *International Conference on Computer Vision (ICCV)*.

Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. 2016. Unsupervised domain adaptation with residual transfer networks. In *Advances in Neural Information Processing Systems*. pages 136–144.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of CoNLL*.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*. Association for Computational Linguistics, pages 95–100.

Preksha Nema, Mitesh Khapra, Anirban Laha, and Balaraman Ravindran. 2017. Diversity driven attention model for query-based abstractive summarization. *arXiv preprint arXiv:1704.08300* .

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304* .

Nanyun Peng and Mark Dredze. 2017. Multi-task domain adaptation for sequence tagging. *ACL 2017* page 91.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *CoRR* abs/1511.06732.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of EMNLP*. pages 379–389.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of ACL*.

Ozan Sener, Hyun Oh Song, Ashutosh Saxena, and Silvio Savarese. 2016. Learning transferrable representations for unsupervised domain adaptation. In *Advances in Neural Information Processing Systems*. pages 2110–2118.

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

Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. 2015. Simultaneous deep transfer across domains and tasks. In *Proceedings of the IEEE International Conference on Computer Vision*. pages 4068–4076.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 3156–3164.

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

Kelvin Xu, Jimmy Ba, Ryan Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proc. of the 32nd International Conference on Machine Learning (ICML)*.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. *arXiv preprint arXiv:1703.10931* .

Guangyou Zhou, Zhiwen Xie, Jimmy Xiangji Huang, and Tingting He. 2016. Bi-transferring deep neural networks for domain adaptation. In *ACL (1)*.

Generating Descriptions from Structured Data Using a Bifocal Attention Mechanism and Gated Orthogonalization

Preksha Nema[†*] **Shreyas Shetty**[†*] **Parag Jain**[•*]
Anirban Laha[•†] **Karthik Sankaranarayanan**[•] **Mitesh M. Khapra**[†‡]
[†]IIT Madras, India [•]IBM Research
[‡] Robert Bosch Center for Data Science and Artificial Intelligence, IIT Madras
{preksha,shshett,miteshk}@cse.iitm.ac.in
{pajain34,anirlaha,kartsank}@in.ibm.com

Abstract

In this work, we focus on the task of generating natural language descriptions from a structured table of facts containing *fields* (such as nationality, occupation, *etc*) and *values* (such as Indian, {actor, director}, *etc*). One simple choice is to treat the table as a sequence of *fields* and *values* and then use a standard seq2seq model for this task. However, such a model is too generic and does not exploit task-specific characteristics. For example, while generating descriptions from a table, a human would attend to information at two levels: (i) the fields (macro level) and (ii) the values within the field (micro level). Further, a human would continue attending to a field for a few timesteps till all the information from that field has been rendered and then never return back to this field (because there is nothing left to say about it). To capture this behavior we use (i) a fused bifocal attention mechanism which exploits and combines this micro and macro level information and (ii) a gated orthogonalization mechanism which tries to ensure that a field is remembered for a few time steps and then forgotten. We experiment with a recently released dataset which contains fact tables about people and their corresponding one line biographical descriptions in English. In addition, we also introduce two similar datasets for French and German. Our experiments show that the proposed model gives 21% relative improvement over a recently proposed state of the art method and 10% relative improvement over basic seq2seq models. The code and the datasets developed as a part of this work are publicly available. [1]

1 Introduction

Rendering natural language descriptions from structured data is required in a wide variety of commercial applications such as generating descriptions of products, hotels, furniture, *etc*., from a corresponding table of facts about the entity. Such a table typically contains {*field, value*} pairs where the field is a property of the entity (*e.g.*, *color*) and the value is a set of possible assignments to this property (*e.g.*, *color* = *red*). Another example of this is the recently introduced task of generating one line biography descriptions from a given Wikipedia infobox (Lebret et al., 2016). The Wikipedia infobox serves as a table of facts about a person and the first sentence from the corresponding article serves as a one line description of the person. Figure 1 illustrates an example input infobox which contains fields such as Born, Residence, Nationality, Fields, Institutions and Alma Mater. Each field further contains some words (*e.g.*, particle physics, many-body theory, *etc*.). The corresponding description is coherent with the information contained in the infobox.

Note that the number of fields in the infobox and the ordering of the fields within the infobox varies from person to person. Given the large size (700K examples) and heterogeneous nature of the dataset which contains biographies of people from different backgrounds (sports, politics, arts, *etc*.), it is hard to come up with simple rule-based templates for generating natural language descriptions from infoboxes, thereby making a case for data-driven models. Based on the recent success of data-driven neural models for various other NLG tasks (Bahdanau et al., 2014; Rush et al., 2015; Yao et al., 2015; Chopra et al., 2016; Nema et al., 2017), one simple choice is to treat the infobox as

* The first three authors have contributed equally to this work.

[1] https://github.com/PrekshaNema25/StructuredData_To_Descriptions

Proceedings of NAACL-HLT 2018, pages 1539–1550
New Orleans, Louisiana, June 1 - 6, 2018.

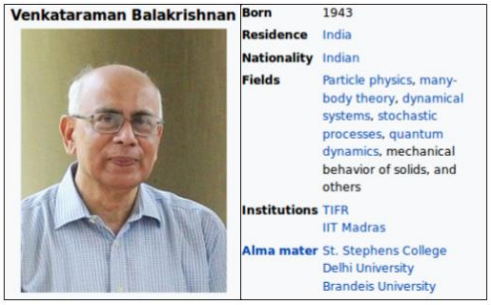


Figure 1: Sample Infobox with description : *V. Balakrishnan (born 1943 as Venkataraman Balakrishnan) is an Indian theoretical physicist who has worked in a number of fields of areas, including particle physics, many-body theory, the mechanical behavior of solids, dynamical systems, stochastic processes, and quantum dynamics.*

a sequence of {field, value} pairs and use a standard seq2seq model for this task. However, such a model is too generic and does not exploit the specific characteristics of this task as explained below.

First, note that while generating such descriptions from structured data, a human keeps track of information at two levels. Specifically, at a macro level, she would first decide which field to mention next and then at a micro level decide which of the values in the field needs to be mentioned next. For example, she first decides that at the current step, the field *occupation* needs attention and then decides which is the next appropriate occupation to attend to from the set of occupations (*actor, director, producer, etc.*). To enable this, we use a bifocal attention mechanism which computes an attention over *fields* at a macro level and over *values* at a micro level. We then fuse these attention weights such that the attention weight for a field also influences the attention over the values within it. Finally, we feed a fused context vector to the decoder which contains both field level and word level information. Note that such two-level attention mechanisms (Nallapati et al., 2016; Yang et al., 2016; Serban et al., 2016) have been used in the context of unstructured data (as opposed to structured data in our case), where at a macro level one needs to pay attention to sentences and at a micro level to words in the sentences.

Next, we observe that while rendering the output, once the model pays attention to a field (say, occupation) it needs to stay on this field for a few timesteps (till all the occupations are produced in the output). We refer to this as the *stay on* behavior. Further, we note that once the tokens of a field are referred to, they are usually not referred to later. For example, once all the occupations have been listed in the output we will never visit the occupation field again because there is nothing left to say about it. We refer to this as the *never look back* behavior. To model the *stay on* behaviour, we introduce a forget (or remember) gate which acts as a signal to decide when to forget the current field (or equivalently to decide till when to remember the current field). To model the *never look back* behaviour we introduce a gated orthogonalization mechanism which ensures that once a field is forgotten, subsequent field context vectors fed to the decoder are orthogonal to (or different from) the previous field context vectors.

We experiment with the WIKIBIO dataset (Lebret et al., 2016) which contains around 700K {infobox, description} pairs and has a vocabulary of around 400K words. We show that the proposed model gives a relative improvement of 21% and 20% as compared to current state of the art models (Lebret et al., 2016; Mei et al., 2016) on this dataset. The proposed model also gives a relative improvement of 10% as compared to the basic seq2seq model. Further, we introduce new datasets for French and German on the same lines as the English WIKIBIO dataset. Even on these two datasets, our model outperforms the state of the art methods mentioned above.

2 Related work

Natural Language Generation has always been of interest to the research community and has received a lot of attention in the past. The approaches for NLG range from (i) rule based approaches (*e.g.*, (Dale et al., 2003; Reiter et al., 2005; Green, 2006; Galanis and Androutsopoulos, 2007; Turner et al., 2010)) (ii) modular statistical approaches which divide the process into three phases (planning, selection and surface realization) and use data driven approaches for one or more of these phases (Barzilay and Lapata, 2005; Belz, 2008; Angeli et al., 2010; Kim and Mooney, 2010; Konstas and Lapata, 2013) (iii) hybrid approaches which rely on a combination of handcrafted rules and corpus statistics (Langkilde and Knight, 1998; Soricut and Marcu, 2006; Mairesse and Walker, 2011) and (iv) the more recent neural network based models (Bahdanau et al., 2014).

Neural models for NLG have been proposed in the context of various tasks such as machine translation (Bahdanau et al., 2014), document summarization (Rush et al., 2015; Chopra et al., 2016), paraphrase generation (Prakash et al., 2016), image captioning (Xu et al., 2015), video summarization (Venugopalan et al., 2014), query based document summarization (Nema et al., 2017) and so on. Most of these models are data hungry and are trained on large amounts of data. On the other hand, NLG from structured data has largely been studied in the context of small datasets such as WEATHERGOV (Liang et al., 2009), ROBOCUP (Chen and Mooney, 2008), NFL RECAPS (Barzilay and Lapata, 2005), PRODIGY-METEO (Belz and Kow, 2009) and TUNA Challenge (Gatt and Belz, 2010). Recently Mei et al. (2016) proposed RNN/LSTM based neural encoder-decoder models with attention for WEATHERGOV and ROBOCUP datasets.

Unlike the datasets mentioned above, the biography dataset introduced by Lebret et al. (2016) is larger (700K {table, descriptions} pairs) and has a much larger vocabulary (400K words as opposed to around 350 or fewer words in the above datasets). Further, unlike the feed-forward neural network based model proposed by (Lebret et al., 2016) we use a sequence to sequence model and introduce components to address the peculiar characteristics of the task. Specifically, we introduce neural components to address the need for attention at two levels and to address the *stay on* and *never look back* behaviour required by the decoder. Kiddon et al. (2016) have explored the use of checklists to track previously visited ingredients while generating recipes from ingredients. Note that two-level attention mechanisms have also been used in the context of summarization (Nallapati et al., 2016), document classification (Yang et al., 2016), dialog systems (Serban et al., 2016), *etc*. However, these works deal with unstructured data (sentences at the higher level and words at a lower level) as opposed to structured data in our case.

3 Proposed model

As input we are given an infobox $\mathcal{I} = \{(g_i, k_i)\}_{i=1}^{M}$, which is a set of pairs (g_i, k_i) where g_i corresponds to *field* names and k_i is the sequence of corresponding *values* and M is the total number of fields in $\mathcal{I}$. For example, (g = *occupation*, k = *actor, writer, director*) could be one such pair in this set. Given such an input, the task is to generate a description $y = y_1, y_2, \ldots, y_m$ containing m words. A simple solution is to treat the infobox as a sequence of *fields* followed by the *values* corresponding to the field in the order of their appearance in the infobox. For example, the infobox could be flattened to produce the following input sequence (the words in bold are field names which act as delimiters)

[Name] *John Doe* ***[Birth_Date]*** *19 March 1981* ***[Nationality]*** *Indian .....*

The problem can then be cast as a seq2seq generation problem and can be modeled using a standard neural architecture comprising of three components (i) an input encoder (using GRU/LSTM cells), (ii) an attention mechanism to attend to important values in the input sequence at each time step and (iii) a decoder to decode the output one word at a time (again, using GRU/LSTM cells). However, this standard model is too generic and does not exploit the specific characteristics of this task. We propose additional components, *viz.*, (i) a fused bifocal attention mechanism which operates on fields (macro) and values (micro) and (ii) a gated orthogonalization mechanism to model *stay on* and *never look back* behavior.

3.1 Fused Bifocal Attention Mechanism

Intuitively, when a human writes a description from a table she keeps track of information at two levels. At the macro level, it is important to decide which is the appropriate field to attend to next and at a micro level (*i.e.*, within a field) it is important to know which values to attend to next. To capture this behavior, we use a bifocal attention mechanism as described below.

Macro Attention: Consider the i-th field g_i which has values $k_i = (w_1, w_2, ..., w_p)$. Let h_i^g be the representation of this field in the infobox. This representation can either be (i) the word embedding of the field name or (ii) some function f of the values in the field or (iii) a concatenation of (i) and (ii). The function f could simply be the sum or average of the embeddings of the values in the field. Alternately, this function could be a GRU (or LSTM) which treats these values within a field as a sequence and computes the field representation as the final representation of this sequence (*i.e.*, the representation of the last timestep). We found that bidirectional GRU is a bet-

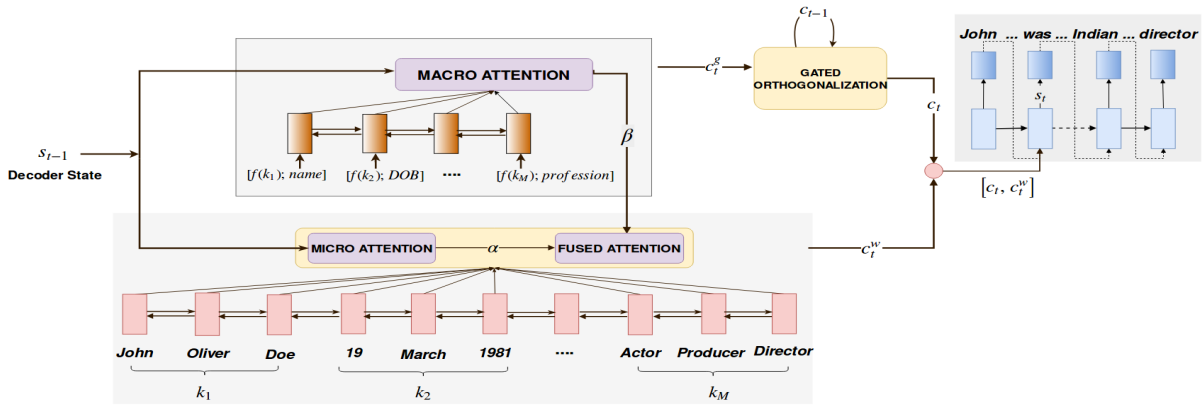


Figure 2: Proposed model

ter choice for f and concatenating the embedding of the field name with this GRU representation works best. Further, using a bidirectional GRU cell to take contextual information from neighboring fields also helps (these are the orange colored cells in the top-left block in Figure 2 with macro attention). Given these representations $\{h_i^g\}_{i=1}^M$ for all the M fields we compute an attention over the fields (macro level).

$$b_{t,i}^g = v_g^T \tanh(U_g s_{t-1} + V_g h_i^g)$$
$$\beta_{t,i} = \frac{\exp(b_{t,i}^g)}{\sum_{l=1}^M \exp(b_{t,l}^g)}$$
$$c_t^g = \sum_{i=1}^M \beta_{t,i} h_i^g \quad (1)$$

where s_{t-1} is the state of the decoder at time step $t-1$. U_g, V_g and v_g are parameters, M is the total number of fields in the input, c_t^g is the macro (field level) context vector at the t-th time step of the decoder.

Micro Attention: Let h_j^w be the representation of the j-th value in a given field. This representation could again either be (i) simply the embedding of this value (ii) or a contextual representation computed using a function f which also considers the other values in the field. For example, if $(w_1, w_2, ..., w_p)$ are the values in a field then these values can be treated as a sequence and the representation of the j-th value can be computed using a bidirectional GRU over this sequence. Once again, we found that using a bi-GRU works better then simply using the embedding of the value. Once we have such a representation computed for all values across all the fields, we compute the attention over these values (micro level) as shown below :

$$a_{t,j}^w = v_w^T \tanh(U_w s_{t-1} + V_w h_j^w) \quad (2)$$
$$\alpha_{t,j}^w = \frac{\exp(a_{t,j}^w)}{\sum_{l=1}^W \exp(a_{t,l}^w)} \quad (3)$$

where s_{t-1} is the state of the decoder at time step $t-1$. U_w, V_w and v_w are parameters, W is the total number of values across all the fields.

Fused Attention: Intuitively, the attention weights assigned to a field should have an influence on all the values belonging to the particular field. To ensure this, we reweigh the micro level attention weights based on the corresponding macro level attention weights. In other words, we fuse the attention weights at the two levels as:

$$\alpha'_{t,j} = \frac{\alpha_{t,j}\beta_{t,F(j)}}{\sum_{l=1}^W \alpha_{t,l}\beta_{t,F(l)}} \quad (4)$$
$$c_t^w = \sum_{j=1}^W \alpha'_{t,j} h_j^w \quad (5)$$

where $F(j)$ is the field corresponding to the j-th value, c_t^w is the macro level context vector.

3.2 Gated Orthogonalization for Modeling Stay-On and Never Look Back behaviour

We now describe a series of choices made to model *stay-on* and *never look back* behavior. We first begin with the *stay-on* property which essentially implies that if we have paid attention to the field i at timestep t then we are likely to pay attention to the same field for a few more time steps. For example, if we are focusing on the *occupation* field at this timestep then we are likely to focus on

it for the next few timesteps till all relevant values in this field have been included in the generated description. In other words, we want to remember the field context vector c_t^g for a few timesteps. One way of ensuring this is to use a remember (or forget) gate as given below which remembers the previous context vector when required and forgets it when it is time to move on from that field.

$$f_t = \sigma(W_t^f c_{t-1}^g + W_g^f c_{t-1} + b_f) \quad (6)$$

$$c_t = (1 - f_t) \odot c_t^g + f_t \odot c_{t-1} \quad (7)$$

where ${W_t}^f, {W_g}^f, b_f$ are parameters to be learned. The job of the forget gate is to ensure that c_t is similar to c_{t-1} when required (i.e., by learning $f_t \rightarrow 1$ when we want to continue focusing on the same field) and different when it is time to move on (by learning that $f_t \rightarrow 0$).

Next, the *never look back* property implies that once we have moved away from a field we are unlikely to pay attention to it again. For example, once we have rendered all the occupations in the generated description there is no need to return back to the occupation field. In other words, once we have moved on ($f_t \rightarrow 0$), we want the successive field context vectors c_t^g to be very different from the previous field vectors c_{t-1}. One way of ensuring this is to orthogonalize successive field vectors using

$$c_t^g = c_t^g - \gamma_t \odot \frac{< c_{t-1}, c_t^g >}{< c_{t-1}, c_{t-1} >} c_{t-1} \quad (8)$$

where $< a, b >$ is the dot product between vectors a and b. The above equation essentially subtracts the component of c_t^g along c_{t-1}. γ_t is a learned parameter which controls the degree of orthogonalization thereby allowing a soft orthogonalization (*i.e.,* the entire component along c_{t-1} is not subtracted but only a fraction of it). The above equation only ensures that c_t^g is soft-orthogonal to c_{t-1}. Alternately, we could pass the sequence of context vectors, $c_1, c_2, ..., c_t$ generated so far through a GRU cell. The state of this GRU cell at each time step would thus be aware of the history of the field vectors till that timestep. Now instead of orthogonalizing c_t^g to c_{t-1} we could orthogonalize c_t^g to the hidden state of this GRU at time-step $t-1$. In practice, we found this to work better as it accounts for all the field vectors in the history instead of only the previous field vector.

In summary, Equation 7 provides a mechanism for remembering the current field vector when appropriate (thus capturing *stay-on* behavior) using a remember gate. On the other hand, Equation 8 explicitly ensures that the field vector is very different (soft-orthogonal) from the previous field vectors once it is time to move on (thus capturing *never look back* behavior). The value of c_t^g computed in Equation 8 is then used in Equation 7. The c_t (macro) thus obtained is then concatenated with c_t^w (micro) and fed to the decoder (see Fig. 2)

4 Experimental setup

We now describe our experimental setup:

4.1 Datasets

We use the WIKIBIO dataset introduced by Lebret et al. (2016). It consists of $728,321$ biography articles from English Wikipedia. A biography article corresponds to a person (sportsman, politician, historical figure, actor, *etc.*). Each Wikipedia article has an accompanying infobox which serves as the structured input and the task is to generate the first sentence of the article (which typically is a one-line description of the person). We used the same train, valid and test sets which were made publicly available by Lebret et al. (2016).

We also introduce two new biography datasets, one in French and one in German. These datasets were created and pre-processed using the same procedure as outlined in Lebret et al. (2016). Specifically, we extracted the infoboxes and the first sentence from the corresponding Wikipedia article. As with the English dataset, we split the French and German datasets randomly into train (80%), test (10%) and valid (10%). The French and German datasets extracted by us has been made publicly available.[2] The number of examples was 170K and 50K and the vocabulary size was 297K and 143K for French and German respectively. Although in this work we focus only on generating descriptions in one language, we hope that this dataset will also be useful for developing models which jointly learn to generate descriptions from structured data in multiple languages.

4.2 Models compared

We compare with the following models:

1. (Lebret et al., 2016): This is a conditional language model which uses a feed-forward neural network to predict the next word in the description conditioned on *local characteristics* (*i.e.,*

[2] https://github.com/PrekshaNema25/StructuredData_To_Descriptions

Model	BLEU-4	NIST-4	ROUGE-4
(Lebret et al., 2016)	34.70	7.98	25.80
(Mei et al., 2016)	35.10	7.27	30.90
Basic Seq2Seq	38.20	8.47	34.28
+Fused bifocal attention	41.22	8.96	38.71
+Gated orthogonalization	**42.03**	**9.17**	**39.11**

Table 1: Comparison of different models on the English WIKIBIO dataset

words within a field) and *global characteristics* (*i.e.,* overall structure of the infobox).

2. (Mei et al., 2016): This model was proposed in the context of the WEATHERGOV and ROBOCUP datasets which have a much smaller vocabulary. They use an improved attention model with additional regularizer terms which influence the weights assigned to the fields.

3. Basic Seq2Seq: This is the vanilla encode-attend-decode model (Bahdanau et al., 2014). Further, to deal with the large vocabulary ($\sim$400K words) we use a copying mechanism as a post-processing step. Specifically, we identify the time steps at which the decoder produces unknown words (denoted by the special symbol UNK). For each such time step, we look at the attention weights on the input words and replace the UNK word by that input word which has received maximum attention at this timestep. This process is similar to the one described in (Luong et al., 2015). Even Lebret et al. (2016) have a copying mechanism tightly integrated with their model.

4.3 Hyperparameter tuning

We tuned the hyperparameters of all the models using a validation set. As mentioned earlier, we used a bidirectional GRU cell as the function f for computing the representation of the fields and the values (see Section 3.1). For all the models, we experimented with GRU state sizes of 128, 256 and 512. The total number of unique words in the corpus is around 400K (this includes the words in the infobox and the descriptions). Of these, we retained only the top 20K words in our vocabulary (same as (Lebret et al., 2016)). We initialized the embeddings of these words with 300 dimensional Glove embeddings (Pennington et al., 2014). We used Adam (Kingma and Ba, 2014) with a learning rate of 0.0004, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We trained the model for a maximum of 20 epochs and used early stopping with the patience set to 5 epochs.

5 Results and Discussions

We now discuss the results of our experiments.

5.1 Comparison of different models

Following Lebret et al. (2016), we used BLEU-4, NIST-4 and ROUGE-4 as the evaluation metrics. We first make a few observations based on the results on the English dataset (Table 1). The basic seq2seq model, as well as the model proposed by Mei et al. (2016), perform better than the model proposed by Lebret et al. (2016). Our final model with bifocal attention and gated orthogonalization gives the best performance and does 10% (relative) better than the closest baseline (basic seq2seq) and 21% (relative) better than the current state of the art method (Lebret et al., 2016). In Table 2, we show some qualitative examples of the output generated by different models.

5.2 Human Evaluations

To make a qualitative assessment of the generated sentences, we conducted a human study on a sample of 500 Infoboxes which were sampled from English dataset. The annotators for this task were undergraduate and graduate students. For each of these infoboxes, we generated summaries using the basic seq2seq model and our final model with bifocal attention and gated orthogonalization. For each description and for each model, we asked three annotators to rank the output of the systems based on i) adequacy (*i.e.* does it capture relevant information from the infobox), (ii) fluency (*i.e.* grammar) and (iii) relative preference (*i.e.*, which of the two outputs would be preferred). Overall the average fluency/adequacy (on a scale of 5) for basic seq2seq model was $4.04/3.6$ and $4.19/3.9$ for our model respectively.

The results from Table 3 suggest that in general gated orthogonalization model performs better than the basic seq2seq model. Additionally, annotators were asked to verify if the generated summaries look natural (*i.e*, as if they were generated by humans). In 423 out of 500 cases, the annotators said "Yes" suggesting that gated orthogonalization model indeed produces good descriptions.

5.3 Performance on different languages

The results on the French and German datasets are summarized in Tables 4 and 5 respectively. Note that the code of (Lebret et al., 2016) is not publicly available, hence we could not report numbers

Reference: Samuel Smiles (23 December 1812 - 16 April 1904), was a Scottish author and government reformer who campaigned on a Chartist platform. **Basic Seq2Seq:** samuel smiles (23 december 1812 – 16 april 1904) was an english books and author. **+Bifocal attention:** samuel smiles (23 december 1812 - 16 april 1904) was a british books and books. **+Gated Orthogonalization:** samuel smiles (23 december 1812 - 16 april 1904) was a british biographies and author.
Reference: Thomas Tenison (29 September 1636 - 14 December 1715) was an English church leader, Archbishop of Canterbury from 1694 until his death. **Basic Seq2Seq:** thomas tenison (14 december 1715 - 29 september 1636) was an english roman catholic archbishop. **+Bifocal attention:** thomas tenison (29 september 1636 - 14 december 1715) was an english clergyman of the roman catholic church. **+Gated Orthogonalization:** thomas tenison (29 september 1636 - 14 december 1715) was archbishop of canterbury from 1695 to 1715.
Reference: Guy F. Cordon (April 24, 1890 - June 8, 1969) was a U.S. politician and lawyer from the state of Oregon. **Basic Seq2Seq:** charles l. mcnary (april 24 , 1890 8 , 1969) was a united states senator from oregon. **+Bifocal attention:**guy cordon (april 24 , 1890 – june 8 , 1969) was an american attorney and politician. **+Gated Orthogonalization:** guy cordon (april 24 , 1890 – june 8 , 1969) was an american attorney and politician from the state of oregon.
Reference: Dr. Harrison B. Wilson Jr. (born April 21, 1925) is an American educator and college basketball coach who served as the second president of Norfolk State University from 1975-1997. **Basic Seq2Seq:** lyman beecher brooks (born april 21 , 1925) is an american educator and educator. **+Bifocal attention:** harrison b. wilson , jr. (born april 21 , 1925) is an american educator and academic administrator. **+Gated Orthogonalization:** harrison b. wilson , jr. (born april 21 , 1925) is an american educator , academic administrator , and former president of norfolk state university.

Table 2: Examples of generated descriptions from different models. For the last two examples, $name$ generated by Basic Seq2Seq model is incorrect because it attended to *preceded by* field.

Metric	A <B	A == B	A >B
Adequacy	186	208	106
Fluency	244	108	148
Preference	207	207	86

Table 3: Qualitative Comparison of Model A (Seq2Seq) and Model B (our model)

Model	BLEU-4	NIST-4	ROUGE-4
(Mei et al., 2016)	10.40	2.51	7.81
Basic Seq2Seq	14.50	3.02	12.22
+Fused bifocal attention	13.80	2.86	12.37
+Gated orthogonalization	**15.52**	**3.30**	**12.80**

Table 4: Comparison of different models on the French WIKIBIO dataset

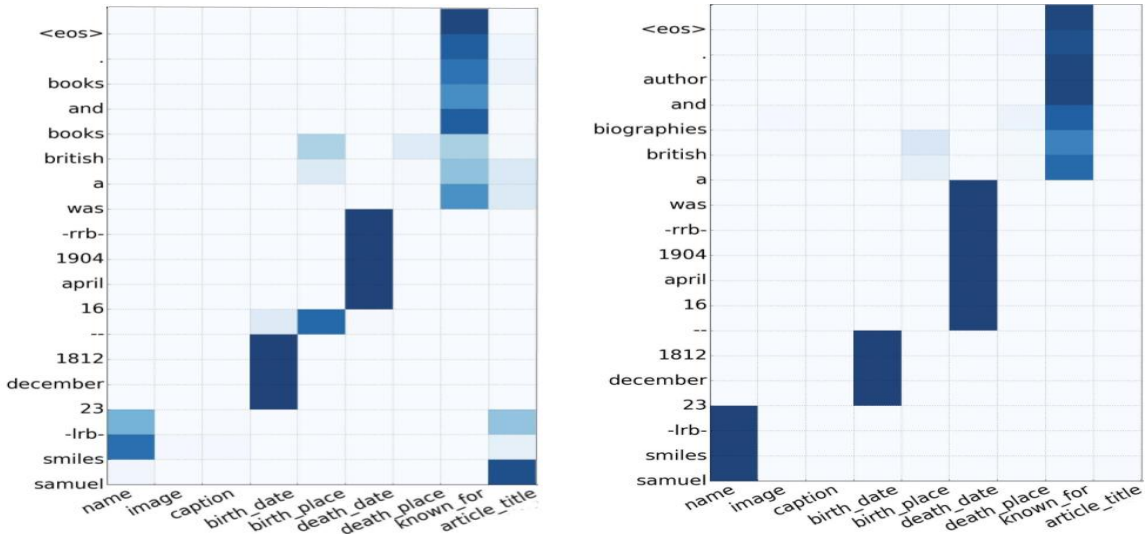
Model	BLEU-4	NIST-4	ROUGE-4
(Mei et al., 2016)	9.30	2.23	5.85
Basic Seq2Seq	17.05	3.09	12.16
+Fused bifocal attention	20.38	3.43	14.89
+Gated orthogonalization	**23.33**	**4.24**	**16.40**

Table 5: Comparison of different models on the German WIKIBIO dataset

for French and German using their model. We observe that our final model gives the best performance - though the bifocal attention model performs poorly as compared to the basic seq2seq model on French. However, the overall performance for French and German are much smaller than those for English. There could be multiple reasons for this. First, the amount of training data in these two languages is smaller than that in English. Specifically, the amount of training data available in French (German) is only 24.2 (7.5)% of that available for English. Second, on average the descriptions in French and German are longer than that in English (EN: 26.0 words, FR: 36.5 words and DE: 32.3 words). Finally, a manual inspection across the three languages suggests that the English descriptions have a more consistent structure than the French descriptions. For example, most English descriptions start with *name* followed by *date of birth* but this is not the case in French. However, this is only a qualitative observation and it is hard to quantify this characteristic of the French and German datasets.

5.4 Visualizing Attention Weights

If the proposed model indeed works well then we should see attention weights that are consistent with the *stay on* and *never look back* behavior. To verify this, we plotted the attention weights in cases where the model with gated orthogonalization does better than the model with only bifocal attention. Figure 3 shows the attention weights corresponding to infobox in Figure 4. Notice that the model without gated orthogonalization has attention on both name field and article title while rendering the name. The model with gated orthogonalization, on the other hand, stays on the name



(a) Fused Bifocal Attention

(b) Fused Bifocal Attention + Gated Orthogonalization

Figure 3: Comparison of the attention weights and descriptions produced for Infobox in Figure 4

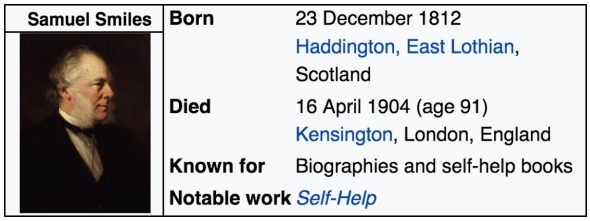


Figure 4: Wikipedia Infobox for Samuel Smiles

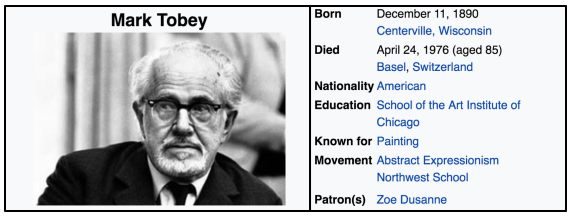


Figure 5: Wikipedia Infobox for Mark Tobey

field for as long as it is required but then moves and never returns to it (as expected).

Due to lack of space, we do not show similar plots for French and German but we would like to mention that, in general, the differences between the attention weights learned by the model with and without gated orthogonalization were more pronounced for the French/German dataset than the English dataset. This is in agreement with the results reported in Table 4 and 5 where the improvements given by gated orthogonalization are more for French/German than for English.

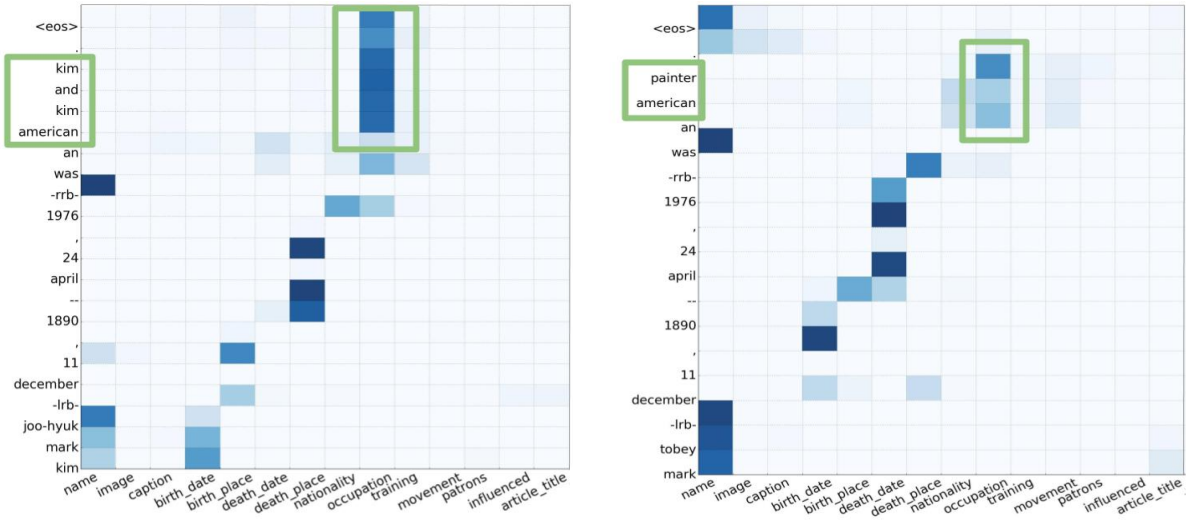
Training data	Target (test) data	
	Arts	Sports
Entire dataset	33.6	52.4
Without target domain data	24.5	29.3
+5k target domain data	31.2	41.8
+10k target domain data	32.2	43.3

Table 6: Out of domain results(BLEU-4)

5.5 Out of domain results

What if the model sees a different *type* of person at test time? For example, what if the training data does not contain any sportspersons but at test time we encounter the infobox of a sportsperson. This is the same as seeing out-of-domain data at test time. Such a situation is quite expected in the products domain where new products with new features (fields) get frequently added to the catalog. We were interested in three questions here. First, we wanted to see if testing the model on out-of-domain data indeed leads to a drop in the performance. For this, we compared the performance of our best model in two scenarios (i) trained on data from all domains (including the target domain) and tested on the target domain (sports, arts) and (ii) trained on data from all domains except the target domain and tested on the target domain. Comparing rows 1 and 2 of Table 6 we observed a significant drop in the performance. Note that the numbers for sports domain in row 1 are much better than the Arts domain because roughly 40% of the WIKIBIO training data contains sportspersons.

Next, we wanted to see if we can use a small



(a) Without fine tuning.

(b) With fine tuning with 5K in-domain data.

Figure 6: Comparison of the attention weights and descriptions (see highlighted boxes) produced by an out-of-domain model with and without fine tuning for the Infobox in Figure 5

amount of data from the target domain to fine tune a model trained on the out of domain data. We observe that even with very small amounts of target domain data the performance starts improving significantly (see rows 3 and 4 of Table 6). Note that if we train a model from scratch with only limited data from the target domain instead of fine-tuning a model trained on a different source domain then the performance is very poor. In particular, training a model from scratch with 10K training instances we get a BLEU score of 16.2 and 28.4 for arts and sports respectively. Finally, even though the actual words used for describing a sportsperson (footballer, cricketer, *etc.*) would be very different from the words used to describe an artist (actor, musician, *etc.*) they might share many fields (for example, date of birth, occupation, *etc.*). As seen in Figure 6 (attention weights corresponding to the infobox in Figure 5), the model predicts the attention weights correctly for common fields (such as occupation) but it is unable to use the right vocabulary to describe the occupation (since it has not seen such words frequently in the training data). However, once we fine tune the model with limited data from the target domain we see that it picks up the new vocabulary and produces a correct description of the occupation.

6 Conclusion

We present a model for generating natural language descriptions from structured data. To address specific characteristics of the problem we propose neural components for fused bifocal attention and gated orthogonalization to address *stay on* and *never look back* behavior while decoding. Our final model outperforms an existing state of the art model on a large scale WIKIBIO dataset by 21%. We also introduce datasets for French and German and demonstrate that our model gives state of the art results on these datasets. Finally, we perform experiments with an out-of-domain model and show that if such a model is fine-tuned with small amounts of in domain data then it can give an improved performance on the target domain.

Given the multilingual nature of the new datasets, as future work, we would like to build models which can jointly learn to generate natural language descriptions from structured data in multiple languages. One idea is to replace the concepts in the input infobox by Wikidata concept ids which are language agnostic. A large amount of input vocabulary could thus be shared across languages thereby facilitating joint learning.

7 Acknowledgements

We thank Google for supporting Preksha Nema through their Google India Ph.D. Fellowship program. We also thank Microsoft Research India for supporting Shreyas Shetty through their generous travel grant for attending the conference.

References

Gabor Angeli, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP '10, pages 502–512. http://dl.acm.org/citation.cfm?id=1870658.1870707.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *ICLR 2015*.

Regina Barzilay and Mirella Lapata. 2005. Collective content selection for concept-to-text generation. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, HLT '05, pages 331–338. https://doi.org/10.3115/1220575.1220617.

Anja Belz. 2008. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Nat. Lang. Eng.* 14(4):431–455. https://doi.org/10.1017/S1351324907004664.

Anja Belz and Eric Kow. 2009. System building cost vs. output quality in data-to-text generation. In *Proceedings of the 12th European Workshop on Natural Language Generation*. Association for Computational Linguistics, Stroudsburg, PA, USA, ENLG '09, pages 16–24. http://dl.acm.org/citation.cfm?id=1610195.1610198.

David L. Chen and Raymond J. Mooney. 2008. Learning to sportscast: A test of grounded language acquisition. In *Proceedings of the 25th International Conference on Machine Learning*. ACM, New York, NY, USA, ICML '08, pages 128–135. https://doi.org/10.1145/1390156.1390173.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 93–98. http://www.aclweb.org/anthology/N16-1012.

Robert Dale, Sabine Geldof, and Jean-Philippe Prost. 2003. Coral : Using natural language generation for navigational assistance. In Michael J. Oudshoorn, editor, *Twenty-Sixth Australasian Computer Science Conference (ACSC2003)*. ACS, Adelaide, Australia, volume 16 of *CRPIT*, pages 35–44.

Dimitrios Galanis and Ion Androutsopoulos. 2007. Generating multilingual descriptions from linguistically annotated owl ontologies: The naturalowl system. In *Proceedings of the Eleventh European Workshop on Natural Language Generation*. Association for Computational Linguistics, Stroudsburg, PA, USA, ENLG '07, pages 143–146. http://dl.acm.org/citation.cfm?id=1610163.1610188.

Albert Gatt and Anja Belz. 2010. Introducing shared tasks to nlg: The tuna shared task evaluation challenges pages 264–293.

Nancy Green. 2006. Generation of biomedical arguments for lay readers. In *Proceedings of the Fourth International Natural Language Generation Conference*. Association for Computational Linguistics, Stroudsburg, PA, USA, INLG '06, pages 114–121. http://dl.acm.org/citation.cfm?id=1706269.1706292.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *EMNLP*. The Association for Computational Linguistics, pages 329–339.

Joohyun Kim and Raymond J. Mooney. 2010. Generative alignment and semantic parsing for learning from ambiguous supervision. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. Association for Computational Linguistics, Stroudsburg, PA, USA, COLING '10, pages 543–551. http://dl.acm.org/citation.cfm?id=1944566.1944628.

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

Ioannis Konstas and Mirella Lapata. 2013. Inducing document plans for concept-to-text generation. In *EMNLP*. ACL, pages 1503–1514. http://dblp.uni-trier.de/db/conf/emnlp/emnlp2013.html#KonstasL13.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '98, pages 704–710. https://doi.org/10.3115/980845.980963.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1203–1213. https://aclweb.org/anthology/D16-1128.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and*

the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '09, pages 91–99. http://dl.acm.org/citation.cfm?id=1687878.1687893.

Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *ACL*.

Franois Mairesse and Marilyn A. Walker. 2011. Controlling user perceptions of linguistic style: Trainable generation of personality traits. *Computational Linguistics* 37(3):455–488. https://doi.org/10.1162/COLI_a_00063.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 720–730. http://www.aclweb.org/anthology/N16-1086.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023* .

Preksha Nema, Mitesh Khapra, Anirban Laha, and Balaraman Ravindran. 2017. Diversity driven attention model for query-based abstractive summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Vancouver, Canada.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Aaditya Prakash, Sadid A. Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2016. Neural paraphrase generation with stacked residual lstm networks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 2923–2934. http://aclweb.org/anthology/C16-1275.

Ehud Reiter, Somayajulu Sripada, Jim Hunter, Jin Yu, and Ian Davy. 2005. Choosing words in computer-generated weather forecasts. *Artif. Intell.* 167(1-2):137–169. http://dblp.uni-trier.de/db/journals/ai/ai167.html#ReiterSHYD05.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 379–389. http://aclweb.org/anthology/D15-1044.

Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press, AAAI'16, pages 3776–3783. http://dl.acm.org/citation.cfm?id=3016387.3016435.

Radu Soricut and Daniel Marcu. 2006. Stochastic language generation using widl-expressions and its application in machine translation and summarization. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL-44, pages 1105–1112. https://doi.org/10.3115/1220175.1220314.

Ross Turner, Somayajulu Sripada, and Ehud Reiter. 2010. Generating approximate geographic descriptions. In Emiel Krahmer and Marit Theune, editors, *Empirical Methods in Natural Language Generation*. Springer, volume 5790 of *Lecture Notes in Computer Science*, pages 121–140. http://dblp.uni-trier.de/db/conf/eacl/enlg2010.html#TurnerSR10.

Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond J. Mooney, and Kate Saenko. 2014. Translating videos to natural language using deep recurrent neural networks. *CoRR* abs/1412.4729. http://arxiv.org/abs/1412.4729.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In David Blei and Francis Bach, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*. JMLR Workshop and Conference Proceedings, pages 2048–2057. http://jmlr.org/proceedings/papers/v37/xuc15.pdf.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 1480–1489. http://www.aclweb.org/anthology/N16-1174.

Kaisheng Yao, Geoffrey Zweig, and Baolin Peng. 2015. Attention with intention for a neural network conversation model. *CoRR* abs/1510.08565. http://arxiv.org/abs/1510.08565.

CliCR: A Dataset of Clinical Case Reports for Machine Reading Comprehension*

Simon Šuster and **Walter Daelemans**
Computational Linguistics & Psycholinguistics Research Center,
University of Antwerp, Belgium
{simon.suster,walter.daelemans}@uantwerpen.be

Abstract

We present a new dataset for machine comprehension in the medical domain. Our dataset uses clinical case reports with around 100,000 gap-filling queries about these cases. We apply several baselines and state-of-the-art neural readers to the dataset, and observe a considerable gap in performance (20% F1) between the best human and machine readers. We analyze the skills required for successful answering and show how reader performance varies depending on the applicable skills. We find that inferences using domain knowledge and object tracking are the most frequently required skills, and that recognizing omitted information and spatio-temporal reasoning are the most difficult for the machines.

1 Introduction

Machine comprehension is a task in which a system reads a text passage and then answers questions about it. The progress in machine comprehension heavily depends on the introduction of new datasets (Burges, 2013), which encourages the development of new algorithms and deepens our understanding of the (linguistic) challenges that can or can not be tackled well by these algorithms. Recently, a number of reading comprehension datasets have been proposed (§ 2), differing in various aspects such as mode of construction, answer-query formulation and required understanding skills. Most are open-domain datasets built from news, fiction and Wikipedia texts. For specialized domains, however, large machine comprehension datasets are extremely scarce (Welbl et al., 2017a), and the required comprehension skills poorly understood. With our work we hope to narrow this gap by proposing a new resource for reading comprehension in the clinical domain, and by analyzing the different types of comprehension skills that are triggered while answering (Sugawara et al., 2017; Lai et al., 2017).

passage:
[...] A gradual improvement in clinical and laboratory status was achieved within 20 days of antituberculous treatment . The patient was then subjected to a thoracic CT scan that also showed significant radiological improvement . *Thereafter , tapering of corticosteroids was initiated with no clinical relapse* . The patient was discharged after being treated for a total of 30 days and continued receiving antituberculous therapy with no reported problems for a total of 6 months under the supervision of his hometown physicians . [...]
query:
If steroids are used , great caution should be exercised on their gradual tapering to avoid _____ .
answer:
relapse (sem_type=problem, cui=C0035020)

Figure 1: An example from the dataset, with the passage sentence relevant for answering italicized. The passage has been shortened for clarity.

Machine comprehension for healthcare and medicine has received little attention so far, although it offers great potential for practical use. A typical application would be clinical decision support, where given a massive amount of text, a clinician asks questions about either external, medical knowledge (reading literature) or about particular patients (reading electronic health records). Currently, patient-specific questions are tackled by manually browsing or searching those records. This task can be facilitated by summarization and QA systems (Demner-Fushman and Lin, 2007; Demner-Fushman et al., 2009), and we believe, by fine-grained machine reading. Reading comprehension systems that perform on a finer level could play an important role especially when combined with

*We provide the information about accessing the dataset, as well as the code for the experiments, at http://github.com/clips/clicr.

Proceedings of NAACL-HLT 2018, pages 1551–1563
New Orleans, Louisiana, June 1 - 6, 2018.

document retrieval to perform machine reading at scale, such as in the models of Chen et al. (2017) and Watanabe et al. (2017) for the general domain.

For our dataset, we construct queries, answers and supporting passages from BMJ Case Reports, the largest online repository of such documents. A case report is a detailed description of a clinical case that focuses on rare diseases, unusual presentation of common conditions and novel treatment methods. Each report contains a *Learning points* section, summarizing the key pieces of information from that report. The learning points are typically paraphrased portions of passage text and do not match passage sentences exactly. We use these learning points to create queries by blanking out a medical entity. To counteract potential errors and inconsistencies due to automated dataset creation, we perform several checks to improve the quality of the dataset (§ 3). Our dataset contains around 100,000 queries on 12,000 case reports, has long support passages (around 1,500 tokens on average) and includes answers which are single- or multi-word medical entities. We show an example from the dataset in Figure 1.

We examine the performance on the dataset in two ways. First, we report machine performance for several baselines and neural readers. To enable a more flexible answer evaluation, we expand the answers with their respective synonyms from a medical knowledge base, and additionally supplement the standard evaluation metrics with BLEU and embedding-based methods. We investigate different ways of representing medical entities in the text and how this affects the neural readers. We obtain the best results with a recurrent neural network (RNN) with gated attention (Dhingra et al., 2017a), but a simple approach based on embedding similarity proves to be a strong baseline as well. Second, we look at how well humans perform on this task, by asking both a medical expert and a novice to answer a portion of the validation set. When categorizing the skills necessary to find the right answer, we observe that a large number of comprehension skills get activated and that prior knowledge in the form of the ability to perform lexico-grammatical inferences matters the most. This suggests that for our dataset and possibly for domain-specific datasets more generally, more background knowledge should be incorporated in machine comprehension models. The current gap between the best machine and the best human performance is nearly 20% F1, which leaves ample space for further study of machine readers on our dataset. In brief, the contributions of our paper are:

- We propose a large dataset for reading comprehension in the medical domain, using clinical case descriptions.
- We carry out an empirical analysis of *a*) system and human performance on reading comprehension, and *b*) comprehension skills that are required for answering the queries correctly and that allow us to position the dataset according to its difficulty on each of the skills.

Dataset	Question origin	Domain	Size
CliCR (this work)	Learning points	Medical	105K
Quasar-S (Dhingra et al., 2017b)	Definitions	Software	37K
SciQ (Welbl et al., 2017a)	Crowdsourced	Science	14K
MedHop (Welbl et al., 2017b)	KB	Drugs	2.5K
Biology (Berant et al., 2014)	Domain expert	Biology	585
Algebra (Kushman et al., 2014)	Crowdsourced	Algebra	514
QA4MRE (Sutcliffe et al., 2013)	Annotator	Various	240

Table 1: Survey of closed-domain reading comprehension datasets. Size: number of questions. We did not include remotely related datasets which concern a different task (e.g. information retrieval) (Roberts et al., 2015; Voorhees and Tice, 2000).

2 Related datasets

Numerous **general-domain datasets** have been recently created to allow machine comprehension using data-intensive methods. These datasets were collected from Wikipedia (Hewlett et al., 2016; Joshi et al., 2017; Rajpurkar et al., 2016), web search queries (Nguyen et al., 2016), news articles (Hermann et al., 2015; Onishi et al., 2016; Trischler et al., 2017), books (Bajgar et al., 2016; Hill et al., 2016; Paperno et al., 2016) and English exams (Lai et al., 2017). In Table 1, we compare our dataset to several **domain-specific datasets** for machine comprehension. In Quasar-S, the queries are constructed from definitions of software entity tags in a community QA website, while in our case the queries are more varied and explicitly relate to the supporting passages. SciQ is a dataset of science exam questions, in which question-answer pairs are used to retrieve the text passages. For each question, four candidate answers are available. In our dataset, the number of candidate answer is much

higher as the candidate answers come from the relatively long passages. Other datasets mentioned in the table are smaller, so they could not be used as training sets for statistical NLP models.

Cloze datasets require the reader to fill in gaps by relying on accompanying text. Representative datasets are Children's Book Test (Hill et al., 2016) and Book Test (Bajgar et al., 2016), in which queries are created by removing a word or a named entity from the running text in a book; and Hermann et al. (2015), who similarly to us blank out entities in abstractive CNN and Daily Mail summaries, but who are only concerned with short proper nouns and short passages. Who-did-what (Onishi et al., 2016) requires the reader to select the person name from a short candidate list that best answers the query about a news event. They do not use summaries for query formation but remove a named entity from the initial sentence in a news article, and then perform information retrieval to find independent passages relevant to the query. Another cloze dataset for language understanding is ROCStories (Mostafazadeh et al., 2016), but it is targeted more towards script knowledge evaluation, and only contains five-sentence stories. Another related task is predicting rare entities only, with a focus on improving a reading comprehension system with external knowledge sources (Long et al., 2017).

Another popular way of creating datasets for reading comprehension is **crowdsourcing** (Rajpurkar et al., 2016; Richardson et al., 2013; Nguyen et al., 2016; Trischler et al., 2017). These datasets exist primarily for the general domain; for specialized domains where background knowledge is crucial, crowdsourcing is intuitively less suitable (Welbl et al., 2017b), although some positive precedent exists for example in crowdsourcing annotations of radiology reports (Cocos et al., 2015). Compared to automated dataset construction, crowdsourcing is more likely to provide high-quality queries and answers. On the other hand, human question generation may also lead to less varied datasets as questions would tend to be of *wh-* type; for cloze datasets, the questions may be more varied and might require readers to possess a different set of skills.[1]

[1]Support for this is given in Sugawara et al. (2017), who show that Who-did-what dataset, for example, requires on average a larger number of reading skills than SQuAD (Rajpurkar et al., 2016) and MCTest (Richardson et al., 2013).

3 Dataset design

We collected the articles from BMJ Case Reports[2]. The data span the years 2005–2016 and amount to almost 12 thousand reports. We removed the HTML boilerplate from the crawled reports using jusText[3], segmented and tokenized the texts with cTakes (Savova et al., 2010), and annotated the medical entities using Clamp (Soysal et al., 2017). We apply two simple heuristics to refine the recognized entities and to decrease their sparsity. Namely, we move the function words (determiners and pronouns) from the beginning of the entity outside of it, and we adjust the entity boundary so that it does not include a parenthetical at the end of the entity. Clamp assigns entities following the i2b2-2010 shared task specifications (Uzuner et al., 2011). For each entity, a concept unique identifier (CUI) is also available, which links it to the UMLS® Metathesaurus® (Lindberg et al., 1993). To check the quality of the recognized entities, we carried out a small manual analysis on 250 entities. We found that in 89% of cases, the boundaries were correct and defined a true entity. Wrongly recognized cases occurred mostly when two entities were coordinated and recognized as one; when a verb was wrongly included in the entity; or when a pre-modifier was left out.

3.1 Query construction

We create a query by replacing a medical entity in one learning point with a blank. For example, in a report describing comorbid disorders of ADHD, we could obtain the following query:

(1) "Patients with ADHD have higher incidence of ___."

The missing entity "enuresis" is taken as the correct answer. Even though one query corresponds to at most one learning point, there can be more than one query built from a learning point. Occasionally, a learning point contains an exact repetition from the passage. These instances would be trivial to answer, so we remove them. We count as an exact match every instance whose longer side to left/right of the query blank coincides with a part in the passage text. This curation step reduces the dataset size by 5%. More commonly, the learning points are paraphrases of crucial parts of the passage. Sometimes, the entity answering the query is expressed

[2]http://casereports.bmj.com/
[3]https://pypi.python.org/pypi/jusText

differently in the passage. For example, in place of "enuresis", the passage might include its synonym "bedwetting". We manage these cases in two ways, by extending the set of answers for a certain query (§ 3.2), and adding a semantic relatedness metric to the standard evaluation (§ 6).

3.2 Answer set

We account for lexical variation of the ground-truth answers (compared to mentions in the passages) by extending each original ground-truth answer a to a set of ground-truth answers A using a knowledge base. Since our entity recognizer already provides the CUI labels, we can use them to obtain the list of alternative word and phrase forms (synonyms, abbreviations and acronyms) from UMLS®.

Similarly to previous work (Choi et al., 2016; Hewlett et al., 2016), for certain queries none of the answers in A occurs verbatim in the passage. We have found upon manual inspection that this is mostly due to lexical variation that is not captured by answer extension, and to a lesser degree, due to the introduction of entirely new information in the learning point and the entity recognition errors. In the empirical part, we use for training only the instances for which at least one answer occurs in the passage, but we evaluate on all instances in the validation and test sets, including those for which $A \cap E = \emptyset$, where E is the set of all entities in the passage. This mimics a likely real-life scenario where the set of ground-truth answers is a priori unknown.

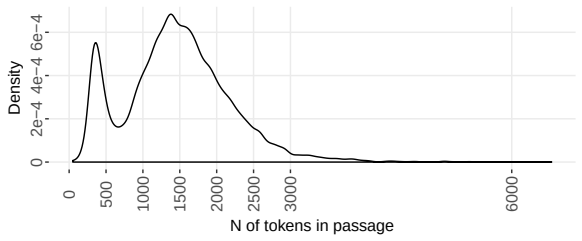
3.3 Task formulation

The reading comprehension problem in our case can be represented as a tuple (q, p, A), where q is the query, built from a learning point; the passage p is the entire report excluding the Learning points section; and A is the set of ground-truth entities answering q. In defining the task, it is important to consider how to take into account entity annotation and how to define the answer output space. We look at these more closely in the rest of this section.

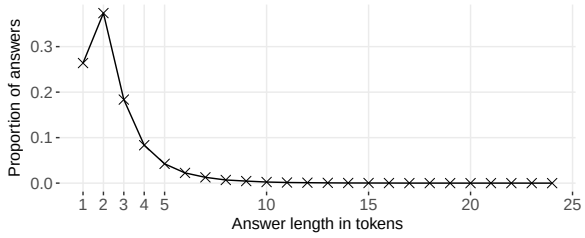
Whenever the entities are marked in the passage, the system can learn to exploit this cue to find the answers more easily (Wang et al., 2017). Although this simplifies the task, it also makes it less realistic as the entities may not be recognized at test time. Realizing that the presence of entities makes the task easier for the machines, Hermann et al. (2015) anonymize the entities, also with a goal of discouraging language model solutions to the queries. In our case, it is not clear how relevant the anonymization is since we deal with medical entities, which have different properties than proper name entities (Kim et al., 2003; Niu et al., 2003). We explore different entity-annotation choices in the empirical part, where we refer to them as **Ent** (entities marked) and **Anonym** (entities marked but anonymized). We further examine a more challenging setup in which the reader can not rely on entity markers as they are not present in the passage (**NoEnt**). In all cases, the reader chooses an answer among the candidates E collected from all entities in the passage.[4] Multi-word entities, which are common in our dataset, are treated as a single token by Ent and Anonym.

N of cases	11,846
N of queries in train/dev/test	91,344/6,391/7,184
N of tokens in passages	16,544,217
N of word types in passages	112,673
N of entity types in passages	591,960
N of distinct answers	56,093
N of distinct answers (incl. extended)	288,211
% answers verbatim in passage	59

Table 2: Data statistics based on the lowercased dataset. For *N of tokens in passages*, we count each passage exactly once, although several queries are normally associated with a passage.



(a)



(b)

Figure 2: Distribution of (a) passage and (b) answer length. Curve (a) is bimodal due to shorter lengths of articles published prior to 2008.

[4] The candidate answers could in principle be obtained also in some other way, so we do not list them in our dataset.

Type	%	Example
problem	67	tuberculosis, abdominal pain, acute myocardial infarction
treatment	22	chemotherapy, surgical intervention, vitamin D suppl.
test	11	MRI, histopathological exam.

Table 3: Answer type statistics.

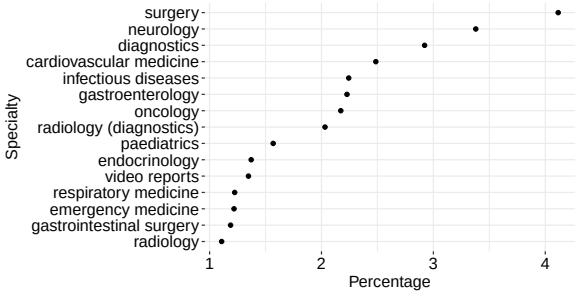


Figure 3: The 15 most common medical specialties represented in the dataset.

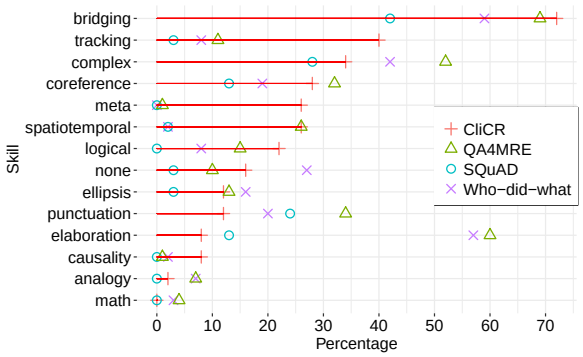


Figure 4: Percentage of times a skill is required in a given dataset. The percentages for the datasets other than ours are from Sugawara et al. (2017).

4 Dataset analysis

We now describe the dataset in more detail, starting with the general statistics summarized in Table 2. It is worth pointing out that the support passages are rather long, which stems from the data origin (journal articles). We show the passage length distribution in Figure 2a, which has the average length of 1,466 tokens. Furthermore, passages are rich with medical entities. There is little repetition of answers—the total of around 100,000 queries are answered by 50,000 distinct entities. Upon extending the answer set with UMLS® we introduce on average four alternative answers for each original one. In 59% of instances, the answer entity is found verbatim in the relevant passage. The answers can belong to any of the problem, treatment or test categories (Table 3), and usually consist of multiple words (Figure 2b). The diversity of medical specialties represented in the articles is shown in Figure 3.

4.1 Analysis of comprehension skills

We estimate the types of skills required in answering by following the categorization of Sugawara et al. (2017). We include the skill definitions with examples from our dataset in Appendix B. We annotated 100 instances in the validation set (with ground-truth answers provided), which yielded on average 2.85 skills per query. The distribution of the required skills is shown in Figure 4. In comparison to the general-domain datasets (SQuAD, Who-did-what), our dataset and QA4MRE (which is also a domain-specific dataset, but with human-generated questions) require more bridging inferences (inferences using background knowledge about the domain), spatio-temporal reasoning and coreference resolution. In our dataset, meta knowledge and object tracking are required more often than in any other dataset. This can be explained by the data origin and the nature of queries. In the case reports, a prominent topic can be discussed which the author refers to in the query, but the query itself is never answered in the passage (meta knowledge). Furthermore, the authors often enumerate medical entities in the query, which leads to the frequent use of object tracking. The queries which were unanswerable are marked as "none". The fraction of these cases was around 16%.

In our experience, the annotation of skills proved quite challenging due to certain confusables. For example, object tracking and coreference both need to maintain the link between objects; object tracking, which includes establishing set relations and membership, may be overlaid with the schematic clause relation skill (subordination); and bridging inference can overlap with coreference resolution. Nevertheless, we adhered to this classification of skills to increase comparability to other datasets included in Figure 4.

5 Methods

5.1 Baselines

Our simplest baselines that we apply on the test set include choosing a random entity (**rand-entity**)

and selecting the most frequent passage entity (**maxfreq-entity**) as the answer. We also include a distance-based method that uses word embeddings (**sim-entity**). Here, we vectorize the passage and the query, and then choose that entity from the passage whose representation has the highest cosine similarity to the query representation:

$$\textit{sim-entity} = \underset{i \in E}{\operatorname{argmax}} \cos\Big(\sum_{j \in C_i} c_j, \sum_{k \in Q} q_k \Big), \quad (1)$$

where $c, q \in \mathbb{R}^d$. The multiset C_i contains the words $\{x_{i-n}, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{i+n}\}$ surrounding the passage entity $i \in E$. We define Q, the context words of the query, likewise. To find out how well the queries can be answered without reading the passage, we also predict the most likely continuation with a language model (**lang-model**). We trained a 4-gram Kneser-Ney model on CliCR training data (with multi-word entities represented as a single token) using SRILM (Stolcke, 2002).

5.2 Neural readers

We apply two types of bidirectional RNNs to our data. Following Wang et al. (2017), we distinguish between *aggregation* readers and *explicit reference* readers, which differ in their formulation of the attention mechanism and how it is being used for answer prediction.

Stanford Attentive (SA) Reader The model proposed by Chen et al. (2016) is an aggregation reader based on the Attentive Reader (Hermann et al., 2015). It predicts the answer using:

$$\hat{a} = \underset{i \in E}{\operatorname{argmax}}\, e_o(i)^T o, \quad (2)$$

where $e_o(i)$ is the answer's output embedding and o is the passage representation obtained by weighting every token representation in the passage with attention: $o = \sum_t \alpha_t h_t$. The attention mechanism is used here to measure the compatibility between token (h_t) and query (q) representations with a bilinear form, $\alpha_t = \text{softmax}_t h_t^T W_\alpha q$. At prediction time, attention should highlight that position t in the passage where the answer occurs. Note that the prediction relies on the aggregate representation o, hence the name of the reader category. As we see in (2), the prediction score does not allow accounting for multi-word entities, unless they are treated as a single token. Returning to our different set-ups based on entity annotation (§ 3.3), this means that we can apply SA reader with Ent and Anonym set-ups, but not with NoEnt, where multi-word answers should be allowed.

Gated-Attention (GA) Reader Dhingra et al. (2017a) investigate neural readers with a fine-grained attention mechanism that learns token representations for the passage that are also conditional on the query, but are in addition refined through multiple hops of the network. The model predicts the answer using attention weights with *explicit reference* to answer positions in the passage:

$$\hat{a} = \underset{i \in E}{\operatorname{argmax}} \sum_{t \in R(i,p)} \alpha_t, \quad (3)$$

where R is the set of indices in passage p at which a token from the candidate i occurs. This operation is also called the pointer sum attention (Kadlec et al., 2016). Since the model marks the references for each token in the answer separately, it allows us to investigate also the NoEnt set-up.[5]

We train each reader with the best hyper-parameters found on the validation set using random search (Bergstra and Bengio, 2012), and evaluate it on the test part of the dataset. We provide more details about parameter optimization in Appendix A. The models use word embeddings pre-trained on biomedical texts.

5.3 Embedding data and pre-training

We induce the word embeddings on a combination of the CliCR training corpus and PubMed abstracts with open-access PMC articles available until 2015 (segmented and tokenized), amounting to over 9 billion tokens (Hakala et al., 2016). Considering the large effect of hyper-parameter selection on the quality of word embeddings (Levy et al., 2015), we optimize the embedding hyper-parameters also using random search.

6 Evaluation

A model f takes as input a passage–query pair and outputs an answer $\hat{a}$.[6] We carry out the evaluation

[5] We assume the candidate entities are known in advance.

[6] In our case, the answer is a word or a word phrase representing a medical entity. Alternatively, one could also take the UMLS® CUI identifier as the answering unit. However, in that case, it would mean that sometimes the original word phrase is lost. This is because entity linking with CUIs can be noisy, and only a part of a word phrase may be linked to the ontology. In the current setup, we are able to keep both the original word phrase as well as the extended answers. The CUI information is still an integral part of the answer field in our dataset, so it can be used by other researchers if preferred.

with different metrics described below. The final score m for a metric v is obtained by averaging over the test set:

$$m_v(f) = \frac{1}{|D_{\text{test}}|} \sum_{(p,q,A) \in D_{\text{test}}} \max_{a \in A} v(f(p,q), a). \quad (4)$$

Since there are multiple correct answers A, we take the highest scoring answer $\hat{a}$ at each instance, as done in Rajpurkar et al. (2016). Note that in the dataset we do not supply the candidate answers; in the experiments, we constrain the candidates to the set of entities in the passage.

The two standardly used metrics for machine comprehension evaluation are the **exact match** (EM) and the **F1** score. For EM, the predicted and the ground truth answers must match precisely, safe for articles, punctuation and case distinction (same for other metrics). F1 metric is applied per instance and measures the overlap between the prediction $\hat{a}$ and the ground truth a, which are treated as bags of words.[7] While these two metrics are arguably sufficient in news-style machine comprehension where the entities are proper nouns which allow for little variation and synonymy, in our case the medical entities are often mostly common nouns modified by specifiers and qualifiers. To take into account potentially large lexical and word-order variation, we use two additional metrics. First, we measure **BLEU** (Papineni et al., 2002) for n-grams of length 2 (shortly, B2) and 4 (B4) using the package by Chen et al. (2015), with which we aim to capture contiguity of tokens in longer answers. Second, it may occur that answers contain no word overlap yet still be good candidates because of their semantical relatedness, as in "renal failure"–"kidney breakdown". We take this into account by using an **embedding metric** (Emb), in which we construct mean vectors for both ground-truth and system answer sequences, and then compare them with the cosine similarity. This and other embedding metrics for evaluation were previously studied in dialog-system research (Liu et al., 2016).

7 Results and analysis

We show the results in Table 4. We see that answer prediction based on contextual representation of queries and passages (sim-entity) achieves a strong base performance that is only outperformed by GA reader. The language model performs poorly on EM and F1, but the embedding-metric score is higher, likely reflecting the fact that the predicted answers—though mostly incorrect—are related to the ground-truth answers. The poor performance means that based on queries alone (without reading the passage), it is difficult to provide accurate answers. The GA reader performs well across all entity set-ups, even when the entities are not marked in the passage. Interestingly, the exact match and BLEU scores in this case are much lower compared to other entity set-ups. Upon inspecting the predicted answers more closely, we have observed that GA-NoEnt tends to predict longer answers than GA-Ent/Anonym. For example, the average predicted answer length for GA-NoEnt was as high as 3.7 tokens, whereas for the other two set-ups and the ground-truth answers the numbers range between 2.3 and 2.5. A plausible explanation for this lies in how GA reaches its prediction (3), which is by accumulating the attention weights without normalizing. This would then drive the model to prefer longer answers. For example, for the ground-truth entity "chest CT", GA-NoEnt predicts "interval CT scans of the chest". Although all neural models use pre-trained word embeddings, for Ent and Anonym the multi-word entities do not have pre-trained embeddings since our embeddings are induced on the word level. This may partly explain the competitive performance of NoEnt compared to Ent. We leave the integration of entity embeddings for the future work.

The results for SA reader are far below the per-

Method	EM	F1	B2	B4	Emb
rand-entity	1.4	5.1	.03	.01	.23
maxfreq-ent.	8.5	12.6	.10	.05	.31
sim-entity	20.8	29.4	.22	.15	.45
lang-model	2.1	3.5	.00	.00	.30
SA-Anonym	19.6	27.2	.22	.16	.43
SA-Ent	6.1	11.4	.07	.05	.31
GA-Anonym	24.5	33.2	.28	.20	.48
GA-Ent	22.2	30.2	.25	.18	.46
GA-NoEnt	14.9	33.9	.21	.11	.51
human-expert	*35*	*53.7*	*.46*	*.23*	*.67*
human-novice	*31*	*45.1*	*.43*	*.24*	*.62*

Table 4: Answering results on the test set. EM and F1 scores are percentages. The human scores (in italics) are based on the validation set.

[7] In precision, the number of correct words is divided by the number of all predicted words. In recall, the former is divided by the number of words in the ground-truth answer.

formance of GA reader. We also see that it performs much better on anonymized entities than on non-anonymized ones. This is in line with Wang et al. (2017) who find that SA reader suffers a drop of 19 points in exact match on Who-did-what dataset when anonymization is not done. A possible explanation is that anonymization reduces the output space to only several hundred entity candidates for which the output embedding needs to be trained. When we do not use anonymization, the set of output entities increases to the set of all entity types found in all passages, which is several orders of magnitude more. While this effect also occurs for GA reader, it is less pronounced because GA reader scores words in the passage and does not need to learn separate answer word embeddings.

7.1 Human performance

To measure the accuracy of human answering, we have used the same sample of data instances as used for the analysis of skills.[8] The queries were answered separately by a novice reader (linguistics background, little-to-none medical knowledge) and by an expert reader (both linguistics and medical background). The annotators needed around 15 minutes on average to read the passage and answer the query. The results are shown at the bottom of Table 4. The expert scores higher across all evaluation metrics, with as much as a 7-point advantage in % F1. This advantage is largely coming from the better performance on those instances where bridging inferences are required (the average F1 score was 10 points higher on these queries), which suggests that domain knowledge is beneficial in the comprehension task. For a novice in a specialized domain, it is harder to build a good situation model that would lead to successful comprehension since it requires more effort—active, strategic processing and establishing ontological relationships in that specific domain. For an expert reader this process is more automatized (Kintsch and Rawson, 2008).

We can see from the table that the best human performance is well below its theoretical upper bound of 100% F1. An important part of explanation for this lies in the automated dataset construction, which leaves certain queries unanswerable, especially when the authors do not refer to a part in the article but introduce completely new information. Another reason is the problem of "answer openness": Typically more than one correct answer is possible and the answers can be correct to various degrees, which we aimed to capture with the use of the embedding metric in the evaluation. Nevertheless, the gap between the best human and machine F1 score is large (around 20 points), leaving considerable space for future applications of machine readers on our dataset.[9]

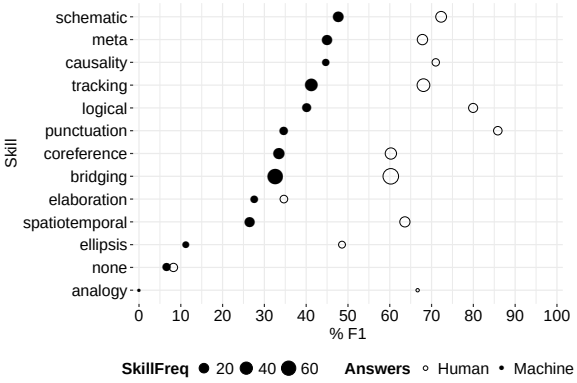


Figure 5: Performance per required skill for the human expert and GA-NoEnt reader.

7.2 Breakdown of results by skill

To see how the answering performance relates to the skill requirements, we have analyzed the part of the validation set annotated with the skills by averaging F1 values for all instances with a particular skill. In this way, we are able to break down both human and machine performance skill-wise, as shown in Figure 5. Because of the small sample size, the results should only be taken as a general indication. The most difficult cases for the GA reader are those annotated with "none" (unanswerable) and "ellipsis" (recognizing implicit and omitted information), ignoring "analogy" for which we only have a single annotated case. Furthermore, spatio-temporal reasoning, elaboration (inferences using general knowledge) and bridging—which is also the most commonly required skill—are the next most difficult ones. The human scores are mostly much higher, which is especially apparent for spatio-temporal reasoning, logical skills and the skill involving punctuation. Our findings align with those of Chu et al. (2017) on the Lambada dataset (Paperno et al., 2016): Although they used a different categorization of comprehension skills, they also find that GA reader has most difficulties with elaboration (which they refer to as "external

[8] Human answers were collected before the skill analysis.

[9] For comparison, the gap for SQuAD was 12.2 and for NewsQA 19.8 (Trischler et al., 2017).

knowledge"), followed by coreference resolution.

8 Conclusion and future work

We have introduced a new dataset for domain-specific reading comprehension in which we have constructed around 100,000 cloze queries from clinical case reports. We analyzed the dataset in terms of the skills required for successful comprehension, and applied various baseline methods and state-of-the-art neural readers. We showed that a large gap still exists between the best machine reader and the expert human reader. One direction for future research is improving the reading models on the queries that are currently the most challenging, i.e. those requiring world and background domain knowledge. Better representing background knowledge by inducing embeddings for entities or otherwise integrating ontological knowledge is in our opinion a promising avenue for future research.

Acknowledgments

We would like to thank Madhumita Sushil and the anonymous reviewers for useful comments. We are also grateful to BMJ Case Reports for allowing the collection of case reports. This work was carried out in the framework of the Accumulate IWT SBO project (nr. 150056), funded by the government agency for Innovation by Science and Technology. We also acknowledge the support of the Nvidia GPU Grant Program.

References

Ondrej Bajgar, Rudolf Kadlec, and Jan Kleindienst. 2016. Embracing data abundance: Booktest dataset for reading comprehension. *arXiv preprint arXiv:1610.00956* .

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D. Manning. 2014. Modeling biological processes for reading comprehension. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 1499–1510. https://doi.org/10.3115/v1/D14-1159.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research* 13(Feb):281–305.

Christopher JC Burges. 2013. Towards the machine comprehension of text: An essay. Technical report, Microsoft Research Technical Report MSR-TR-2013-125, 2013, pdf.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 2358–2367. https://doi.org/10.18653/v1/P16-1223.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to Answer Open-Domain Questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1870–1879. https://doi.org/10.18653/v1/P17-1171.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. 2015. Microsoft COCO captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325* .

E. Choi, D. Hewlett, A. Lacoste, I. Polosukhin, J. Uszkoreit, and J. Berant. 2016. Hierarchical question answering for long documents. *arXiv preprint arXiv:1611.01839* .

Zewei Chu, Hai Wang, Kevin Gimpel, and David McAllester. 2017. Broad context language modeling as reading comprehension. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, pages 52–57. http://www.aclweb.org/anthology/E17-2009.

Anne Cocos, Aaron Masino, Ting Qian, Ellie Pavlick, and Chris Callison-Burch. 2015. Effectively crowdsourcing radiology report annotations. In *Sixth International Workshop on Health Text Mining and Information Analysis (Louhi)*. https://doi.org/10.18653/v1/W15-2614.

Dina Demner-Fushman, Wendy W. Chapman, and Clement J. McDonald. 2009. What can natural language processing do for clinical decision support? *Journal of Biomedical Informatics* 42(5):760 – 772.

Dina Demner-Fushman and Jimmy Lin. 2007. Answering clinical questions with knowledge-based and statistical techniques. *Computational Linguistics* 33(1):63–103.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2017a. Gated-attention readers for text comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1832–1846. https://doi.org/10.18653/v1/P17-1168.

Bhuwan Dhingra, Kathryn Mazaitis, and William W Cohen. 2017b. Quasar: Datasets for Question Answering by Search and Reading. *arXiv preprint arXiv:1707.03904* .

Kai Hakala, Suwisa Kaewphan, Tapio Salakoski, and Filip Ginter. 2016. Syntactic analyses and named entity recognition for pubmed and pubmed central — up-to-the-minute. In *Proceedings of the 15th Workshop on Biomedical Natural Language Processing*. Association for Computational Linguistics, pages 102–107. https://doi.org/10.18653/v1/W16-2913.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*.

Daniel Hewlett, Alexandre Lacoste, Llion Jones, Illia Polosukhin, Andrew Fandrianto, Jay Han, Matthew Kelcey, and David Berthelot. 2016. WikiReading: A Novel Large-scale Language Understanding Task over Wikipedia. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1535–1545. https://doi.org/10.18653/v1/P16-1145.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The Goldilocks Principle: Reading Children's Books with Explicit Memory Representations. In *ICLR*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1601–1611. https://doi.org/10.18653/v1/P17-1147.

Rudolf Kadlec, Martin Schmid, Ondřej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 908–918. https://doi.org/10.18653/v1/P16-1086.

J-D Kim, Tomoko Ohta, Yuka Tateisi, and Junichi Tsujii. 2003. GENIA corpus: a semantically annotated corpus for bio-textmining. *Bioinformatics* 19(suppl 1):i180–i182.

Walter Kintsch and Katherine A. Rawson. 2008. *Comprehension*, Blackwell Publishing Ltd, pages 211–226. https://doi.org/10.1002/9780470757642.ch12.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 271–281. https://doi.org/10.3115/v1/P14-1026.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 796–805. http://aclweb.org/anthology/D17-1083.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association of Computational Linguistics* 3:211–225. http://www.aclweb.org/anthology/Q15-1016.

Donald AB Lindberg, Betsy L Humphreys, and Alexa T McCray. 1993. The Unified Medical Language System. *Methods of information in medicine* 32(04):281–291.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 2122–2132. http://aclweb.org/anthology/D16-1230.

Teng Long, Emmanuel Bengio, Ryan Lowe, Jackie Chi Kit Cheung, and Doina Precup. 2017. World knowledge for reading comprehension: Rare entity prediction with hierarchical lstms using external descriptions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 825–834. http://aclweb.org/anthology/D17-1086.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *ICLR Workshop Papers*.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 839–849. http://www.aclweb.org/anthology/N16-1098.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *arXiv preprint arXiv:1611.09268* .

Yun Niu, Graeme Hirst, Gregory McArthur, and Patricia Rodriguez-Gianolli. 2003. Answering clinical questions with role identification. In *Proceedings of the ACL 2003 workshop on Natural language processing in biomedicine*. Association for Computational Linguistics, pages 73–80.

Takeshi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2016. Who did what: A large-scale person-centered cloze dataset. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 2230–2235. http://aclweb.org/anthology/D16-1241.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernandez. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1525–1534. https://doi.org/10.18653/v1/P16-1144.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '02, pages 311–318. https://doi.org/10.3115/1073083.1073135.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 2383–2392. https://doi.org/10.18653/v1/D16-1264.

Matthew Richardson, J.C. Christopher Burges, and Erin Renshaw. 2013. MCTest: A Challenge Dataset for the Open-Domain Machine Comprehension of Text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 193–203. http://aclweb.org/anthology/D13-1020.

Kirk Roberts, Matthew S Simpson, Ellen M Voorhees, and William R Hersh. 2015. Overview of the TREC 2015 Clinical Decision Support Track. In *TREC*.

Guergana K Savova, James J Masanz, Philip V Ogren, Jiaping Zheng, Sunghwan Sohn, Karin C Kipper-Schuler, and Christopher G Chute. 2010. Mayo clinical Text Analysis and Knowledge Extraction System (cTAKES): architecture, component evaluation and applications. *Journal of the American Medical Informatics Association* 17(5):507–513.

Ergin Soysal, Jingqi Wang, Min Jiang, Yonghui Wu, Serguei Pakhomov, Hongfang Liu, and Hua Xu. 2017. CLAMP — a toolkit for efficiently building customized clinical natural language processing pipelines. *Journal of the American Medical Informatics Association* https://doi.org/10.1093/jamia/ocx132.

Andreas Stolcke. 2002. Srilm – an extensible language modeling toolkit. In *Proc. Int. Conf. Spoken Language Processing (ICSLP 2002)*.

Saku Sugawara, Yusuke Kido, Hikaru Yokono, and Akiko Aizawa. 2017. Evaluation metrics for machine reading comprehension: Prerequisite skills and readability. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 806–817. https://doi.org/10.18653/v1/P17-1075.

Richard FE Sutcliffe, Anselmo Peñas, Eduard H Hovy, Pamela Forner, Álvaro Rodrigo, Corina Forascu, Yassine Benajiba, and Petya Osenova. 2013. Overview of QA4MRE Main Task at CLEF 2013. In *CLEF (Working Notes)*.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. NewsQA: A Machine Comprehension Dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*. Association for Computational Linguistics, pages 191–200. http://www.aclweb.org/anthology/W17-2623.

Özlem Uzuner, Brett R South, Shuying Shen, and Scott L DuVall. 2011. 2010 i2b2/va challenge on concepts, assertions, and relations in clinical text. *Journal of the American Medical Informatics Association* 18(5):552–556.

Ellen M Voorhees and Dawn M Tice. 2000. Building a question answering test collection. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, pages 200–207.

Hai Wang, Takeshi Onishi, Kevin Gimpel, and David McAllester. 2017. Emergent predication structure in hidden state vectors of neural readers. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*. Association for Computational Linguistics. https://doi.org/10.18653/v1/W17-2604.

Yusuke Watanabe, Bhuwan Dhingra, and Ruslan Salakhutdinov. 2017. Question answering from unstructured text by retrieval and comprehension. *arXiv preprint arXiv:1703.08885* .

Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017a. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*. Association for Computational Linguistics, pages 94–106. http://www.aclweb.org/anthology/W17-4413.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2017b. Constructing datasets for multi-hop reading comprehension across documents. *arXiv preprint arXiv:1710.06481* .

A Training details and hyper-parameter optimization

We train the word embeddings using word2vec (Mikolov et al., 2013), and optimize the window size, the model type (CBOW, skip-gram), the dimensionality and the number of negative samples using random search. For the embedding baseline sim-entity, the evaluation was carried out 20 times on the validation part of our dataset, and we chose the parameter configuration that led to the highest-performing embedding model as measured by F1. We find that higher embedding dimensionality works better, that CBOW obtains somewhat better scores than Skipgram, and that medium-sized word windows work best. The best configuration: 'win_size': 5, 'min_freq': 200, 'model': 'cbow', 'dimension': 750, 'neg_samples': 5. The difference between the lowest and the highest scoring model was 3.4 F1. At prediction time (equation (1)) we set the window size to 3, which worked best on the validation set.

For inclusion in the neural readers, it would be impractical to use the high embedding dimensionality found in the hyper-parameter search from the previous paragraph, so we fix the input embedding dimensionality to 200, as done in Chen et al. (2016) to keep the training time practical. We optimize the remaining embedding hyper-parameters just like above. The best parameters were: 'win_size': 4, 'min_freq': 200, 'model': 'cbow', 'dimension': 200, 'neg_samples': 9.

For SA reader, we optimized the hidden state size and the dropout rate using 20 different random configurations. The best values were 70 and 0.57, respectively. We explore the same parameters for the GA reader, but add to the search space the feature that indicates the presence of a passage token in the query, which was found useful in the NoEnt set-up. The best hidden state number and dropout rate were 64 and 0.5, respectively. We used the default values for all the remaining hyper-parameters.

B List of skills with selected examples

In annotating the skills, we followed the categorization by Sugawara et al. (2017):

1. Object tracking: tracking or grasping multiple objects; it is a version of list/enumeration skill used in previous skill classifications
2. Mathematical reasoning: whenever a mathematical operation is involved in finding the answer
3. Coreference resolution: direct reference to an object, includes anaphoras. These include inferential processes based on background knowledge or context.
4. Logical reasoning: conditionals, quantifiers, negation, transitivity
5. Analogy: metaphors, metonymy
6. Causal relation: explicit expression such as "why", "the reason of"
7. Spatio-temporal relations
8. Ellipsis: recognizing implicit or omitted information
9. Bridging: inference through grammatical and lexical knowledge (synonymy, idioms etc). This link however is not automatic or stereotypical, as in the category of elaboration.
10. Elaboration: inference through commonsense reasoning. Note that unlike in the previous category, there is no direct way in which grammatical, lexical or ontological knowledge could help.
11. Meta-knowledge: knowing about the text genre and the main topic being discussed assists in comprehending. In our dataset, knowing the way the queries are constructed (Learning points) is sometimes beneficial.
12. Schematic clause relation: complex sentences that include coordination or subordination
13. Punctuation: understanding parentheses, dashes, quotations, colons etc.

In the following examples, we mark the medical entities in blue, and italicize the parts in the passage that are crucial for answering. Whenever we shorten a part of the passage, we use [...].

B.1 Bridging inference

passage

We report a case of a 72 - year - old Caucasian woman with pl-7 positive antisynthetase syndrome . Clinical presentation included interstitial lung disease , myositis , mechanic 's hands and dysphagia

. As lung injury was the main concern , treatment consisted of *prednisolone and cyclophosphamide* . Complete remission with reversal of pulmonary damage was achieved , as reported by CT scan , pulmonary function tests and functional status . [...]
query
Therefore , in severe cases an aggressive treatment , combining _____ and glucocorticoids as used in systemic vasculitis , is suggested .
answer
cyclophoshamide
explanation The reader needs to have the background knowledge that prednisolone is a glucocorticoid, then it becomes obvious that the answer is cyclophoshamide.

B.2 Object tracking

passage
[...] The patient was managed with supportive measures and the National Poisons Information Service was contacted . A toxicology consultant was involved in view of the unusual mode of administration . Although there was no precedent on how to treat a significant rectal overdose of amitriptyline , *it was advised that the patient be administered a phosphate enema and if failed to adequately remove the tablets then the patient should be given whole bowel irrigation* with 2 litre of Klean - Prep via a nasogastric tube . It was also advised that we admit the patient to a high dependency unit and manage him according to the usual protocol for a tricyclic overdose if complications arose . [...]
query
It seems reasonable to attempt careful removal of the drug from the rectum and if that fails to consider _____ and whole bowel irrigation .
answer
phosphate enemas
explanation The query mentions removal (A), then _____ (B) and whole bowel irrigation (C). In the passage, one needs to track those elements and choose the right one. This skill should be considered whenever the gap is part of an enumeration or is mentioned as a part of another entity.

B.3 Meta knowledge

query
bedaquiline , a new agent with bactericidal and sterilising activity against mycobacterium tuberculosis , is effective against _____ when given together with a background regimen , and is well tolerated and safe if there is awareness of drug interactions and precautions are taken to avoid potential qt prolongation .
answer
tuberculosis
explanation The right answer can be inferred from several parts in the passage (not shown), or even from the title or the query. The query, though, is nowhere in the document explicitly answered.

Learning to Collaborate for Question Answering and Asking

Duyu Tang[‡], **Nan Duan**[‡], **Zhao Yan**[†], **Zhirui Zhang**[♭], **Yibo Sun**[§],
Shujie Liu[‡], **Yuanhua Lv**[♮], **Ming Zhou**[‡]
[‡]Microsoft Research Asia, Beijing, China
[♮]Microsoft AI and Research, Sunnyvale CA, USA
[†]Beihang University, Beijing, China
[♭]University of Science and Technology of China, Anhui, China
[§]Harbin Institute of Technology, Harbin, China
{dutang,nanduan,v-zhaoya,v-zhirzhi,v-yibsu,shujliu,yuanhual,mingzhou}@microsoft.com

Abstract

Question answering (QA) and question generation (QG) are closely related tasks that could improve each other; however, the connection of these two tasks is not well explored in literature. In this paper, we give a systematic study that seeks to leverage the connection to improve both QA and QG. We present a training algorithm that generalizes both Generative Adversarial Network (GAN) and Generative Domain-Adaptive Nets (GDAN) under the question answering scenario. The two key ideas are improving the QG model with QA through incorporating additional QA-specific signal as the loss function, and improving the QA model with QG through adding artificially generated training instances. We conduct experiments on both document based and knowledge based question answering tasks. We have two main findings. Firstly, the performance of a QG model (e.g in terms of BLEU score) could be easily improved by a QA model via a policy gradient. Secondly, directly applying GAN that regards all the generated questions as negative instances could not improve the accuracy of the QA model. Learning when to regard generated questions as positive instances could bring performance boost.

1 Introduction

In this work, we consider the task of joint learning of question answering and question generation. Question answering (QA) and question generation (QG) are closely related natural language processing tasks. The goal of QA is to obtain an answer given a question. The goal of QG is almost reverse which is to generate a question from the answer. In this work, we consider answer selection (Yang et al., 2015; Balakrishnan et al., 2015) as the QA task, which assigns a numeric score to each candidate answer, and selects the top ranked one as the answer. We consider QG as a generation problem and exploit sequence-to-sequence learning (Seq2Seq) (Du et al., 2017; Zhou et al., 2017) as the backbone of the QG model.

The key idea of this work is that QA and QG are two closely tasks and we seek to leverage the connection between these two tasks to improve both QA and QG. Our primary motivations are twofolds. **On one hand**, the Seq2Seq based QG model is trained by maximizing the literal similarity between the generated sentence and the ground truth sentence with maximum-likelihood estimation objective function (Du et al., 2017). However, there is no signal indicating whether or not the generated sentence could be correctly answered by the input. This problem could be precisely mitigated through incorporating QA-specific signal into the QG loss function. **On the other hand**, the capacity of a statistical model depends on the quality and the amount of the training data (Sun et al., 2017). In our scenario, the capacity of the QA model depends on the difference between the positive and negative patterns embodied in the training examples. A desirable training dataset should contain the question-answer pairs that are literally similar yet have different category labels, i.e. some question-answer pairs are correct and some are wrong. However, this kind of dataset is hard to obtain in most situations because of the lack of manual annotation efforts. From this perspective, the QA model could exactly benefit from the QG model through incorporating additional question-answer pairs whose questions are automatically generated by the QG model[1].

To achieve this goal, we present a training algorithm that improves the QA model and the

[1]An alternative way is to automatically generate answers for each question. Solving the problem in this condition requires an answer generation model (He et al., 2017), which is out of the focus of this work. Our algorithm could also be adapted to this scenario.

Proceedings of NAACL-HLT 2018, pages 1564–1574
New Orleans, Louisiana, June 1 - 6, 2018.

QG model in a loop. The QA model improves QG through introducing an additional QA-specific loss function, the objective of which is to maximize the expectation of the QA scores of the generated question-answer pairs. Policy gradient method (Williams, 1992; Yu et al., 2017) is used to update the QG model. In turn, the QG model improves QA through incorporating additional training instances. Here the key problem is how to label the generated question-answer pair. The application of Generative Adversarial Network (GAN) (Goodfellow et al., 2014; Wang et al., 2017) in this scenario regards every generated question-answer pair as a negative instance. On the contrary, Generative Domain-Adaptive Nets (GDAN) (Yang et al., 2017) regards every generated question-answer pair appended with special domain tag as a positive instance. However, it is non-trivial to label the generated question-answer pairs because some of which are good paraphrases of the ground truth yet some might be negative instances with similar utterances. To address this, we bring in a collaboration detector, which takes two question-answer pairs as the input and determines their relation as collaborative or competitive. The output of the collaboration detector is regarded as the label of the generated question-answer pair.

We conduct experiments on both document based (Yang et al., 2015) and knowledge (e.g. web table) based question answering tasks (Balakrishnan et al., 2015). Results show that the performance of a QG model (e.g in terms of BLEU score) could be consistently improved by a QA model via policy gradient. However, regarding all the generated questions as negative instances (*competitive*) could not improve the accuracy of the QA model. Learning when to regard generated questions as positive instances (*collaborative*) could improve the accuracy of the QA model.

2 Related Work

Our work connects to existing works on question answering (QA), question generation (QG), and the use of generative adversarial nets in question answering and text generation.

We consider two kinds of answer selection tasks in this work, one is table as the answer (Balakrishnan et al., 2015) and another is sentence as the answer (Yang et al., 2015). In natural language processing community, there are also other types of QA tasks including knowledge based QA (Berant et al., 2013), community based QA (Qiu and Huang, 2015) and reading comprehension (Rajpurkar et al., 2016). We believe that our algorithm is generic and could also be applied to these tasks with dedicated QA and QG model architectures. Despite the use of sophisticated features could learn a more accurate QA model, in this work we implement a simple yet effective neural network based QA model, which could be conventionally jointly learned with the QG model through back propagation.

Question generation draws a lot of attentions recently, which is partly influenced by the remarkable success of neural networks in natural language generation. There are several works on generating questions from different resources, including a sentence(Heilman, 2011), a topic (Chali and Hasan, 2015), a fact from knowledge base (Serban et al., 2016), etc. In computer vision community, there are also recent studies on generating questions from an image (Mostafazadeh et al., 2016). Our QG model belongs to sentence-based question generation.

GAN has been successfully applied in computer vision tasks (Goodfellow et al., 2014). There are also some recent trials that adapt GAN to text generation (Yu et al., 2017), question answering (Wang et al., 2017), dialogue generation (Li et al., 2016), etc. The relationship of the discriminator and the generator in GAN are competitive. The key finding of this work is that, directly applying the idea of "competitive" in GAN does not improve the accuracy of a QA model. We contribute a generative collaborative network that learns when to collaborate and yields empirical improvements on two QA tasks.

This work relates to recent studies which attempt to improve the performance of a discriminative QA model with generative models (Wang et al., 2017; Yang et al., 2017; Dong et al., 2017; Duan et al., 2017). These works regard QA as the primary task and use auxiliary task, such as question generation and question paraphrasing, to improve the primary task. This is one part of our goal and our another goal is to improve the QG model with the QA system and further to increasingly improve both tasks in a loop.

In terms of assigning category label to the generated question, Generative Adversarial Network (GAN) (Goodfellow et al., 2014; Wang et al.,

2017) and Generative Domain-Adaptive Nets (GDAN) (Yang et al., 2017) could be viewed as special cases of our algorithm. Our algorithm learns when to assign positive or negative labels, while GAN always assigns negative labels and GDAN always assigns positive labels. Besides, our work differs from (Wang et al., 2017) in that our question generation model is a generative model while theirs is actually a discriminative matching model. The approach of (Dong et al., 2017) learns to generate question from question via paraphrasing, and use the generated questions in the inference process. In this work, the QA model and the QG model are applied separately in the inference process. This inspires us to jointly conduct QA and QG in the inference process, which we leave as a future work.

3 Generative Collaborative Network

In this section, we first formulate the task of QA and QG, and then present our algorithm that jointly trains the QA and QG models.

3.1 Task Formulation

This work involves two tasks: question answering (QA) and question generation (QG).

There are different kinds of QA tasks in the natural language processing area. To verify the scalability of our algorithm, we consider two types of answer selection tasks, both of which are fundamental QA tasks in research community and of great importance in industrial applications including web search and chatbot. Both tasks take a question q and a list of candidate answers $A = \{a_1, a_2, ..., a_n\}$ as input, and outputs an answer a_i which has the largest probability to correctly answer the question. The only difference is that the answer in the task of answer sentence selection (Yang et al., 2015) is a natural language sentence, while the answer in table search (Balakrishnan et al., 2015) is a structured table consisting of caption, attributes and cells. Our QA model is abbreviated as $P_{qa}(a, q; \theta_{qa})$, whose output is the probability that q and a being a correct question-answer pair, and the parameter is denoted as θ_{qa}.

The task of QG takes an answer a which is a natural language sentence or a structured table, and outputs a natural language question q which could be answered by a. Inspired by the remarkable progress of sequence-to-sequence (Seq2Seq) learning in natural language generation, we deal with QG in an end-to-end fashion and develop a generative model based on Seq2Seq learning. Our QG model is abbreviated as $P_{qg}(q|a; \theta_{qg})$, whose output is the probability of generating q from a and the parameter is denoted as θ_{qg}.

3.2 Algorithm Description

We describe the joint learning algorithm in this part. The end goal is to leverage the connection of QA and QG to improve the performances on both QA and QG tasks. A brief illustration of the training progress is given in Figure 1 , which includes a QA model, a QG model and a collaboration detector (CD). A formal description of the algorithm is given in Algorithm 1. We can see that the QA model and the QG model both have two training objectives. One part is the standard supervised learning objective based on task-specific supervisions. Another part of the objective is obtained by leveraging auxiliary tasks.

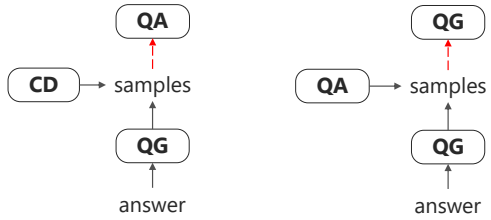


Figure 1: An brief illustrating of the joint training process. The red dashed line stands for the model being updated. QA, QG and CD stand for question answering, question generation and collaboration detection, respectively.

The supervised objective of the QA model is to maximize the probability of the correct category label, and the supervised objective of the QG model is to maximize the probability of the correct question sequence. Since the goal of QA is to predict whether a question-answer pair is correct or not, training the QA model requires negative QA pairs whose labels are zero. But these negative QA pairs are not used for training the QG model as the goal of QG is to generate the correct question.

The main contribution of this work is to explore effective learning objectives that leverage auxiliary tasks. In order to improve the QA model, we generate additional training instances, each of which is composed of a question, an answer and a category label. In this work, we clamp the answer part and feed the answer to the QG model to

Algorithm 1 Generative Collaborative Network for QA and QG

Input: training data D; the batch size for QG training m; the beam size for QG inference k; hyper parameters λ_{qg} and λ_{qg}; hyper parameters b_{qa} and b_{qg}; pretrained collaboration detector $P_{cd}(q, q')$
Output: QA model $P_{qa}(a, q)$ parameterized by θ_{qa}; QG model $P_{qg}(q|a)$ parameterized by θ_{qg}

pretrain $P_{qa}(a, q)$ and $P_{qg}(q|a)$ separately on D
repeat
  get a minibatch of positive QA pairs $PD = \{q_i^p, a_i^p\} \in D, 1 \leq i \leq m$, in which a_i^p is the answer of q_i^p
  get a minibatch of negative QA pairs $ND = \{q_i^n, a_i^n\} \in D, 1 \leq i \leq m$, in which a_i^n is not the answer of q_i^n
  apply $P_{qg}(q|a)$ on PD to generate in a list of question-answer beams $GD = \{q_{ij}^g, a_i^g\}, 1 \leq i \leq m, 1 \leq j \leq k$
  apply $P_{qa}(a, q)$ on GD to assign a QA-specific score to each generated instance
  choose the top ranked result from each beam in GD, and then apply $P_{cd}(q, q')$ on the selected instance
  update θ_{qa} by maximizing the following objective

$$\sum_{i=1}^{m} \Big(logP_{qa}(a_i^p, q_i^p) + log\big(1 - P_{qa}(a_i^n, q_i^n)\big) \Big) + \lambda_{qa} \sum_{i=1}^{m} \Big(\mathbb{I}_{b_{qa}}[P_{cd}(q_i^p, q_{i0}^g)] logP_{qa}(a_i^p, q_{i0}^g) \Big) + \lambda_{qa} \sum_{i=1}^{m} \Big(\big(1 - \mathbb{I}_{b_{qa}}[P_{cd}(q_i^p, q_{i0}^g)]\big) log\big(1 - P_{qa}(a_i^p, q_{i0}^g)\big) \Big) \quad (1)$$

  update θ_{qg} by maximizing the following objective

$$\sum_{i=1}^{m} logP_{qa}(q_i^p|a_i^p) + \lambda_{qg} \sum_{i=1}^{m} \sum_{j=1}^{k} P_{qa}(a_i^p, q_{ij}^g) logP_{qg}(q_{ij}^g|a_i^p) \quad (2)$$

until models converge

generate the question. We use beam search and select the top ranked result as the question.[2] Here the issue is how to infer the label of the generated instance. We believe that heuristically assigning the label as 0 or 1 is problematic. For instance, let us suppose the answer sentence is *"Microsoft was founded by Paul Allen and Bill Gates on April 4, 1975."*, and the ground truth question is *"who founded Microsoft"*. In this case, the generated question *"who is the founder of Microsoft"* is a good one yet *"who is the founder of Google"* and *"how old is Bill Gates"* are both bad cases. To address this, we introduce an additional collaboration detector (CD) to infer the label of the generated instance. Intuitively, the CD acts as a discriminative paraphrase model, which measures the semantic similarity between the ground truth question and generated question. In equation (1), the $\mathbb{I}_{b_{qa}}(x)$ is an indicator function whose value is 1 if the value of x is larger than a threshold b_{qa}, such as 0.5 or 0.3. The hyper parameter λ_{qa} controls the weight of the auxiliary objective to the QA model.

In turn, the QA model is used to assign a QA-specific score $P_{qa}(a, q')$ to each generated question q'. We follow the recent reinforcement learning based approach for dialogue prediction (Li et al., 2016), and define simple heuristic rewards that characterize good questions. The goodness of the generated question is measured by the prediction of the QA model. Similar to the strategy adopted by (Zaremba and Sutskever, 2015), we use a baseline strategy b_{qg} (e.g. 0.5) to decrease the learning variance. The expected reward (Williams, 1992; Yu et al., 2017) for a generated question is given in Equation (2). In this way, the parameters of the QG model could be conventionally updated with stochastic gradient descent.

We pretrain the QA model and the QG model before the joint learning process. The main reason is that a randomized QA model will provide unreliable rewards to the QG model, and a randomized QG model will generate bad questions.

4 The Neural Architecture of Each Module

Our algorithm includes a question answer (QA) model, a question generation (QG) model and a collaboration detector (CD) model. We implement these models with dedicated neural networks.

As we have mentioned before, our training algorithm is applied to both document-based and web table based question answer tasks. In this section, we take table based QA and QG tasks to describe the neural architecture of each module. A question/query q is a natural language expression consisting of a list of words. A table t has fixed

[2] We also implemented using all the beam search results or sampling one result from the beam. However, these tricks do not bring performance boost.

schema including one or more $headers$, one or more $cells$, and a $caption$. A header indicates the property of a column, and a cell is a unit where a row and a column intersects. The caption is typically an explanatory text about the table.

4.1 The Question Answer (QA) Model

We develop a neural network to match a natural language question/query to a structured table. Since a table has multiple aspects including headers, cells and the caption, the model is developed to capture the semantic relevance between a query and a table at different levels.

As the meaning of a query is sensitive to the word order, i.e. the intentions of "*list of flights london to berlin*" and "*list of flights berlin to london*" are totally different, we represent a query with a sequential model. In this work, recurrent neural network (RNN) is used to map a query of variable length to a fixed-length vector. We use gated recurrent unit (GRU) (Cho et al., 2014) as the basic computation unit, which adaptively forgets the history and remembers the input.

$$z_i = \sigma(W_z e_i^q + U_z h_{i-1}) \quad (3)$$

$$r_i = \sigma(W_r e_i^q + U_r h_{i-1}) \quad (4)$$

$$\widetilde{h}_i = \tanh(W_h e_i^q + U_h(r_i \odot h_{i-1})) \quad (5)$$

$$h_i = z_i \odot \widetilde{h}_i + (1 - z_i) \odot h_{i-1} \quad (6)$$

where z_i and r_i are update and reset gates of GRU. We use bi-directional RNN to get the meaning of a query from forward and backward directions, and concatenate two last hidden states as the query vector.

An important property of a table is that exchanging two rows or two columns does not change its meaning. To satisfy this condition, we develop an attention based approach, in which the header and cells are regarded as the external memory. Each header/cell is represented as a vector. Given a query vector, the model calculates the weight of each memory unit and then output a query-specific header/cell representation through weighted average (Bahdanau et al., 2015; Sukhbaatar et al., 2015). This process could be repeated executed for several times, so that more abstractive evidences could be retrieved and composed to support the final decision. Similar techniques have been successfully applied in table-based question answering (Yin et al., 2015b; Neelakantan et al., 2015).

We represent the table caption with RNN, the same strategy we have adopted to represent the query. Element-wised multiplication is used to compose the query vector and the caption vector. Furthermore, since the number of co-occurred words directly reflect the relatedness between the question and the answer, we incorporate the embedding of co-occurred word count as additional features. Finally, we feed the concatenation of all the vectors to a $softmax$ layer whose output length is 2. We have implemented a ranking based loss function $l_{qa} = max(0, 1 - P_{qa}(a, q) + P_{qa}(a', q))$ and a negative log-likelihood (NLL) base loss function $l_{qa} = -\log(P_{qa}(a, q))$. We find that NLL works better and use it in the following experiments.

4.2 The Question Generation (QG) Model

Inspired by the notable progress that sequence-to-sequence learning (Seq2Seq) (Sutskever et al., 2014) has made in natural language generation, we implement a table-to-sequence (Table2Seq) approach that generates natural language question from a structured table.

Table2Seq is composed of an encoder and a decoder. The encoder maps the caption, headers, and cells into continuous vectors, which will be fed to the decoder to generate a question in a sequential way. Similar with the way we have adopted in the QA model, we represent the caption with bidirectional GRU based RNN. The vector of each word in the caption is the concatenation of hidden states from both directions. The vectors of headers and cells are regarded as additional hidden states of the encoder. The representation of each cell is also mixed with the corresponding header representation. The initial vector of the decoder is the average of the caption vector, header vector, and the cell vector.

The backbone of the decoder is an attention based GRU RNN, which generates a word at each time step and repeats the process until generating the end-of-sentence symbol. We made two modifications to adapt the decoder to the table structure. The first modification is that the attention model is calculated over the headers, cells and the caption of a table. Ideally, the decoder should learn to focus on a region of the table when generating a word. The second modification is a table based copying mechanism. It has been proven that the copying mechanism (Gulcehre et al., 2016; Gu

et al., 2016) is an effective way to replicate low-frequent words from the source to the target sequence in sequence-to-sequence learning. In the decoding process, a word is generated either from the target vocabulary via standard $softmax$ or from a table via the copy mechanism. A neural gate g_t is used to trade-off between generating from the target vocabulary and copying from the table. The probability of generating a word y calculated as follows, where $\alpha_t(y)$ is the attention probability of the word y from the table at time step t and $\beta_t(y)$ is the probability of predicting the word y from the $softmax$ at time step t.

$$p_t(y) = g_t \odot \alpha_t(y) + (1 - g_t) \odot \beta_t(y) \quad (7)$$

Since every component of the Table2Seq is differentiable, the parameters could be optimized in an end-to-end fashion with back-propagation. Given a question-answer pair (x, y), the supervised training objective is to maximize the probability of question word at each time step. In the inference process, beam search is used to generate top-k confident results, where k is the beam size.

4.3 The Collaboration Detector (CD)

The goal of a collaboration detector is to determine the label of the instance generated by the QG model. The positive prediction, namely the predicted value is equals to 1, stands for the collaborative relationship between the generated instance and the ground truth, while the negative prediction stands for the competitive relationship.

We consider this task as predicting the category of the given two question-answer pairs, one of which is the ground truth, and another is the generated question-answer pair. Since the answer part is the same, we simplify the problem as classifying two questions as related or not, which is a binary classification problem.

The neural architecture of the collaboration detector (CD) is exactly the same as the caption component in the QA model. We represent two questions with bidirectional RNN, and use element-wise multiplication to do the composition. The result is further concatenated with a co-occurred word count embedding, followed by a $softmax$ layer. The model is trained by minimizing the negative log-likelihood label, which is provided in the training data.

The training data of the CD model includes two parts. The first part is from Quora dataset[3], which is built for detecting if pairs of question text are semantically equivalent. The Quora dataset has 345,989 positive question pairs and 255,027 negative pairs. We further obtain the second part of the training data from the web queries, which are more similar to the web queries in our two QA task. We obtain the query dataset from query logs through clustering the web queries that click the same web page. In this way, we obtain 6,118,023 positive query pairs. We use a heuristic rule to generate the negative instances for the query dataset. For each pair of query text, we clamp the first query and retrieve a query that is mostly similar to the second query. To improve the efficiency of this process, we randomly sample 10,000 queries and define the "similarity" as the number of co-occurred words in two questions. In this way we collect another 6,118,023 negative pairs of query text. We initialize the values of word embeddings with $300d$ Glove vectors[4], which is learned on Wikipedia texts. We use a held-out data consisting of 20K query pairs to check the performance of the CD model. The accuracy of the CD model on the held-out dataset is 83%. In the joint training process, we clamp the parameters of the CD model and use its outputs to facilitate the learning of the QA model.

5 Experiment

We conduct experiments on table-based QA and document-based QA tasks. We will describe experimental settings and report results on these two tasks in this section.

5.1 Table based QA and QG

Setting We take table retrieval (Balakrishnan et al., 2015) as the table-based QA task. Given a query and a collection of candidate table answers, the task aims to return a table that is most relevant to the query. Figure 2 gives an example of this task, in which a query matches to different aspects of a table. We regard document-based QA tasks as a special case of the table-based QA task, in which the cells and the headers are both empty.

We conduct experiments on the web data. The queries come from real-world user queries which we obtain from the search log of a commercial

[3] https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs

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

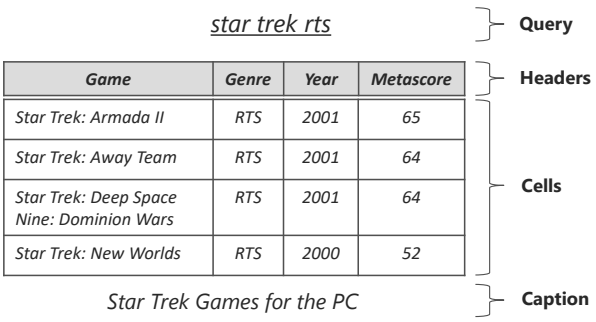


Figure 2: An example illustrating the table-based QA task.

search engine. We filter them down to only those that are directly answered by a table. In this way, we collect 1.49M query-table pairs. An example of the data is given in Figure 2. We randomly select 1.29M as the training set, 0.1M as the dev set and 0.1M as the test set.

We evaluate the performance on table-based QA with *Mean Average Precision (MAP)* and *Precision@1 (P@1)* (Manning et al., 2008). We use the same candidate retrieval adopted in (Yan et al., 2017), namely representing a table as bag-of-words, to guarantee the efficiency of the approach. Each query has 50 candidate tables on average. It is still an open problem to automatically evaluate the performance of a natural language generation system (Lowe et al., 2017). In this work, we use BLEU-4 (Papineni et al., 2002) score as the evaluation metric, which measures the overlap between the generated question and the referenced question. The hyper parameters are tuned on the validation set and the performance is reported on the test set.

Results and Analysis We report the results and our analysis on table-based QA and QG respectively in this part.

We first report the results of single systems on table-based QA. We compare to four single systems implemented by (Yan et al., 2017). In **BM25**, each table is represented as a flattened vector, and the similarity between a query and a table is calculated with the BM25 algorithm. **WordCnt** uses the number of co-occurred words in query-caption pair, query-header pair, and query-cell pair, respectively. **MT based PP** is a phrase-level feature. The features come from a phrase table which is extracted from bilingual corpus via statistical machine translation approach (Koehn et al., 2003). LambdaMART (Burges, 2010) is used to train the ranker. **CNN** uses convolutional neural network to measure the similarity between the query and table caption, table headers, and table cells, respectively. **TQNN** is the table-based QA model implemented in this work, which is regard as the baseline for the joint learning algorithm. Results of single systems are given in Table 1. We can see that BM25 is a simple yet very effective baseline method. Our basic model performs better than all the single models in terms of MAP.

Method	MAP	Acc@1
BM25	0.429	0.294
WordCnt	0.318	0.190
MT based PP	0.327	0.213
CNN	0.359	0.238
TQNN (baseline)	0.439	0.285
Seq2SeqPara	0.437	0.283
GCN (competitive)	0.436	0.282
GCN (collaborative)	0.446	0.292
GCN (final)	**0.456**	**0.301**

Table 1: The performances of single systems on table-based question answering (p-value < 0.01 with t-test between TQNN and GCN).

We also implement four different joint learning settings. In these settings, the QA model and the QG model are all pretrained, and the same way (policy gradient) is used to improve the QG model via the QA predictions. The only difference is how the QA model benefits from the QG model. As we use external resources to train a CD model, we also implement **Seq2SeqPara** for comparison. We train a question generator with a Seq2Seq model on the CD training data, and regard the generated questions as positive instances. Our generative collaborative network is abbreviated as **GCN**. **GCN (competitive)** is analogous to (Goodfellow et al., 2014), where all the generated questions are regarded as negative instances (with label as zero). On the contrary, **GCN (collaborative)** is analogous to (Yang et al., 2017), where the generated questions are regard as positive instances. Our main observation from Table 1 is that simply regarding all the generated questions as negative instances ("competitive") could not bring performance boost. On the contrary, regarding the generated questions as positive ones ("collaborative") improves the QA model. Our algorithm (GCN) significantly improves the TQNN model. Based on these results, we believe that the relationship

between the QA model and the QG model should not be always competitive. Learning when to collaborate through leveraging a CD model is a practical way to improve the performance on question answering.

As described in Equation (1), the influence of the CD model on the QA model also depends on the value of the hyper parameter b_{qa}. A small value of b_{qa} stands for a preference of "collaborative", while a large value of b_{qa} represents a preference of "competitive". Results are given in Figure 3. The GCN model performs better when b_{qa} is in the range $[0.3, 0.5]$, in which the model prefers "collaborative".

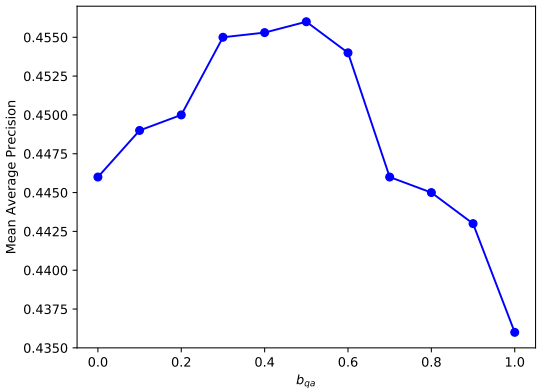


Figure 3: The performances of GCN on table-based QA with different values of b_{qa}. GCN falls back to "competitive" when b_{qa} equals to one. GCN is totally collaborative when b_{qa} equals to zero.

We conduct an additional experiment to test whether our algorithm could improve an existing system. We take BM25 as the baseline, and incorporate one of the five joint models as an additional feature. LambdaMART is used to train the combined ranker. Results are given in Table 2. We can see that the baseline system could be dramatically improved by our system, despite the improvements of different approaches are on par.

Method	MAP	Acc@1
BM25	0.429	0.294
BM25 + TQNN (baseline)	0.650	0.513
BM25 + GAN	0.654	0.519
BM25 + Seq2SeqPara	0.650	0.513
BM25 + GDAN	0.658	0.523
BM25 + GCN (this work)	**0.660**	**0.526**

Table 2: The performances of combined systems on table-based question answering..

We have reported the results on table-based QA. Here we show the performances of different approaches on table-based QG. Results in terms of BLEU-4 are given in Table 3. Different from the trends on QA, "competitive" performs better than "collaborative" on QG. This is reasonable because as the joint training progresses, the QA model in "collaborative" keeps telling the QG model that the generated instances are good enough. On the contrary, the "competitive" model is more critical, which tells the QG model how wrong the generated questions are. In this way, the QG model could be increasingly improved by the QA signal. The QG model is easier to be improved compared to the QA model. Our GCN approach obtains a significant improvement over the baseline model on this task.

Method	Dev	Test
Table2Seq (baseline)	15.71	15.54
Seq2SeqPara	17.01	16.95
GCN (competitive)	17.28	17.34
GCN (collaborative)	16.77	16.66
GCN (final)	**17.59**	**17.61**

Table 3: The performances on table-based question generation. Evaluation metric is BLEU-4 score.

We also report the learning curve of the GCN model as the joint training progresses. The performance on the dev set is given in Figure 4.

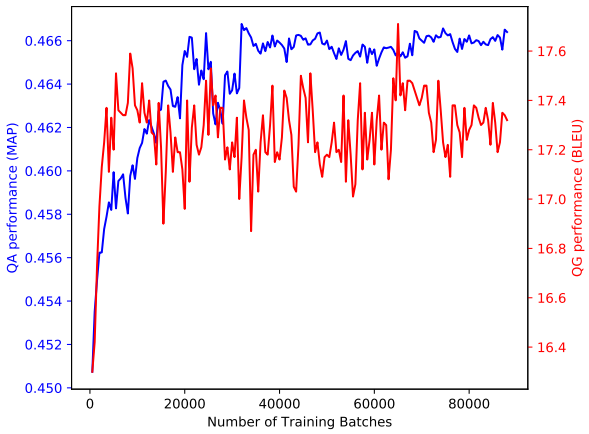


Figure 4: The learning curve of GCN on the dev data. The evaluation metrics are MAP (for QA) and BLEU (for QG).

5.2 Document based QA and QG

To test the scalability of the algorithm, we also apply it to document based QA and QG tasks. The QA task is answer sentence selection (Yang et al.,

Method	MAP	P@1
WordCnt	0.395	0.179
CDSSM (Shen et al., 2014)	0.442	0.228
ABCNN (Yin et al., 2015a)	0.469	0.263
DSL (Tang et al., 2017)	0.484	0.275
DQNN (baseline)	0.471	0.263
Seq2SeqPara	0.470	0.260
GCN (competitive)	0.468	0.257
GCN (collaborative)	0.476	0.272
GCN (final)	**0.492**	**0.282**

Table 4: The performance on document-based QA task (p-value < 0.05 with t-test between DQNN and GCN).

2015). Given a question and a list of candidate answer sentences from a document, the goal is to find a most relevant answer sentence as the answer. Since the WikiQA dataset (Yang et al., 2015) is too small to learn a powerful question generator, we use the MARCO dataset (Nguyen et al., 2016), which is originally designed for reading comprehension yet also has manually annotated labels for sentence/passage selection. A characteristic of MARCO dataset is that the ground truth of the test is invisible to the public. Therefore, we follow (Tang et al., 2017) and split the original validation set into the dev set and the test set. The results on QA and QG are given in Table 4 and Table 5. We can see that the results are almost consistent with the results on table-based QA and QG tasks. Our GCN algorithms achieves promising performances compared to strong baseline methods.

Method	BLEU-4
Seq2Seq (baseline)	8.87
DSL (Tang et al., 2017)	9.31
Seq2SeqPara	9.16
GCN (competitive)	9.22
GCN (collaborative)	9.04
GCN (final)	**9.89**

Table 5: The performance on document-based QG task.

6 Conclusion

We present an algorithm dubbed generative collaborative network for jointly training the question answering (QA) model and the question generation (QG) model. Different from standard GAN, the relationship between QA model (discriminator) and the QG model (generator) in our algorithm is not always competitive. We show that "collaborative" performs better than "competitive" in terms of QA accuracy, and our algorithm that learns when to collaborate obtains further improvement on both QA and QG tasks.

This work could be further improved from several directions. Our current algorithm focuses on the joint training of QA and QG models, while the inferences of these two models are independent. How to conduct joint inference is an interesting future work. Besides, the samples are currently generated from the QG model via beam search. Improving the diversity of the samples requires different sampling mechanisms. Another potential direction is to jointly learn the collaboration detector together with the QA and QG models.

References

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

Sreeram Balakrishnan, Alon Y Halevy, Boulos Harb, Hongrae Lee, Jayant Madhavan, Afshin Rostamizadeh, Warren Shen, Kenneth Wilder, Fei Wu, and Cong Yu. 2015. Applying webtables in practice. In *CIDR*.

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

Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Microsoft Research Technical Report MSR-TR-2010-82* 11(23-581):81.

Yllias Chali and Sadid A Hasan. 2015. Towards topic-to-question generation. *Computational Linguistics* .

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

Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. 2017. Learning to paraphrase for question answering. *arXiv preprint arXiv:1708.06022* .

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Association for Computational Linguistics (ACL)*.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. *Proceeding of EMNLP* .

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

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393* .

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. *arXiv preprint arXiv:1603.08148* .

Shizhu He, Cao Liu, Kang Liu, and Jun Zhao. 2017. Generating natural answers by incorporating copying and retrieving mechanisms in sequence-to-sequence learning. In *Proceedings of ACL*. pages 199–208.

Michael Heilman. 2011. *Automatic factual question generation from text*. Ph.D. thesis, Carnegie Mellon University.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. *Proceedings of NAACL-HLT* 1:48–54.

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

Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic turing test: Learning to evaluate dialogue responses. In *Proceedings of ACL*. pages 1116–1126.

Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. 2008. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge.

Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Margaret Mitchell, Xiaodong He, and Lucy Vanderwende. 2016. Generating natural questions about an image. *arXiv preprint arXiv:1603.06059* .

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

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268* .

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*. pages 311–318.

Xipeng Qiu and Xuanjing Huang. 2015. Convolutional neural tensor network architecture for community-based question answering. In *IJCAI*. pages 1305–1311.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of EMNLP*. pages 2383–2392.

Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. In *ACL*. pages 588–598.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the Conference on Information and Knowledge Management (CIKM)*. pages 101–110.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems (NIPS)*. pages 2431–2439.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. 2017. Revisiting unreasonable effectiveness of data in deep learning era. *arXiv preprint arXiv:1707.02968* .

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

Duyu Tang, Nan Duan, Tao Qin, and Ming Zhou. 2017. Question answering and question generation as dual tasks. *arXiv preprint arXiv:1706.02027* .

Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. Irgan: A minimax game for unifying generative and discriminative information retrieval models. *arXiv preprint arXiv:1705.10513* .

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Zhao Yan, Duyu Tang, Nan Duan, Junwei Bao, Yuanhua Lv, Ming Zhou, and Zhoujun Li. 2017. Content-based table retrieval for web queries. *arXiv preprint arXiv:1706.02427* .

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *EMNLP*. Citeseer, pages 2013–2018.

Zhilin Yang, Junjie Hu, Ruslan Salakhutdinov, and William W Cohen. 2017. Semi-supervised qa with generative domain-adaptive nets. *arXiv preprint arXiv:1702.02206* .

Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. 2015a. Neural generative question answering. *arXiv preprint arXiv:1512.01337* .

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

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*. pages 2852–2858.

Wojciech Zaremba and Ilya Sutskever. 2015. Reinforcement learning neural turing machines. *arXiv preprint arXiv:1505.00521* 419.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural question generation from text: A preliminary study. *arXiv preprint arXiv:1704.01792* .

Learning to Rank Question-Answer Pairs using Hierarchical Recurrent Encoder with Latent Topic Clustering

Seunghyun Yoon, Joongbo Shin and Kyomin Jung
Dept. of Electrical and Computer Engineering
Seoul National University, Seoul, Korea
{mysmilesh,jbshin,kjung}@snu.ac.kr

Abstract

In this paper, we propose a novel end-to-end neural architecture for ranking candidate answers, that adapts a hierarchical recurrent neural network and a latent topic clustering module. With our proposed model, a text is encoded to a vector representation from an word-level to a *chunk-level* to effectively capture the entire meaning. In particular, by adapting the hierarchical structure, our model shows very small performance degradations in longer text comprehension while other state-of-the-art recurrent neural network models suffer from it. Additionally, the latent topic clustering module extracts semantic information from target samples. This clustering module is useful for any text related tasks by allowing each data sample to find its nearest topic cluster, thus helping the neural network model analyze the entire data. We evaluate our models on the Ubuntu Dialogue Corpus and consumer electronic domain question answering dataset, which is related to Samsung products. The proposed model shows state-of-the-art results for ranking question-answer pairs.

1 Introduction

Recently neural network architectures have shown great success in many machine learning fields such as image classification, speech recognition, machine translation, chat-bot, question answering, and other task-oriented areas. Among these, the automatic question answering (QA) task has long been considered a primary objective of artificial intelligence.

In the commercial sphere, the QA task is usually tackled by using pre-organized knowledge bases and/or by using information retrieval (IR) based methods, which are applied in popular intelligent voice agents such as *Siri*, *Alexa*, and *Google Assistant* (from Apple, Amazon, and Google, respectively). Another type of advanced QA systems is IBM's Watson who builds knowledge bases from unstructured data. These raw data are also indexed in search clusters to support user queries (Fan et al., 2012; Chu-Carroll et al., 2012).

In academic literature, researchers have intensely studied sentence pair ranking task which is core technique in QA system. The ranking task selects the best answer among candidates retrieved from knowledge bases or IR based modules. Many neural network architectures with end-to-end learning methods are proposed to address this task (Yin et al., 2016; Wang and Jiang, 2016; Wang et al., 2017). These works focus on matching sentence-level text pair (Wang et al., 2007; Yang et al., 2015; Bowman et al., 2015). Therefore, they have limitations in understanding longer text such as multi-turn dialogue and explanatory document, resulting in performance degradation on ranking as the length of the text become longer.

With the advent of the huge multi-turn dialogue corpus (Lowe et al., 2015), researchers have proposed neural network models to rank longer text pair (Kadlec et al., 2015; Baudiš et al., 2016). These techniques are essential for capturing context information in multi-turn conversation or understanding multiple sentences in explanatory text.

In this paper, we focus on investigating a novel neural network architecture with additional data clustering module to improve the performance in ranking answer candidates which are longer than a single sentence. This work can be used not only for the QA ranking task, but also to evaluate the relevance of next utterance with given dialogue generated from the dialogue model. The key contributions of our work are as follows:

First, we introduce a Hierarchical Recurrent Dual Encoder (HRDE) model to effectively calculate the affinity among question-answer pairs to determine the ranking. By encoding texts from an word-level to a chunk-level with hierarchi-

Proceedings of NAACL-HLT 2018, pages 1575–1584
New Orleans, Louisiana, June 1 - 6, 2018.

cal architecture, the HRDE prevents performance degradations in understanding longer texts while other state-of-the-art neural network models suffer.

Second, we propose a Latent Topic Clustering (LTC) module to extract latent information from the target dataset, and apply these additional information in end-to-end training. This module allows each data sample to find its nearest topic cluster, thus helping the neural network model analyze the entire data. The LTC module can be combined to any neural network as a source of additional information. This is a novel approach using latent topic cluster information for the QA task, especially by applying the combined model of HRDE and LTC to the QA pair ranking task.

Extensive experiments are conducted to investigate efficacy and properties of the proposed model. Our proposed model outperforms previous state-of-the-art methods in the Ubuntu Dialogue Corpus, which is one of the largest text pair scoring datasets. We also evaluate the model on real world QA data crawled from crowd-QA web pages and from Samsung's official web pages. Our model also shows the best results for the QA data when compared to previous neural network based models.

2 Related Work

Researchers have released question and answer datasets for research purposes and have proposed various models to solve these datasets. (Wang et al., 2007; Yang et al., 2015; Tan et al., 2015) introduced small dataset to rank sentences that have higher probabilities of answering questions such as WikiQA and insuranceQA. To alleviate the difficulty in aggregating datasets, that are large and have no license restrictions, some researchers introduced new datasets for sentence similarity rankings (Baudiš et al., 2016; Lowe et al., 2015). As of now, the Ubuntu Dialogue dataset is one of the largest corpus openly available for text ranking.

To tackle the Ubuntu dataset, (Lowe et al., 2015) adopted the "term frequency-inverse document frequency" approach to capture important words among context and next utterances (Ramos et al., 2003). (Bordes et al., 2014; Yu et al., 2014) proposed deep neural network architecture for embedding sentences and measuring similarities to select answer sentence for a given question. (Kadlec et al., 2015) used convolution neural network (CNN) architecture to embed the sentence while a final output vector was compared to the target text to calculate the matching score. They also tried using long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997), bidirectional LSTM and ensemble method with all of those neural network architectures and achieved the best results on the Ubuntu Dialogues Corpus dataset. Another type of neural architecture is the RNN-CNN model, which encodes each token with a recurrent neural network (RNN) and then feeds them to the CNN (Baudiš et al., 2016). Researchers also introduced an attention based model to improve the performance (Tan et al., 2015; Wang and Jiang, 2016; Wang et al., 2017).

Recently, the hierarchical recurrent encoder-decoder model was proposed to embed contextual information in user query prediction and dialogue generation tasks (Sordoni et al., 2015; Serban et al., 2016). This shows improvement in the dialogue generation model where the context for the utterance is important. As another type of neural network architecture, memory network was proposed by (Sukhbaatar et al., 2015). Several researchers adopted this architecture for the reading comprehension (RC) style QA tasks, because it can extract contextual information from each sentence and use it in finding the answer (Xiong et al., 2016; Kumar et al., 2016). However, none of this research is applied to the QA pair ranking task directly.

3 Model

In this section, we depict a previously released neural text ranking model, and then introduce our proposed neural network model.

3.1 Recurrent Dual Encoder (RDE)

A subset of sequential data is fed into the recurrent neural network (RNN) which leads to the formation of the network's internal hidden state h_t to model the time series patterns. This internal hidden state is updated at each time step with the input data w_t and the hidden state of the previous time step h_{t-1} as follows:

$$h_t = f_\theta(h_{t-1}, w_t), \tag{1}$$

where f_θ is the RNN function with weight parameter θ, h_t is hidden state at t-th word input, w_t is t-th word in a target question $\boldsymbol{w}^Q = \{w^Q_{1:t_q}\}$ or an answer text $\boldsymbol{w}^A = \{w^A_{1:t_a}\}$.

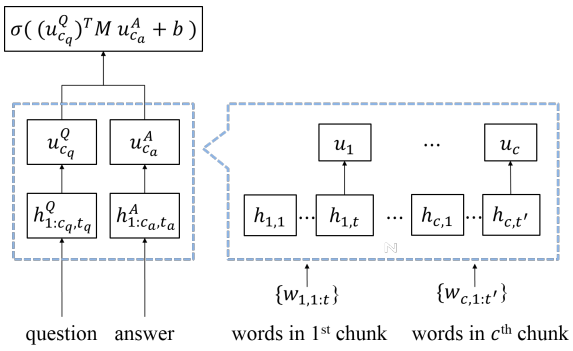


Figure 1: Diagram of the HRDE model. The word-lever RNN encodes words sequences of each chunk. The the final hidden status of the word-level RNN is fed into chunk-level RNN.

The previous RDE model uses two RNNs for encoding question text and answer text to calculate affinity among texts (Lowe et al., 2015). After encoding each part of the data, the affinity among the text pairs is calculated by using the final hidden state value of each question and answer RNNs. The matching probability between question text $\boldsymbol{w}^Q$ and answer text $\boldsymbol{w}^A$ with the training objective are as follows:

$$
\begin{aligned}
p(\text{label}) &= \sigma((h_{t_q}^Q)^T M\ h_{t_a}^A + b), \\
\mathcal{L} &= -\log \prod_{n=1}^{N} p(\text{label}_n | h_{n,t_q}^Q, h_{n,t_a}^A),
\end{aligned}
\tag{2}
$$

where $h_{t_q}^Q$ and $h_{t_a}^A$ are last hidden state of each question and answer RNN with the dimensionality $h_t \in \mathbb{R}^d$. The $M \in \mathbb{R}^{d \times d}$ and bias b are learned model parameters. The N is total number of samples used in training and σ is the sigmoid function.

3.2 Hierarchical Recurrent Dual Encoder (HRDE)

From now we explain our proposed model. The previous RDE model tries to encode the text in question or in answer with RNN architecture. It would be less effective as the length of the word sequences in the text increases because RNN's natural characteristic of forgetting information from long ranging data. To address this RNN's forgetting phenomenon, (Bahdanau et al., 2014) proposed an attention mechanism, however, we found that it still showed a limitation when we consider very large sequential length data such as 162 steps average in the Ubuntu Dialogue Corpus dataset (see Table 1). To overcome this limitation, we designed the HRDE architecture. The HRDE model divides long sequential text data into small chunk such as sentences, and encodes the whole text from word-level to chunk-level by using two hierarchical level of RNN architecture.

Figure 1 shows a diagram of the HRDE model. The word-level RNN part is responsible for encoding the words sequence $\boldsymbol{w}_c = \{w_{c,1:t}\}$ in each chunk. The chunk can be sentences in paragraph, paragraphs in essay, turns in dialogue or any kinds of smaller meaningful sub-set from the text. Then the final hidden states of each chunk will be fed into chunk-level RNN with its original sequence order kept. Therefore the chunk-level RNN can deal with pre-encoded chunk data with less sequential steps. The hidden states of the hierarchical RNNs are as follows:

$$
\begin{aligned}
h_{c,t} &= f_\theta(h_{c,t-1}, w_{c,t}), \\
u_c &= g_\theta(u_{c-1}, h_c),
\end{aligned}
\tag{3}
$$

where f_θ and g_θ are the RNN function in hierarchical architecture with weight parameters θ, $h_{c,t}$ is word-level RNN's hidden status at t-th word in c-th chunk. The $w_{c,t}$ is t-th word in c-th chunk of target question or answer text. The u_c is chunk-level RNN's hidden state at c-th chunk sequence, and h_c is word-level RNN's last hidden state of each chunk $h_c \in \{h_{1:c,t}\}$.

We use the same training objective as the RDE model, and the final matching probability between question and answer text is calculated using chunk-level RNN as follows:

$$
p(\text{label}) = \sigma((u_{c_q}^Q)^T M\ u_{c_a}^A + b), \tag{4}
$$

where $u_{c_q}^Q$ and $u_{c_a}^A$ are chunk-level RNN's last hidden state of each question and answer text with the dimensionality $u_c \in \mathbb{R}^{d^u}$, which involves the $M \in \mathbb{R}^{d^u \times d^u}$.

3.3 Latent Topic Clustering (LTC)

To learn how to rank QA pairs, a neural network should be trained to find the proper feature that represents the information within the data and fits the model parameter that can approximate the true-hypothesis. For this type of problem, we propose the LTC module for grouping the target data to help the neural network find the true-hypothesis with more information from the topic cluster in end-to-end training.

The blue-dotted box on the right-side of Figure 2 shows LTC structure diagram. To assign topic information, we build internal latent topic memory

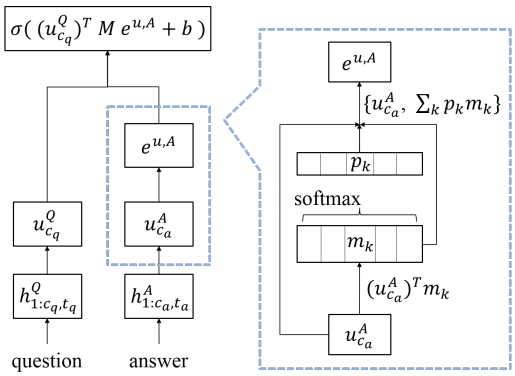


Figure 2: Diagram of the HRDE-LTC. Input vector is compared to each latent topic memory m_k to calculate cluster-info contained vector. This vector will be concatenated to original input vector.

$m \in \mathbb{R}^{d^m \times K}$, which is only model parameter to be learned, where d^m is vector dimension of each latent topic and K is number of latent topic cluster. For a given input sequence $\boldsymbol{x} = \{x_{1:t}\}$with these K vectors, we construct LTC process as follows:

$$
\begin{aligned}
p_k &= \text{softmax}((\boldsymbol{x})^T m_k), \\
\boldsymbol{x}_K &= \sum_{k=1}^{K} p_k m_k, \\
\boldsymbol{e} &= \text{concat}\{\boldsymbol{x}, \boldsymbol{x}_K\}.
\end{aligned}
\tag{5}
$$

First, the similarity between the $\boldsymbol{x}$ and each latent topic vector is calculated by dot-product. Then the resulting K values are normalized by the softmax function $\text{softmax}(z_k) = e^{z_k} / \sum_i e^{z_i}$ to produce a similarity probability p_k. After calculating the latent topic probability p_k, x_K is retrieved from summing over m_k weighted by the p_k. Then we concatenate this result with the original encoding vector to generate the final encoding vector $\boldsymbol{e}$ with the LTC information added.

Note that the input sequence of the LTC could be any type of neural network based encoding function $\boldsymbol{x} = f^{\text{enc}}_\theta(\boldsymbol{w})$ such as RNN, CNN and multilayer perceptron model (MLP). In addition, if the dimension size of $\boldsymbol{x}$ is different from that of memory vector, additional output projection layer should be placed after $\boldsymbol{x}$ before applying dot-product to the memory.

3.4 Combined Model of (H)RDE and LTC

As the LTC module extracts additional topic cluster information from the input data, we can combine this module with any neural network in their end-to-end training flow. In our experiments, we combine the LTC module with the RDE and HRDE models.

3.4.1 RDE with LTC

The RDE model encodes question and answer texts to $h^Q_{t_q}$ and $h^A_{t_a}$, respectively. Hence, the LTC module could take these vectors as the input to generate latent topic cluster information added vector $\boldsymbol{e}$. With this vector, we calculate the affinity among question and answer texts as well as additional cluster information. The following equation shows our RDE-LTC process:

$$
p(\text{label}) = \sigma((h^Q_{t_q})^T M\, e^A + b). \tag{6}
$$

In this case, we applied the LTC module only for the answer side, assuming that the answer text is longer than the question. Thus, it needs to be clustered. To train the network, we use the same training objective, to minimize cross-entropy loss, as in equation (2).

3.4.2 HRDE with LTC

The LTC can be combined with the HRDE model, in the same way it is applied to the RDE-LTC model by modifying equation (6 as follows:

$$
p(\text{label}) = \sigma((u^Q_{c_q})^T M\, e^{u,A} + b), \tag{7}
$$

where $u^Q_{c_q}$ is the final network hidden state vector of the chunk-level RNN for a question input sequence. The $e^{u,A}$ is the LTC information added vector from equation (5), where the LTC module takes the input $\boldsymbol{x} = \boldsymbol{u}^A$ from the HRDE model equation (3). The HRDE-LTC model also use the same training objective, minimizing cross-entropy loss, as in equation (2). Figure 2 shows a diagram of the combined model with the HRDE and the LTC.

4 Experimental Setup and Dataset

4.1 The Ubuntu Dialogue Corpus

The Ubuntu Dialogue Corpus has been developed by expanding and preprocessing the Ubuntu Chat Logs[1], which refer to a collection of logs from the Ubuntu-related chat room for solving problem in using the Ubuntu system by (Lowe et al., 2015).

Among the utterances in the dialogues, they consider each utterance, starting from the third one, as a potential {response} while the previous utterance is considered as a {context}. The data

[1]These logs are available from http://irclogs.ubuntu.com

Dataset	# Samples			Message (Avg.)			Response (Avg.)		
	Train	Val.	Test	# tokens	# groups	# tokens /group	# tokens	# groups	# tokens /group
Ubuntu-v1	1M	35,609	35,517	162.47 ±132.47	8.43 ±6.32	20.14 ±18.41	14.44 ±13.93	1	-
Ubuntu-v2	1M	19,560	18,920	85.92 ±74.71	4.95 ±2.98	20.73 ±20.19	17.01 ±16.41	1	-
Samsung QA	163,616	10,000	10,000	12.84 ±6.42	1	-	173.48 ±192.12	6.09 ±5.58	29.28 ±31.91

Table 1: Properties of the Ubuntu and Samsung QA dataset. The message and response are {context}, {response} in Ubuntu and {question}, {answer} in the Samsung QA dataset.

was processed extracting ({context}, {response}, flag) tuples from the dialogues.

We called this original Ubuntu dataset as Ubuntu-v1 dataset. After releasing the Ubuntu-v1 dataset, researchers published v2 version of this dataset. Main updates are separating train/valid/test dataset by time so that mimics real life implementation, where we are training a model on past data to predict future data, changing sampling procedure to increase average turns in the {context}. We consider this Ubuntu dataset is one of the best dataset in terms of its quality, quantity and availability for evaluating the performance of the text ranking model.

To encode the text with the HRDE and HRDE-LTC model, a text needs to be divided into several chunk sequences with predefined criteria. For the Ubuntu-v1 dataset case, we divide the {context} part by splitting with end-of-sentence delimiter "_eos_", and we do not split the {response} part since it is normally short and does not contain "_eos_" information. For the Ubuntu-v2 dataset case, we split the {context} part in the same way as we do in the Ubuntu-v1 dataset while only using end-of-turn delimiter "_eot_". Table 1 shows properties of the Ubuntu dataset.

Question
how do i set a timer of clock in applications and development for samsung galaxy s4 mini?

Answer
1 from within the clock application, tap timer tab. 2 tap the hours, minutes, or seconds field and use the on-screen keypad to enter the hour, minute, or seconds. the timer plays an alarm at the end of the countdown. 3 tap start to start the timer. 4 tap stop to stop the timer or reset to reset the timer and start over. 5 tap restart to resume the timer counter.

Table 2: Example of the Samsung QA dataset.

4.2 Consumer Product QA Corpus

To test the robustness of the proposed model, we introduce an additional question and answer pair dataset related to an actual user's interaction with the consumer electronic product domain. We crawled data from various sources like the Samsung Electronics' official web site[2] and crowd QA web sites[3][4] in a similar way that (Yoon et al., 2016) did in building QA system for consumer products. On the official web page, we can retrieve data consisting of user questions and matched answers like frequently asked questions and troubleshooting. From the crowd QA sites, there are many answers from various users for each question. Among these answers, we choose answers from company certificated users to keep the reliability of the answers high. If there are no such answers, we skip that question answer pair. Table 2 shows an example of question-answer pair crawled from the web page. In addition, we crawl hierarchical product category information related to QA pairs. In particular, *mobile*, *office*, *photo*, *tv/video*, *accessories*, and *home appliance* as top-level categories, and specific categories like *galaxy s7*, *tablet*, *led tv*, and *others* are used. We collected these meta-information for further use. The total size of the Samsung QA data is over 100,000 pairs and we split the data into approximately 80,000/10,000/10,000 samples to create train/valid/test sets, respectively. To create the train set, we use a QA pair sample as a ground-truth and perform negative sampling for answers among training sets to create *false*-label datasets. In this way, we generated ({question}, {answer}, flag) triples (see Table 1). We do the same procedure to create valid and test sets by only differentiating more negative sampling within each dataset to generate 9 *false*-label samples with one

[2]http://www.samsung.com/us
[3]http://answers.yahoo.com
[4]http://answers.us.samsung.com

ground-truth sample. We apply the same method in such a way that the Ubuntu dataset is generated from the Ubuntu Dialogue Corpus to maintain the consistency. The Samsung QA dataset is available via web repository. We refer the readers to Appendix A for more examples of each dataset.

4.3 Implementation Details

4.3.1 Ubuntu dataset case

To implement the RDE model, we use two single layer Gated Recurrent Unit (GRU) (Chung et al., 2014) with 300 hidden units . Each GRU is used to encode {context} and {response}, respectively. The weight for the two GRU are shared. The hidden units weight matrix of the GRU are initialized using orthogonal weights (Saxe et al., 2013), while input embedding weight matrix is initialized using a pre-trained embedding vector, the Glove (Pennington et al., 2014), with 300 dimension. The vocabulary size is 144,953 and 183,045 for the Ubuntu-v1/v2 case, respectively. We use the Adam optimizer (Kingma and Ba, 2014), with gradients clipped with norm value 1. The maximum time step for calculating gradient of the RNN is determined according to the input data statistics in Table 1.

For the HRDE model, we use two single layer GRU with 300 hidden units for word-level RNN part, and another two single layer GRU with 300 hidden units for chunk-level RNN part. The weight of the GRU is shared within the same hierarchical part, word-level and chunk-level. The other settings are the same with the RDE model case. As for the combined model with the (H)RDE and the LTC, we choose the latent topic memory dimensions as 256 in both ubuntu-v1 and ubuntu-v2. The number of the cluster in LTC module is decided to 3 for both the RDE-LTC and the HRDE-LTC cases. In HRDE-LTC case, we applied LTC module to the {context} part because we think it is longer having enough information to be clustered with. All of these hyper-parameters are selected from additional parameter searching experiments.

The dropout (Srivastava et al., 2014) is applied for the purpose of regularization with the ratio of: 0.2 for the RNN in the RDE and the RDE-LTC, 0.3 for the word-level RNN part in the HRDE and the HRDE-LTC, 0.8 for the latent topic memory in the RDE-LTC and the HRDE-LTC.

We need to mention that our implementation of the RDE module has the same architecture as the LSTM model (Kadlec et al., 2015) in ubuntu-v1/v2 experiments case. It is also the same architecture with the RNN model (Baudiš et al., 2016) in ubuntu-v2 experiment case. We implement the same model ourselves, because we need a baseline model to compare with other proposed models such as the RDE-LTC, HRDE and HRDE-LTC.

4.3.2 Samsung QA dataset case

To test the Samsung QA dataset, we use the same implementation of the model (RDE, RDE-LTC, HRDE and HRDE-LTC) used in testing the Ubuntu dataset. Only the differences are, we use 100 hidden units for the RDE and the RDE-LTC, 300 hidden units for the HRDE and 200 hidden units for the HRDE-LTC, and the vocabulary size of 28,848. As for the combined model with the (H)RDE and LTC, the dimensions of the latent topic memory is 64 and the number of latent cluster is 4. We chose best performing hyper-parameter of each model by additional extensive hyper-parameter search experiments.

All of the code developed for the empirical results are available via web repository [5].

5 Empirical Results

5.1 Evaluation Metrics

We regards all the tasks as selecting the best answer among text candidates for the given question. Following the previous work (Lowe et al., 2015), we report model performance as recall at k (R@k) relevant texts among given 2 or 10 candidates (e.g., 1 in 2 R@1). Though this metric is useful for ranking task, R@1 metric is also meaningful for classifying the best relevant text.

Each model we implement is trained multiple times (10 and 15 times for Ubuntu and the Samsung QA datasets in our experiments, respectively) with random weight initialization, which largely influences performance of neural network model. Hence we report model performance as mean and standard derivation values (Mean±Std).

5.2 Performance Evaluation

5.2.1 Comparison with other methods

As Table 3 shows, our proposed HRDE and HRDE-LTC models achieve the best performance for the Ubuntu-v1 dataset. We also find that the RDE-LTC model shows improvements from the baseline model, RDE.

[5] http://github.com/david-yoon/QA_HRDE_LTC

Model	Ubuntu-v1			
	1 in 2 R@1	1 in 10 R@1	1 in 10 R@2	1 in 10 R@5
TF-IDF [1]	0.659	0.410	0.545	0.708
CNN [2]	0.848	0.549	0.684	0.896
LSTM [2]	0.901	0.638	0.784	0.949
CompAgg [3]	0.884	0.631	0.753	0.927
BiMPM [4]	0.897	0.665	0.786	0.938
RDE	0.898 ±0.002	0.643 ±0.009	0.784 ±0.007	0.945 ±0.002
RDE-LTC	0.903 ±0.001	0.656 ±0.003	0.794 ±0.003	0.948 ±0.001
HRDE	0.915 ±0.001	0.681 ±0.001	0.820 ±0.001	0.959 ±0.001
HRDE-LTC	**0.916** ±0.001	**0.684** ±0.001	**0.822** ±0.001	**0.960** ±0.001

Table 3: Model performance results for the Ubuntu-v1 dataset. Models [1-4] are from (Lowe et al., 2015; Kadlec et al., 2015; Wang and Jiang, 2016; Wang et al., 2017), respectively.

Model	Ubuntu-v2			
	1 in 2 R@1	1 in 10 R@1	1 in 10 R@2	1 in 10 R@5
LSTM [1]	0.869	0.552	0.721	0.924
RNN [5]	0.907 ±0.002	0.664 ±0.004	0.799 ±0.004	0.951 ±0.001
CNN [5]	0.863 ±0.003	0.587 ±0.004	0.721 ±0.005	0.907 ±0.003
RNN-CNN [5]	0.911 ±0.001	**0.672** ±0.002	0.809 ±0.002	0.956 ±0.001
Attention [6] (RNN-CNN)	0.903 ±0.002	0.653 ±0.005	0.788 ±0.005	0.945 ±0.002
CompAgg [3]	0.895	0.641	0.776	0.937
BiMPM [4]	0.877	0.611	0.747	0.921
RDE	0.894 ±0.002	0.610 ±0.008	0.776 ±0.006	0.947 ±0.002
RDE-LTC	0.899 ±0.002	0.625 ±0.004	0.788 ±0.004	0.951 ±0.001
HRDE	0.914 ±0.001	0.649 ±0.001	0.813 ±0.001	0.964 ±0.001
HRDE-LTC	**0.915** ±0.002	0.652 ±0.003	**0.815** ±0.001	**0.966** ±0.001

Table 4: Model performance results for the Ubuntu-v2 dataset. Models [1,3-6] are from (Lowe et al., 2015; Wang and Jiang, 2016; Wang et al., 2017; Baudiš et al., 2016; Tan et al., 2015), respectively.

For the ubuntu-v2 dataset case, Table 4 reveals that the HRDE-LTC model is best for three cases (1 in 2 R@1, 1 in 10 R@2 and 1 in 10 R@5). Comparing the same model with our implementation (RDE) and (Baudiš et al., 2016)'s implementation (RNN), there is a large gap in the accuracy (0.610 and 0.664 of 1 in 10 R@1 for RDE and RNN, receptively). We think this is largely influenced by the data preprocessing method, because the only differences between these models is the data preprocessing, which is (Baudiš et al., 2016)'s contribution to the research. We are certain that our model performs better with the exquisite datasets which adapts extensive preprocessing method, because we see improvements from the RDE model to the HRDE model and additional improvements with the LTC module in all test cases (the Ubuntu-v1/v2 and the Samsung QA).

Model	Samsung QA			
	1 in 2 R@1	1 in 10 R@1	1 in 10 R@2	1 in 10 R@5
TF-IDF	0.939	0.834	0.897	0.953
RDE	0.978 ±0.002	0.869 ±0.009	0.966 ±0.003	0.997 ±0.001
RDE-LTC	0.981 ±0.002	0.880 ±0.009	0.970 ±0.003	0.997 ±0.001
HRDE	0.981 ±0.002	0.885 ±0.011	0.971 ±0.004	0.997 ±0.001
HRDE-LTC	**0.983** ±0.002	**0.890** ±0.010	**0.972** ±0.003	**0.998** ±0.001

Table 5: Model performance results for the Samsung QA dataset.

# clusters	Accuracy (1 in 10 R@1)		
	Ubuntu-v1	Ubuntu-v2	Samsung QA
1	0.643 ±0.009	0.610 ±0.008	0.869 ±0.009
2	0.655 ±0.005	0.616 ±0.006	0.876 ±0.011
3	**0.656** ±0.003	**0.625** ±0.004	0.877 ±0.010
4	0.651 ±0.005	0.622 ±0.005	**0.880** ±0.009

Table 6: The RDE-LTC model results with different numbers of latent clusters. "Cluster 1" is the baseline model, RDE.

In the Samsung QA case, Table 5 indicates that the proposed RDE-LTC, HRDE, and the HRDE-LTC model show performance improvements when compared to the baseline model, TF-IDF and RDE. The average accuracy statistics are higher in the Samsung QA case when compared to the Ubuntu case. We think this is due to in the smaller vocabulary size and context variety. The Samsung QA dataset deals with narrower topics than in the Ubuntu dataset case. We are certain that our proposed model shows robustness in several datasets and different vocabulary size environments.

5.2.2 Degradation Comparison for Longer Texts

To verify the HRDE model's ability compared to the baseline model RDE, we split the testset of the Ubuntu-v1/v2 datasets based on the "number of chunks" in the {context}. Then, we measured the top-1 recall (same case as 1 in 10 R@1 in Table 3, and 4) for each group. Figure 3 demonstrates that the HRDE models, in darker blue and red colors, shows better performance than the RDE models, in lighter colors, for every "number of chunks" evaluations. In particular, the HRDE models are consistent when the "number-of-chunks" increased, while the RDE models degrade as the "number-of-chunks" increased.

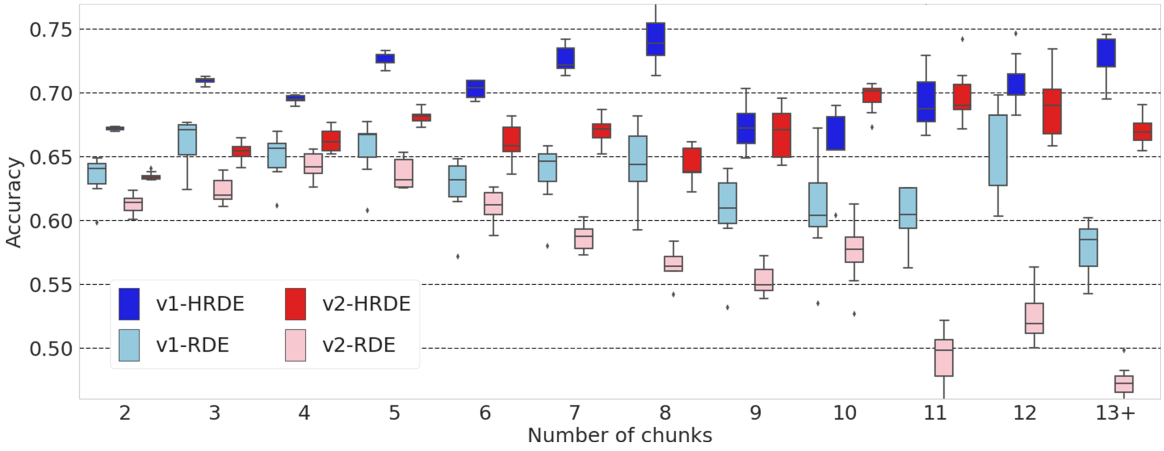


Figure 3: The HRDE and RDE model performance comparisons for the number-of-chunk in the Ubuntu dataset. Each boxplot shows average accuracy with standard deviation. The HRDE models, in darker blue and red colors, show consistent performances as the number-of-chunks increased. Meanwhile, the RDE models in lighter colors show performance degradation as the number-of-chunks increased. Furthermore, 13+ indicates all data over 13-chunks.

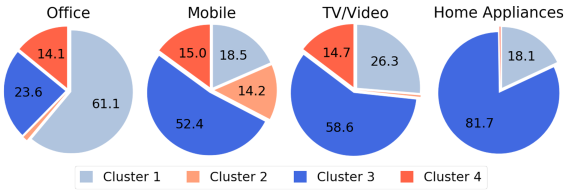


Figure 4: Examples of the cluster proportions for four real *categories* from $20k$ evaluated samples. Each color corresponds to each cluster.

Cluster	Example
1	How to adjust the brightness on the s**d300 series monitors
2	How do I reject an incoming call on my Samsung Galaxy Note 3?
3	How should I clean and maintain the microwave?
4	How do I connnect my surround sound to this TV and what type of cables do I need

Table 7: Example sentences for each cluster.

5.2.3 Effects of the LTC Numbers

We analyze the RDE-LTC model for different numbers of latent clusters. Table 6 indicates that the model performances increase as the number of latent clusters increase (until 3 for the Ubuntu and 4 for the Samsung QA case). This is probably a major reason for the different number of subjects in each dataset. The Samsung QA dataset has an internal *category* related to the type of consumer electronic products (6 top-level *categories*; *mobile*, *office*, *photo*, *tv/video*, *accessories*, and *home appliance*), so that the LTC module makes clusters these *categories*. The Ubuntu dataset, however, has diverse contents related to issues in using the Ubuntu system. Thus, the LTC module has fewer clusters with the sparse topic compared to the Samsung QA dataset.

5.2.4 Comprehensive Analysis of LTC

We conduct quantitative and qualitative analysis on the HRDE-LTC model for four latent topic clusters. The Samsung QA dataset has *category* information; hence, latent topic clustering results can be compared with real *categories*. We randomly choose $20k$ samples containing real *category* information and evaluate each sample with the HRDE-LTC model. The cluster with the highest similarity among the latent topic clusters is considered a representative cluster of each sample.

Figure 4 shows proportion of four latent clusters among these samples according to real *category* information. Even though the HRDE-LTC model is trained without any ground-truth *category* labels, we observed that the latent cluster is formed accordingly. For instance, cluster 2 is shown mostly in "Mobile" *category* samples while "clusters 2 and 4" are rarely shown in "Home Appliance" *category* samples.

Additionally, we explore sentences with higher similarity score from the HRDE-LTC module for each four cluster. As can be seen in Table 7, "cluster 1" contains "screen" related sentences (e.g., brightness, pixel, display type) while "cluster 2" contains sentences with exclusive information re-

lated to the "Mobile" *category* (e.g., call rejection, voice level). This qualitative analysis explains why "cluster 2" is shown mostly in the "Mobile" *category* in Figure 4. We also discover that "cluster 3" has the largest portion of samples. As "cluster 3" contains "security" and "maintenance" related sentences (e.g., password, security, log-on, maintain), we assume that this is one of the frequently asked issues across all *categories* in the Samsung QA dataset. Table 7 shows example sentences with high scores from each cluster.

6 Conclusion

In this paper, we proposed the HRDE model and LTC module. HRDE showed higher performances in ranking answer candidates and less performance degradations when dealing with longer texts compared to conventional models. The LTC module provided additional performance improvements when combined with both RDE and HRDE models, as it added latent topic cluster information according to dataset properties. With this proposed model, we achieved state-of-the-art performances in Ubuntu datasets. We also evaluated our model in real world question answering dataset, Samsung QA. This demonstrated the robustness of the proposed model with the best results.

Acknowledgments

K. Jung is with the Department of Electrical and Computer Engineering, ASRI, Seoul National University, Seoul, Korea. This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2016M3C4A7952587), the Ministry of Trade, Industry & Energy (MOTIE, Korea) under Industrial Technology Innovation Program (No.10073144).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Petr Baudiš, Jan Pichl, Tomáš Vyskočil, and Jan Šedivỳ. 2016. Sentence pair scoring: Towards unified framework for text comprehension. *arXiv preprint arXiv:1603.06127* .

Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014. Open question answering with weakly supervised embedding models. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, pages 165–180.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326* .

Jennifer Chu-Carroll, James Fan, BK Boguraev, David Carmel, Dafna Sheinwald, and Chris Welty. 2012. Finding needles in the haystack: Search and candidate generation. *IBM Journal of Research and Development* 56(3.4):6–1.

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

James Fan, Aditya Kalyanpur, David C Gondek, and David A Ferrucci. 2012. Automatic knowledge extraction from documents. *IBM Journal of Research and Development* 56(3.4):5–1.

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

Rudolf Kadlec, Martin Schmid, and Jan Kleindienst. 2015. Improved deep learning baselines for ubuntu corpus dialogs. *arXiv preprint arXiv:1510.03753* .

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*. pages 1378–1387.

Ryan Lowe, Nissan Pow, Iulian V Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. page 285.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pages 1532–1543.

Juan Ramos et al. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*. volume 242, pages 133–142.

Andrew M Saxe, James L McClelland, and Surya Ganguli. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120* .

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*. pages 3776–3784.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, pages 553–562.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research* 15(1):1929–1958.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*. pages 2440–2448.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Lstm-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108* .

Mengqiu Wang, Noah A Smith, and Teruko Mitamura. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Shuohang Wang and Jing Jiang. 2016. A compare-aggregate model for matching text sequences. *arXiv preprint arXiv:1611.01747* .

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. AAAI Press, pages 4144–4150.

Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In *International Conference on Machine Learning*. pages 2397–2406.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 2013–2018.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association of Computational Linguistics* 4(1):259–272.

Seunghyun Yoon, Mohan Sundar, Abhishek Gupta, and Kyomin Jung. 2016. Automatic question answering system for consumer products. In *Proceedings of SAI Intelligent Systems Conference*. Springer, pages 1012–1016.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632* .

A More examples of the dataset

A.1 Ubuntu dataset

Question
"what will happend if i unmounted the ubuntu partition", "it will unmount , unless it is in use", "srr i did n't got it"

Answer	*flag*
"you cannot unmount a partition if it is currently in use"	1
"why do you not have a backup if the data is important ?"	0

Table 8: Example of the Ubuntu-v2 dataset.

A.2 Samsung QA dataset

Question
how can i place the current call on hold at any point during a conversation ?

Answer	*flag*
you can place the current call on hold at any point during a conversation . you can also make another call while you have a call in progress if your network supports this service . 1 while on a call , tap hold . this action places the current caller on hold . 2 you can later reactivate this call by tapping unhold .	1
please try to do a soft reset . turn of the phone , remove and put the battery back after 1-2 minutes . we also recommend you to clear the data of the samsung keyboard . 1 from the home screen , touch application 2 select settings 3 select application manager 4 touch the all tab 5 select samsung keyboard 6 tap on clear data .	0

Table 9: Example of the Samsung QA dataset.

Supervised and Unsupervised Transfer Learning for Question Answering

Yu-An Chung[1] **Hung-Yi Lee**[2] **James Glass**[1]
[1]MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, USA
{andyyuan,glass}@mit.edu

[2]Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan
hungyilee@ntu.edu.tw

Abstract

Although transfer learning has been shown to be successful for tasks like object and speech recognition, its applicability to question answering (QA) has yet to be well-studied. In this paper, we conduct extensive experiments to investigate the transferability of knowledge learned from a source QA dataset to a target dataset using two QA models. The performance of both models on a TOEFL listening comprehension test (Tseng et al., 2016) and MCTest (Richardson et al., 2013) is significantly improved via a simple transfer learning technique from MovieQA (Tapaswi et al., 2016). In particular, one of the models achieves the state-of-the-art on all target datasets; for the TOEFL listening comprehension test, it outperforms the previous best model by 7%. Finally, we show that transfer learning is helpful even in unsupervised scenarios when correct answers for target QA dataset examples are not available.

1 Introduction

1.1 Question Answering

One of the most important characteristics of an intelligent system is to understand stories like humans do. A story is a sequence of sentences, and can be in the form of plain text (Trischler et al., 2017; Rajpurkar et al., 2016; Weston et al., 2016; Yang et al., 2015) or spoken content (Tseng et al., 2016), where the latter usually requires the spoken content to be first transcribed into text by automatic speech recognition (ASR), and the model will subsequently process the ASR output. To evaluate the extent of the model's understanding of the story, it is asked to answer questions about the story. Such a task is referred to as question answering (QA), and has been a long-standing yet challenging problem in natural language processing (NLP).

Several QA scenarios and datasets have been introduced over the past few years. These scenarios differ from each other in various ways, including the length of the story, the format of the answer, and the size of the training set. In this work, we focus on context-aware multi-choice QA, where the answer to each question can be obtained by referring to its accompanying story, and each question comes with a set of answer choices with only one correct answer. The answer choices are in the form of open, natural language sentences. To correctly answer the question, the model is required to understand and reason about the relationship between the sentences in the story.

1.2 Transfer Learning

Transfer learning (Pan and Yang, 2010) is a vital machine learning technique that aims to use the knowledge learned from one task and apply it to a different, but related, task in order to either reduce the necessary fine-tuning data size or improve performance. Transfer learning, also known as domain adaptation[1], has achieved success in numerous domains such as computer vision (Sharif Razavian et al., 2014), ASR (Doulaty et al., 2015; Huang et al., 2013), and NLP (Zhang et al., 2017; Mou et al., 2016). In computer vision, deep neural networks trained on a large-scale image classification dataset such as ImageNet (Russakovsky et al., 2015) have proven to be excellent feature extractors for a broad range of visual tasks such as image captioning (Lu et al., 2017; Karpathy and Fei-Fei, 2015; Fang et al., 2015) and visual

[1]In this paper, we do not distinguish conceptually between transfer learning and domain adaptation. A 'domain' in the sense we use throughout this paper is defined by datasets.

Proceedings of NAACL-HLT 2018, pages 1585–1594
New Orleans, Louisiana, June 1 - 6, 2018.

question answering (Xu and Saenko, 2016; Fukui et al., 2016; Yang et al., 2016; Antol et al., 2015), among others. In NLP, transfer learning has also been successfully applied to tasks like sequence tagging (Yang et al., 2017), syntactic parsing (McClosky et al., 2010) and named entity recognition (Chiticariu et al., 2010), among others.

1.3 Transfer Learning for QA

Although transfer learning has been successfully applied to various applications, its applicability to QA has yet to be well-studied. In this paper, we tackle the TOEFL listening comprehension test (Tseng et al., 2016) and MCTest (Richardson et al., 2013) with transfer learning from MovieQA (Tapaswi et al., 2016) using two existing QA models. Both models are pre-trained on MovieQA and then fine-tuned on each target dataset, so that their performance on the two target datasets are significantly improved. In particular, one of the models achieves the state-of-the-art on all target datasets; for the TOEFL listening comprehension test, it outperforms the previous best model by 7%.

Transfer learning without any labeled data from the target domain is referred to as unsupervised transfer learning. Motivated by the success of unsupervised transfer learning for speaker adaptation (Chen et al., 2011; Wallace et al., 2009) and spoken document summarization (Lee et al., 2013), we further investigate whether unsupervised transfer learning is feasible for QA.

Although not well studied in general, transfer Learning for QA has been explored recently. To the best of our knowledge, Kadlec et al. (2016) is the first work that attempted to apply transfer learning for machine comprehension. The authors showed only limited transfer between two QA tasks, but the transferred system was still significantly better than a random baseline. Wiese et al. (2017) tackled a more specific task of biomedical QA with transfer learning from a large-scale dataset. The work most similar to ours is by Min et al. (2017), where the authors used a simple transfer learning technique and achieved significantly better performance. However, none of these works study unsupervised transfer learning, which is especially crucial when the target dataset is small. Golub et al. (2017) proposed a two-stage synthesis network that can generate synthetic questions and answers to augment insufficient training data without annotations. In this work, we aim to handle the case that the questions from the target domain are available.

2 Task Descriptions and Approaches

Among several existing QA settings, in this work we focus on multi-choice QA (MCQA). We are particularly interested in understanding whether a QA model can perform better on one MCQA dataset with knowledge transferred from another MCQA dataset. In Section 2.1, we first formalize the task of MCQA. We then describe the procedures for transfer learning from one dataset to another in Section 2.2. We consider two kinds of settings for transfer learning in this paper, one is supervised and the other is unsupervised.

2.1 Multi-Choices QA

In MCQA, the inputs to the model are a story, a question, and several answer choices. The story, denoted by $\mathbf{S}$, is a list of sentences, where each of the sentences is a sequence of words from a vocabulary set V. The question and each of the answer choices, denoted by $\mathbf{Q}$ and $\mathbf{C}$, are both single sentences also composed of words from V. The QA model aims to choose one correct answer from multiple answer choices based on the information provided in $\mathbf{S}$ and $\mathbf{Q}$.

2.2 Transfer Learning

The procedure of transfer learning in this work is straightforward and includes two steps. The first step is to pre-train the model on one MCQA dataset referred to as the **source** task, which usually contains abundant training data. The second step is to fine-tune the same model on the other MCQA dataset, which is referred to as the **target** task, that we actually care about, but that usually contains much less training data. The effectiveness of transfer learning is evaluated by the model's performance on the target task.

Supervised Transfer Learning

In supervised transfer learning, both the source and target datasets provide the correct answer to each question during pre-training and fine-tuning, and the QA model is guided by the correct answer to optimize its objective function in a supervised manner in both stages.

Unsupervised Transfer Learning

We also consider unsupervised transfer learning where the correct answer to each question in the target dataset is not available. In other words, the entire process is supervised during pre-training, but unsupervised during fine-tuning. A self-labeling technique inspired by Lee et al. (2013); Chen et al. (2011); Wallace et al. (2009) is used during fine-tuning on the target dataset. We present the proposed algorithm for unsupervised transfer learning in Algorithm 1.

Algorithm 1 Unsupervised QA Transfer Learning

Input: Source dataset with correct answer to each question; Target dataset without any answer; Number of training epochs.

Output: Optimal QA model M^*

1: Pre-train QA model M on the source dataset.
2: **repeat**
3: For each question in the target dataset, use M to predict its answer.
4: For each question, assign the predicted answer to the question as the correct one.
5: Fine-tune M on the target dataset as usual.
6: **until** Reach the number of training epochs.

3 Datasets

We used MovieQA (Tapaswi et al., 2016) as the source MCQA dataset, and TOEFL listening comprehension test (Tseng et al., 2016) and MCTest (Richardson et al., 2013) as two separate target datasets. Examples of the three datasets are shown in Table 1.

MovieQA is a dataset that aims to evaluate automatic story comprehension from both video and text. The dataset provides multiple sources of information such as plot synopses, scripts, subtitles, and video clips that can be used to infer answers. We only used the plot synopses of the dataset, so our setting is the same as pure textual MCQA. The dataset contains 9,848/1,958 train/dev examples; each question comes with a set of five possible answer choices with only one correct answer.

TOEFL listening comprehension test is a recently published, very challenging MCQA dataset that contains 717/124/122 train/dev/test examples. It aims to test knowledge and skills of academic English for global English learners whose native languages are not English. There are only four answer choices for each question. The stories in this dataset are in audio form. Each story comes with two transcripts: manual and ASR transcriptions, where the latter is obtained by running the CMU Sphinx recognizer (Walker et al., 2004) on the original audio files. We use TOEFL-manual and TOEFL-ASR to denote the two versions, respectively. We highlight that the questions in this dataset are not easy because most of the answers cannot be found by simply matching the question and the choices without understanding the story. For example, there are questions regarding the gist of the story or the conclusion for the conversation.

MCTest is a collection of 660 elementary-level children's stories. Each question comes with a set of four answer choices. There are two variants in this dataset: MC160 and MC500. The former contains 280/120/240 train/dev/test examples, while the latter contains 1,200/200/600 train/dev/test examples and is considered more difficult.

The two chosen target datasets are challenging because the stories and questions are complicated, and only small training sets are available. Therefore, it is difficult to train statistical models on only their training sets because the small size limits the number of parameters in the models, and prevents learning any complex language concepts simultaneously with the capacity to answer questions. We demonstrate that we can effectively overcome these difficulties via transfer learning in Section 5.

4 QA Neural Network Models

Among numerous models proposed for multiple-choice QA (Trischler et al., 2016; Fang et al., 2016; Tseng et al., 2016), we adopt the End-to-End Memory Network (MemN2N)[2] (Sukhbaatar et al., 2015) and Query-Based Attention CNN (QACNN)[3] (Liu et al., 2017), both open-sourced, to conduct the experiments. Below we briefly introduce the two models in Section 4.1 and Section 4.2, respectively. For the details of the models, please refer to the original papers.

4.1 End-to-End Memory Networks

An End-to-End Memory Network (MemN2N) first transforms **Q** into a vector representation with

[2]MemN2N was originally designed to output a single word within a fixed vocabulary set. To apply it to MCQA, some modification is needed. We describe the modifications in Section 4.1.

[3]https://github.com/chun5212021202/ACM-Net

	Source Dataset	Target Dataset	
	MovieQA	TOEFL	MCTest
$\mathbf{S}$	After entering the boathouse, the trio witness Voldemort telling Snape that the elder Wand cannot serve Voldemort until Snape dies ... Before dying, Snape tells Harry to take his memories to the Pensieve ...	I just wanted to take a few minutes to meet with everyone to make sure your class presentations for next week are all in order and coming along well. And as you know, you're supposed to report on some areas of recent research on genetics ...	James the Turtle was always getting in trouble. Sometimes he'd reach into the freezer and empty out all the food ... Then he walked to the fast food restaurant and ordered 15 bags of fries. He didn't pay, and instead headed home ...
$\mathbf{Q}$	What does Snape tell Harry before he dies?	Why does the professor meet with the student?	What did James do after he ordered the fries?
$\mathbf{C}_1$	To bury him in the forest	To find out if the student is interested in taking part in a genetics project	went to the grocery store
$\mathbf{C}_2$	That he always respected him	To discuss the student's experiment on the taste perception	**went home without paying**
$\mathbf{C}_3$	To remember to him for the good deeds	**To determine if the student has selected an appropriate topic for his class project**	ate them
$\mathbf{C}_4$	**To take his memories to the Pensieve**	To explain what the student should focus on for his class presentation	made up his mind to be a better turtle
$\mathbf{C}_5$	To write down his memories in a book		

Table 1: Example of the story-question-choices triplet from MovieQA, TOEFL listening comprehension test, and MCTest datasets. $\mathbf{S}, \mathbf{Q}$, and $\mathbf{C}_i$ denote the story, question, and one of the answer choices, respectively. For MovieQA, each question comes with five answer choices; and for TOEFL and MCTest, each question comes with only four answer choices. The correct answer is marked in bold.

an embedding layer B. At the same time, all sentences in $\mathbf{S}$ are also transformed into two different sentence representations with two additional embedding layers A and C. The first sentence representation is used in conjunction with the question representation to produce an attention-like mechanism that outputs the similarity between each sentence in $\mathbf{S}$ and $\mathbf{Q}$. The similarity is then used to weight the second sentence representation. We then obtain the sum of the question representation and the weighted sentence representations over $\mathbf{S}$ as $\mathbf{Q}'$. In the original MemN2N, $\mathbf{Q}'$ is decoded to provide the estimation of the probability of being an answer for each word within a fixed set. The word with the highest probability is then selected as the answer. However, in multiple-choice QA, $\mathbf{C}$ is in the form of open, natural language sentences instead of a single word. Hence we modify MemN2N by adding an embedding layer F to encode $\mathbf{C}$ as a vector representation $\mathbf{C}'$ by averaging the embeddings of words in $\mathbf{C}$. We then compute the similarity between each choice representation $\mathbf{C}'$ and $\mathbf{Q}'$. The choice $\mathbf{C}$ with the highest probability is then selected as the answer.

4.2 Query-Based Attention CNN

A Query-Based Attention CNN (QACNN) first uses an embedding layer E to transform $\mathbf{S}, \mathbf{Q}$, and $\mathbf{C}$ into a word embedding. Then a compare layer generates a story-question similarity map $\mathbf{SQ}$ and a story-choice similarity map $\mathbf{SC}$. The two similarity maps are then passed into a two-stage CNN architecture, where a question-based attention mechanism on the basis of $\mathbf{SQ}$ is applied to each of the two stages. The first stage CNN generates a word-level attention map for each sentence in $\mathbf{S}$, which is then fed into the second stage CNN to generate a sentence-level attention map, and yield choice-answer features for each of the choices. Finally, a classifier that consists of two fully-connected layers collects the information from every choice answer feature and outputs the most likely answer. The trainable parameters are the embedding layer E that transforms $\mathbf{S}, \mathbf{Q}$, and $\mathbf{C}$ into word embeddings, the two-stage CNN $W_{CNN}^{(1)}$ and $W_{CNN}^{(2)}$ that integrate information from the word to the sentence level, and from the sentence to the story level, and the two fully-connected layers $W_{FC}^{(1)}$ and $W_{FC}^{(2)}$ that make the final prediction. We mention the trainable parameters here because in Section 5 we will conduct experiments to analyze the transferability of the QACNN by fine-tuning some parameters while keeping others fixed. Since QACNN is a newly proposed QA model has a relatively complex structure, we illustrate its architecture in Figure 1, which is enough for understanding the rest of the paper. Please refer to the original paper (Liu et al., 2017) for more details.

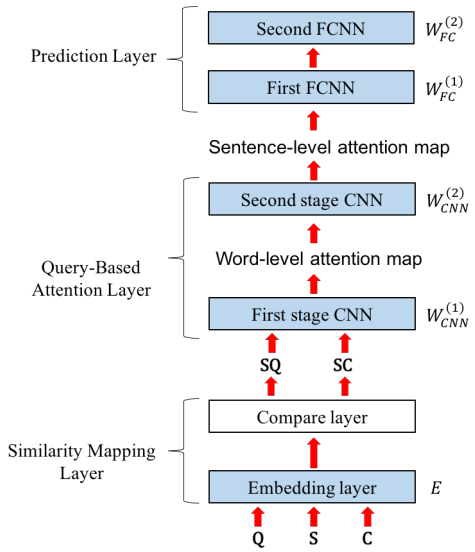


Figure 1: QACNN architecture overview. QACNN consists of a similarity mapping layer, a query-based attention layer, and a prediction layer. The two-stage attention mechanism takes place in the query-based attention layer, yielding word-level and sentence-level attention map, respectively. The trainable parameters, including $E, W_{CNN}^{(1)}, W_{CNN}^{(2)}, W_{FC}^{(1)}$, and $W_{FC}^{(2)}$, are colored in light blue.

5 Question Answering Experiments

5.1 Training Details

For pre-training MemN2N and QACNN on MovieQA, we followed the exact same procedure as in Tapaswi et al. (2016) and Liu et al. (2017), respectively. Each model was trained on the training set of the MovieQA task and tuned on the dev set, and the best performing models on the dev set were later fine-tuned on the target dataset. During fine-tuning, the model was also trained on the training set of target datasets and tuned on the dev set, and the performance on the testing set of the target datasets was reported as the final result. We use accuracy as the performance measurement.

5.2 Supervised Transfer Learning

Experimental Results

Table 2 reports the results of our transfer learning on TOEFL-manual, TOEFL-ASR, MC160, and MC500, as well as the performance of the previous best models and several ablations that did not use pre-training or fine-tuning. From Table 2, we have the following observations.

Model	Training	TOEFL		MCTest	
		manual	ASR	MC160	MC500
QACNN	(a) Target Only	48.9	47.5	57.5	56.4
	(b) Source Only	51.2	49.2	68.1	61.5
	(c) Source + Target	52.5	49.7	72.1	64.6
	(d) Fine-tuned (1)	53.4 (4.5)	51.5 (4.0)	**76.4** (18.9)	68.7 (12.3)
	(e) Fine-tuned (2)	**56.1** (7.2)	**55.3** (7.8)	73.8 (16.3)	**72.3** (15.9)
	(f) Fine-tuned (all)	56.0 (7.1)	55.1 (7.6)	69.3 (11.8)	67.7 (11.3)
MemN2N	(g) Target Only	45.2	44.4	57.2	53.6
	(h) Source Only	43.7	41.9	56.8	52.3
	(i) Source + Target	46.8	45.7	60.4	56.9
	(j) Fine-tuned	48.6 (3.4)	46.6 (2.2)	66.7 (9.5)	62.8 (9.2)
Fang et al. (2016)		49.1	48.8	-	-
Trischler et al. (2016)		-	-	74.6	71.0
Wang et al. (2015)		-	-	75.3	69.9

Table 2: Results of transfer learning on the target datasets. The number in the parenthesis indicates the accuracy increased via transfer learning (compared to rows (a) and (g)). The best performance for each target dataset is marked in bold. We also include the results of the previous best performing models on the target datasets in the last three rows.

Transfer learning helps. Rows (a) and (g) show the respective results when the QACNN and MemN2N are trained directly on the target datasets without pre-training on MovieQA. Rows (b) and (h) show results when the models are trained only on the MovieQA data. Rows (c) and (i) show results when the models are trained on both MovieQA and each of the four target datasets, and tested on the respective target dataset. We observe that the results achieved in (a), (b), (c), (g), (h), and (i) are worse than their fine-tuned counterparts (d), (e), (f), and (j). Through transfer learning, both QACNN and MemN2N perform better on all the target datasets. For example, QACNN only achieves 57.5% accuracy on MC160 without pre-training on MovieQA, but the accuracy increases by 18.9% with pre-training (rows (d) vs. (a)). In addition, with transfer learning, QACNN outperforms the previous best models on TOEFL-manual by 7%, TOEFL-ASR (Fang et al., 2016) by 6.5%, MC160 (Wang et al., 2015) by 1.1%, and MC500 (Trischler et al., 2016) by 1.3%, and becomes the state-of-the-art on all target datasets.

Which QACNN parameters to transfer? For the QACNN, the training parameters are $E, W_{CNN}^{(1)}, W_{CNN}^{(2)}, W_{FC}^{(1)}$, and $W_{FC}^{(2)}$ (Section 4.2). To better understand how transfer learning affects the performance of QACNN, we

also report the results of keeping some parameters fixed and only fine-tuning other parameters. We choose to fine-tune either only the last fully-connected layer $W_{FC}^{(2)}$ while keeping other parameters fixed (row (d) in Table 2), the last two fully-connected layers $W_{FC}^{(1)}$ and $W_{FC}^{(2)}$ (row (e)), and the entire QACNN (row (f)). For TOEFL-manual, TOEFL-ASR, and MC500, QACNN performs the best when only the last two fully-connected layers were fine-tuned; for MC160, it performs the best when only the last fully-connected layer was fine-tuned. Note that for training the QACNN, we followed the same procedure as in Liu et al. (2017), whereby pre-trained GloVe word vectors (Pennington et al., 2014) were used to initialize the embedding layer, which were not updated during training. Thus, the embedding layer does not depend on the training set, and the effective vocabularies are the same.

Fine-tuning the entire model is not always best. It is interesting to see that fine-tuning the entire QACNN doesn't necessarily produce the best result. For MC500, the accuracy of QACNN drops by 4.6% compared to just fine-tuning the last two fully-connected layers (rows (f) vs. (e)). We conjecture that this is due to the amount of training data of the target datasets - when the training set of the target dataset is too small, fine-tuning all the parameters of a complex model like QACNN may result in overfitting. This discovery aligns with other domains where transfer learning is well-studied such as object recognition (Yosinski et al., 2014).

A large quantity of mismatched training examples is better than a small training set. We expected to see that a MemN2N, when trained directly on the target dataset without pre-training on MovieQA, would outperform a MemN2N pre-trained on MovieQA without fine-tuning on the target dataset (rows (g) vs. (h)), since the model is evaluated on the target dataset. However, for the QACNN this is surprisingly not the case - QACNN pre-trained on MovieQA without fine-tuning on the target dataset outperforms QACNN trained directly on the target dataset without pre-training on MovieQA (rows (b) vs. (a)). We attribute this to the limited size of the target dataset and the complex structure of the QACNN.

Percentage of the target dataset used for fine-tuning	TOEFL		MCTest	
	manual	ASR	MC160	MC500
0	51.2	49.2	68.1	61.5
25%	53.9 (2.7)	52.3 (3.1)	70.3 (2.2)	65.6 (4.1)
50%	54.8 (0.9)	54.4 (2.1)	71.9 (1.6)	68.0 (2.4)
75%	55.3 (0.5)	54.8 (0.4)	72.5 (0.6)	71.1 (3.1)
100%	56.0 (0.7)	55.1 (0.3)	73.8 (1.3)	72.3 (1.2)

Table 3: Results of varying sizes of the target datasets used for fine-tuning QACNN. The number in the parenthesis indicates the accuracy increases from using the previous percentage for fine-tuning to the current percentage.

Varying the fine-tuning data size

We conducted experiments to study the relationship between the amount of training data from the target dataset for fine-tuning the model and the performance. We first pre-train the models on MovieQA, then vary the training data size of the target dataset used to fine-tune them. Note that for QACNN, we only fine-tune the last two fully-connected layers instead of the entire model, since doing so usually produces the best performance according to Table 2. The results are shown in Table 3[4]. As expected, the more training data is used for fine-tuning, the better the model's performance is. We also observe that the extent of improvement from using 0% to 25% of target training data is consistently larger than using from 25% to 50%, 50% to 75%, and 75% to 100%. Using the QACNN fine-tuned on TOEFL-manual as an example, the accuracy of the QACNN improves by 2.7% when varying the training size from 0% to 25%, but only improves by 0.9%, 0.5%, and 0.7% when varying the training size from 25% to 50%, 50% to 75%, and 75% to 100%, respectively.

Varying the pre-training data size

We also vary the size of MovieQA for pre-training to study how large the source dataset should be to make transfer learning feasible. The results are shown in Table 4. We find that even a small amount of source data can help. For example, by using only 25% of MovieQA for pre-training, the accuracy increases 6.3% on MC160. This is because 25% of MovieQA training set (2,462 examples) is still much larger than the MC160 training set (280 examples). As the size of the source dataset increases, the performance of QACNN continues to improve.

[4]We only include the results of QACNN in Table 3, but the results of MemN2N are very similar to QACNN.

Percentage of MovieQA used for pre-training	TOEFL		MCTest	
	manual	ASR	MC160	MC500
0	48.9	47.6	57.5	56.4
25%	51.7 (2.8)	50.7 (3.1)	63.8 (6.3)	62.4 (6.0)
50%	53.5 (1.8)	52.3 (1.6)	67.3 (3.5)	66.7 (4.3)
75%	54.8 (1.3)	54.6 (2.3)	71.2 (3.9)	70.2 (3.5)
100%	56.0 (1.2)	55.1 (0.5)	73.8 (2.6)	72.3 (2.1)

Table 4: Results of varying sizes of the MovieQA used for pre-training QACNN. The number in the parenthesis indicates the accuracy increases from using the previous percentage for pre-training to the current percentage.

Analysis of the Questions Types

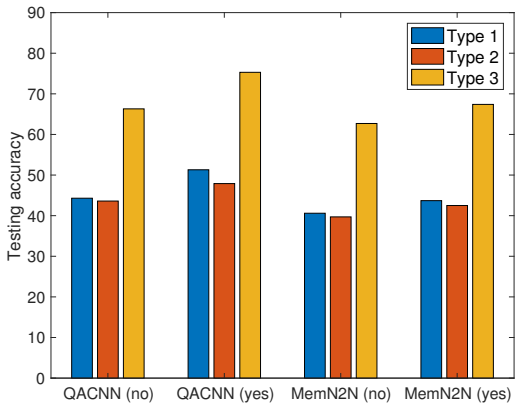
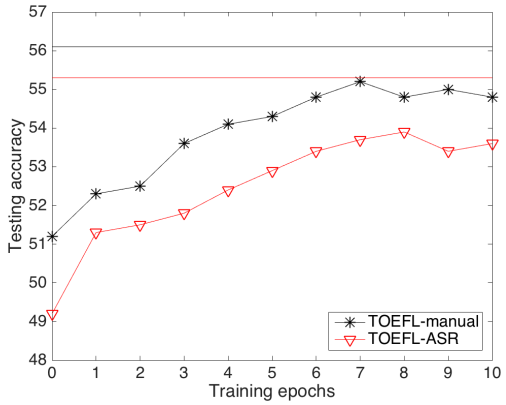


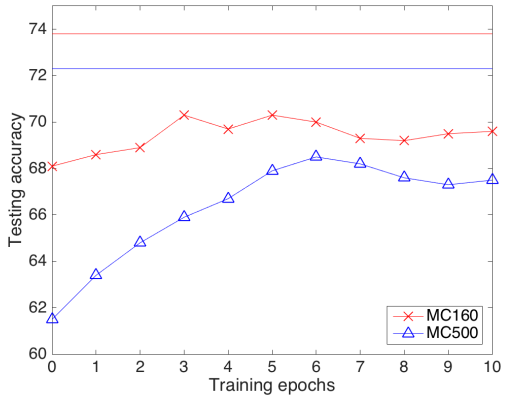
Figure 2: The performance of QACNN and MemN2N on different types of questions in TOEFL-manual with and without pre-training on MovieQA. 'No' in the parenthesis indicates the models are not pre-trained, while 'Yes' indicates the models are pre-trained on MovieQA.

We are interested in understanding what types of questions benefit the most from transfer learning. According to the official guide to the TOEFL test, the questions in TOEFL can be divided into 3 types. Type 1 questions are for basic comprehension of the story. Type 2 questions go beyond basic comprehension, but test the understanding of the functions of utterances or the attitude the speaker expresses. Type 3 questions further require the ability of making connections between different parts of the story, making inferences, drawing conclusions, or forming generalizations. We used the split provided by Fang et al. (2016), which contains 70/18/34 Type 1/2/3 questions. We compare the performance of the QACNN and MemN2N on different types of questions in TOEFL-manual with and without pre-training on MovieQA, and show the results in Figure 2. From Figure 2 we can observe that for both the QACNN and MemN2N, their performance on all three types of questions improves after pre-training, showing that the effectiveness of transfer learning is not limited to specific types of questions.

5.3 Unsupervised Transfer Learning



(a) Results of TOEFL-manual and TOEFL-ASR



(b) Results of MC160 and MC500

Figure 3: The figures show the results of unsupervised transfer learning. The x-axis is the number of training epochs, and the y-axis is the corresponding testing accuracy on the target dataset. When training epoch = 0, the performance of QACNN is equivalent to row (b) in Table 2. The horizontal lines, where each line has the same color to its unsupervised counterpart, are the performances of QACNN with supervised transfer learning (row (e) in Table 2), and are the upperbounds for unsupervised transfer learning.

So far, we have studied the property of supervised transfer learning for QA, which means

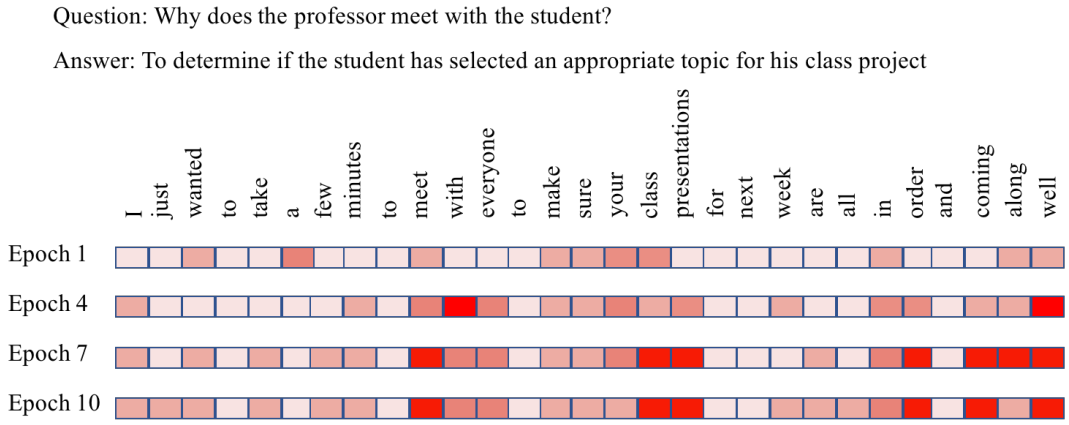


Figure 4: Visualization of the changes of the word-level attention map in the first stage CNN of QACNN in different training epochs. The more red, the more the QACNN views the word as a key feature. The input story-question-choices triplet is same as the one in Table 1.

that during pre-training and fine-tuning, both the source and target datasets provide the correct answer for each question. We now conduct unsupervised transfer learning experiments described in Section 2.2 (Algorithm 1), where the answers to the questions in the target dataset are not available. We used QACNN as the QA model and all the parameters ($E, W^{(1)}_{CNN}, W^{(2)}_{CNN}, W^{(1)}_{FC}$, and $W^{(2)}_{FC}$) were updated during fine-tuning in this experiment. Since the range of the testing accuracy of the TOEFL-series (TOEFL-manual and TOEFL-ASR) is different from that of MCTest (MC160 and MC500), their results are displayed separately in Figure 3(a) and Figure 3(b), respectively.

Experimental Results

From Figure 3(a) and Figure 3(b) we can observe that without ground truth in the target dataset for supervised fine-tuning, transfer learning from a source dataset can still improve the performance through a simple iterative self-labeling mechanism. For TOEFL-manual and TOEFL-ASR, QACNN achieves the highest testing accuracy at Epoch 7 and 8, outperforming its counterpart without fine-tuning by approximately 4% and 5%, respectively. For MC160 and MC500, the QACNN achieves the peak at Epoch 3 and 6, outperforming its counterpart without fine-tuning by about 2% and 6%, respectively. The results also show that the performance of unsupervised transfer learning is still worse than supervised transfer learning, which is not surprising, but the effectiveness of unsupervised transfer learning when no ground truth labels are provided is validated.

Attention Maps Visualization

To better understand the unsupervised transfer learning process of QACNN, we visualize the changes of the word-level attention map during training Epoch 1, 4, 7, and 10 in Figure 4. We use the same question from TOEFL-manual as shown in Table 1 as an example. From Figure 4 we can observe that as the training epochs increase, the QACNN focuses more on the context in the story that is related to the question and the correct answer choice. For example, the correct answer is related to "class project". In Epoch 1 and 4, the model does not focus on the phrase "class representation", but the model attends on the phrase in Epoch 7 and 10. This demonstrates that even without ground truth, the iterative process in Algorithm 1 is still able to lead the QA model to gradually focus more on the important part of the story for answering the question.

6 Conclusion and Future Work

In this paper we demonstrate that a simple transfer learning technique can be very useful for the task of multi-choice question answering. We use a QACNN and a MemN2N as QA models, with MovieQA as the source task and a TOEFL listening comprehension test and MCTest as the target tasks. By pre-training on MovieQA, the performance of both models on the target datasets improves significantly. The models also require much less training data from the target dataset to achieve similar performance to those without pre-training. We also conduct experiments to study the influence of transfer learning on different types

of questions, and show that the effectiveness of transfer learning is not limited to specific types of questions. Finally, we show that by a simple iterative self-labeling technique, transfer learning is still useful, even when the correct answers for target QA dataset examples are not available, through quantitative results and visual analysis.

One area of future research will be generalizing the transfer learning results presented in this paper to other QA models and datasets. In addition, since the original data format of the TOEFL listening comprehension test is audio instead of text, it is worth trying to initialize the embedding layer of the QACNN with semantic or acoustic word embeddings learned directly from speech (Chung and Glass, 2018, 2017; Chung et al., 2016) instead of those learned from text (Mikolov et al., 2013; Pennington et al., 2014).

References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual question answering. In *ICCV*.

Langzhou Chen, Mark J. F. Gales, and K. K. Chin. 2011. Constrained discriminative mapping transforms for unsupervised speaker adaptation. In *ICASSP*.

Laura Chiticariu, Rajasekar Krishnamurthy, Yunyao Li, Frederick Reiss, and Shivakumar Vaithyanathan. 2010. Domain adaptation of rule-based annotators for named-entity recognition tasks. In *EMNLP*.

Yu-An Chung and James Glass. 2017. Learning word embeddings from speech. In *NIPS ML4Audio Workshop*.

Yu-An Chung and James Glass. 2018. Speech2vec: A sequence-to-sequence framework for learning word embeddings from speech. *CoRR* abs/1803.08976.

Yu-An Chung, Chao-Chung Wu, Chia-Hao Shen, Hung-Yi Lee, and Lin-Shan Lee. 2016. Audio word2vec: Unsupervised learning of audio segment representations using sequence-to-sequence autoencoder. In *INTERSPEECH*.

Mortaza Doulaty, Oscar Saz, and Thomas Hain. 2015. Data-selective transfer learning for multi-domain speech recognition. In *INTERSPEECH*.

Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K. Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C Platt, et al. 2015. From captions to visual concepts and back. In *CVPR*.

Wei Fang, Juei-Yang Hsu, Hung-Yi Lee, and Lin-Shan Lee. 2016. Hierarchical attention model for improved machine comprehension of spoken content. In *SLT*.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding. In *EMNLP*.

David Golub, Po-Sen Huang, Xiaodong He, and Li Deng. 2017. Two-stage synthesis networks for transfer learning in machine. In *EMNLP*.

Jui-Ting Huang, Jinyu Li, Dong Yu, Li Deng, and Yifan Gong. 2013. Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers. In *ICASSP*.

Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst. 2016. From particular to general: A preliminary case study of transfer learning in reading comprehension. In *NIPS Machine Intelligence Workshop*.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *CVPR*.

Hung-Yi Lee, Yu-Yu Chou, Yow-Bang Wang, and Lin-Shan Lee. 2013. Unsupervised domain adaptation for spoken document summarization with structured support vector machine. In *ICASSP*.

Tzu-Chien Liu, Yu-Hsueh Wu, and Hung-Yi Lee. 2017. Query-based attention CNN for text similarity map. In *ICCV Workshop*.

Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. 2017. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *CVPR*.

David McClosky, Eugene Charniak, and Mark Johnson. 2010. Automatic domain adaptation for parsing. In *NAACL HLT*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

Sewon Min, Minjoon Seo, and Hannaneh Hajishirzi. 2017. Question answering through transfer learning from large fine-grained supervision data. In *ACL*.

Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How transferable are neural networks in nlp applications? In *EMNLP*.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering* 22(10):1345–1359.

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

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Matthew Richardson, Christopher J. C. Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision* 115(3):211–252.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. 2014. CNN features off-the-shelf: An astounding baseline for recognition. In *CVPRW*.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *NIPS*.

Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2016. MovieQA: Understanding stories in movies through question-answering. In *CVPR*.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. NewsQA: A machine comprehension dataset. In *RepL4NLP*.

Adam Trischler, Zheng Ye, and Xingdi Yuan. 2016. A parallel-hierarchical model for machine comprehension on sparse data. In *ACL*.

Bo-Hsiang Tseng, Sheng-Syun Shen, Hung-Yi Lee, and Lin-Shan Lee. 2016. Towards machine comprehension of spoken content: Initial TOEFL listening comprehension test by machine. In *INTERSPEECH*.

Willie Walker, Paul Lamere, Philip Kwok, Bhiksha Raj, Rita Singh, Evandro Gouvea, Peter Wolf, and Joe Woelfel. 2004. Sphinx-4: A flexible open source framework for speech recognition. Technical report.

R. Wallace, Thambiratnam K., and F. Seide. 2009. Unsupervised speaker adaptation for telephone call transcription. In *ICASSP*.

Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2015. Machine comprehension with syntax, frames, and semantics. In *ACL*.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2016. Towards AI-complete question answering: A set of prerequisite toy tasks. In *ICLR*.

Georg Wiese, Dirk Weissenborn, and Mariana Neves. 2017. Neural domain adaptation for biomedical question answering. In *CoNLL*.

Huijuan Xu and Kate Saenko. 2016. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In *ECCV*.

Yi Yang, Wen-Tau Yih, and Christopher Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *EMNLP*.

Zhilin Yang, Ruslan Salakhutdinov, and William W. Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. In *ICLR*.

Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. 2016. Stacked attention networks for image question answering. In *CVPR*.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *NIPS*.

Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2017. Aspect-augmented adversarial networks for domain adaptation. *Transactions of the Association for Computational Linguistics* 5:515–528.

Tracking State Changes in Procedural Text: A Challenge Dataset and Models for Process Paragraph Comprehension

Bhavana Dalvi Mishra[1*], **Lifu Huang**[2*], **Niket Tandon**[1], **Wen-tau Yih**[1], **Peter Clark**[1]
[1]Allen Institute for AI, Seattle, [2]Rensselaer Polytechnic Institute, Troy
{bhavanad,nikett,scottyih,peterc}@allenai.org, {warrior.fu}@gmail.com

Abstract

We present a new dataset and models for comprehending paragraphs about processes (e.g., photosynthesis), an important genre of text describing a dynamic world. The new dataset, ProPara, is the first to contain natural (rather than machine-generated) text about a changing world along with a full annotation of entity states (location and existence) during those changes (81k datapoints). The end-task, tracking the location and existence of entities through the text, is challenging because the causal effects of actions are often implicit and need to be inferred. We find that previous models that have worked well on synthetic data achieve only mediocre performance on ProPara, and introduce two new neural models that exploit alternative mechanisms for state prediction, in particular using LSTM input encoding and span prediction. The new models improve accuracy by up to 19%. The dataset and models are available to the community at http://data.allenai.org/propara.

1 Introduction

Building a reading comprehension (RC) system that is able to read a text document and to answer questions accordingly has been a long-standing goal in NLP and AI research. Impressive progress has been made in factoid-style reading comprehension, e.g., (Seo et al., 2017a; Clark and Gardner, 2017), enabled by well-designed datasets and modern neural network models. However, these models still struggle with questions that require *inference* (Jia and Liang, 2017).

Consider the paragraph in Figure 1 about photosynthesis. While top systems on SQuAD (Rajpurkar et al., 2016) can reliably answer lookup questions such as:
Q1: What do the roots absorb? (A: water, minerals)
they struggle when answers are not explicit, e.g.,
Q2: Where is sugar produced? (A: in the leaf)[1]

Chloroplasts in the leaf of the plant trap light from the sun. The roots absorb water and minerals from the soil. This combination of water and minerals flows from the stem into the leaf. Carbon dioxide enters the **leaf**. Light, water and minerals, and the carbon dioxide all combine into a mixture. This mixture forms **sugar** (glucose) which is what the plant eats.

Q: Where is sugar produced?
A: in the leaf

Figure 1: A paragraph from *ProPara* about photosynthesis (bold added, to highlight question and answer elements). Processes are challenging because questions (e.g., the one shown here) often require inference about the process states.

To answer Q2, it appears that a system needs knowledge of the world and the ability to reason with state transitions in multiple sentences: If carbon dioxide *enters* the leaf (stated), then it will be *at* the leaf (unstated), and as it is then used to produce sugar, the sugar production will be at the leaf too.

This challenge of modeling and reasoning with a changing world is particularly pertinent in text about *processes*, demonstrated by the paragraph in Figure 1. Understanding what is happening in such texts is important for many tasks, e.g., procedure execution and validation, effect prediction. However, it is also difficult because the world state is changing, and the causal effects of actions on that state are often implicit.

To address this challenging style of reading comprehension problem, researchers have created several datasets. The bAbI dataset (Weston et al., 2015) includes questions about objects moved throughout a paragraph, using machine-generated language over a deterministic domain with a small lexicon. The SCoNE dataset (Long et al., 2016) contains paragraphs describing a changing world state in three synthetic, deterministic domains, and

[*]Bhavana Dalvi Mishra and Lifu Huang contributed equally to this work.
[1]For example, the RC system BiDAF (Seo et al., 2017a) answers "glucose" to this question.

Proceedings of NAACL-HLT 2018, pages 1595–1604
New Orleans, Louisiana, June 1 - 6, 2018.

Paragraph (seq. of steps):		Participants: water	light	CO2	mixture	sugar
	state0	soil	sun	?	-	-
Roots absorb water from soil						
	state1	roots	sun	?	-	-
The water flows to the leaf.						
	state2	leaf	sun	?	-	-
Light from the sun and CO2 enter the leaf.						
	state3	leaf	leaf	leaf	-	-
The light, water, and CO2 combine into a mixture.						
	state4	-	-	-	leaf	-
Mixture forms sugar.						
	state5	-	-	-	-	leaf

Time

Figure 2: A (simplified) annotated paragraph from ProPara. Each filled row shows the existence and location of participants between each step ("?" denotes "unknown", "-" denotes "does not exist"). For example in state0, water is located at the soil.

assumes that a complete and correct model of the initial state is given for each task. However, approaches developed using synthetic data often fail to handle the inherent complexity in language when applied to organic, real-world data (Hermann et al., 2015; Winograd, 1972).

In this work, we create a new dataset, *ProPara* (Process Paragraphs), containing 488 human-authored paragraphs of procedural text, along with 81k annotations about the changing states (existence and location) of entities in those paragraphs, with an end-task of predicting location and existence changes that occur. This is the first dataset containing annotated, natural text for real-world processes, along with a simple representation of entity states during those processes. A simplified example is shown in Figure 2.

When applying existing state-of-the-art systems, such as Recurrent Entity Networks (Henaff et al., 2016) and Query-reduction Networks (Seo et al., 2017b), we find that they do not perform well on *ProPara* and the results are only slightly better than the majority baselines. As a step forward, we propose two new neural models that use alternative mechanisms for state prediction and propagation, in particular using LSTM input encoding and span prediction. The new models improve accuracy by up to 19%.

Our contributions in this work are twofold: (1) we create *ProPara*, a new dataset for process paragraph comprehension, containing annotated, natural language paragraphs about real-world processes, and (2) we propose two new models that learn to infer and propagate entity states in novel ways, and outperform existing methods on this dataset.

2 Related Work

Datasets: Large-scale reading comprehension datasets, e.g., SQuAD (Rajpurkar et al., 2016), TriviaQA (Joshi et al., 2017), have successfully driven progress in question answering, but largely targeting explicitly stated facts. Often, the resulting systems can be fooled (Jia and Liang, 2017), prompting efforts to create harder datasets where a deeper understanding of the text appears necessary (Welbl et al., 2017; Araki et al., 2016).

Procedural text is a genre that is particularly challenging, because the worlds they describe are largely implicit and changing. While there are few large datasets in this genre, two exceptions are bAbI (Weston et al., 2015) and SCoNE (Long et al., 2016), described earlier[2]. bAbI has helped advance methods for reasoning over text, such as memory network architectures (Weston et al., 2014), but has also been criticized for using machine-generated text over a simulated domain. SCoNE is closer to our goal, but has a different task (*given* a perfect world model of the initial state, predict the end state) and different motivation (handling ellipsis and coreference in context). It also used a deterministic, simulated world to generate data.

Models: For answering questions about procedural text, early systems attempted to extract a process structure (events, arguments, relations) from the paragraph, e.g., ProRead (Berant et al., 2014) and for newswire (Caselli et al., 2017). This allowed questions about event ordering to be answered, but not about state changes, unmodelled by these approaches.

More recently, several neural systems have been developed to answer questions about the world state after a process, inspired in part by the bAbI dataset. Building on the general Memory Network architecture (Weston et al., 2014) and gated recurrent models such as GRU (Cho et al., 2014), Recurrent Entity Networks (EntNet) (Henaff et al., 2016) is a state-of-the-art method for bAbI. EntNet uses a dynamic memory of hidden states (memory blocks) to maintain a representation of the world state, with a gated update at each step. Memory keys can be preset ("tied") to particular entities in the text, to encourage the memories to record information about those entities. Similarly, Query Reduction Networks (QRN) (Seo et al., 2017b) tracks state in

[2]The ProcessBank (Berant et al., 2014) dataset is smaller and does not address state change, instead containing 585 questions about event ordering and event arguments.

a paragraph, represented as a hidden vector h. QRN performs gated propagation of h across each time-step (corresponding to a state update), and uses h to modify ("reduce") the query to keep pointing to the answer at each step (e.g., "Where is the apple?" at step_1 might be modified to "Where is Joe?" at step_2 if Joe picks up the apple). A recent proposal, Neural Process Networks (NPN) (Bosselut et al., 2018), also models each entity's state as a vector (analogous to EntNet's tied memories). NPN computes the state change at each step from the step's predicted action and affected entity(s), then updates the entity(s) vectors accordingly, but does not model different effects on different entities by the same action.

Both EntNet and QRN find a final answer by decoding the final vector(s) into a vocabulary entry via softmax classification. In contrast, many of the best performing factoid QA systems, e.g., (Seo et al., 2017a; Clark and Gardner, 2017), return an answer by finding a *span* of the original paragraph using attention-based span prediction, a method suitable when there is a large vocabulary. We combine this span prediction approach with state propagation in our new models.

3 The ProPara Dataset

Task: Our dataset, *ProPara*, focuses on a particular genre of procedural text, namely simple scientific processes (e.g., photosynthesis, erosion). A system that understands a process paragraph should be able to answer questions such as: "*What are the inputs to the process?*", "*What is converted into what?*", and "*Where does the conversion take place?*"[3] Many of these questions reduce to understanding the basic dynamics of entities in the process, and we use this as our task: Given a process paragraph and an entity e mentioned in it, identify: (1) **Is** e created (destroyed, moved) in the process? (2) **When** (step #) is e created (destroyed, moved)? (3) **Where** is e created (destroyed, moved from/to)? If we can track the entities' *states* through the process and answer such questions, many of the higher-level questions can be answered too. To do this, we now describe how these states are representated in ProPara, and how the dataset was built.

Process State Representation: The states of the world throughout the whole process are represented as a grid. Each column denotes a *participant* entity (a span in the paragraph, typically a noun phrase) that undergoes some creation, destruction, or movement in the process. Each row denotes the *states* of all the participants after a *step*. Each sentence is a step that may change the state of one or more participants. Therefore, a process paragraph with m sentences and n participants will result in an $(m + 1) \times n$ grid representation. Each cell l_{ij} in this grid records the *location* of the j-th participant after the i-th step, and l_{0j} stores the location of j-th participant before the process.[4] Figure 2 shows one example of this representation.

Paragraph Authoring: To collect paragraphs, we first generated a list of 200 process-evoking prompts, such as "*What happens during photosynthesis?*", by instantiating five patterns[5], with nouns of the corresponding type from a science vocabulary, followed by manual rewording. Then, crowdsourcing (MTurk) workers were shown one of the prompts and asked to write a sequence of event sentences describing the process. Each prompt was given to five annotators to produce five (independent) paragraphs. Short paragraphs (4 or less sentences) were then removed for a final total of 488 paragraphs describing 183 processes. An example paragraph is the one shown earlier in Figure 1.

Grid and Existence: Once the process paragraphs were authored, we asked expert annotators[6] to create the initial grids. First, for each paragraph, they listed the participant entities that underwent a state change during the process, thus creating the column headers. They then marked the steps where a participant was created or destroyed. All state cells before a Create or after a Destroy marker were labeled as "not exists". Each initial grid annotation was checked by a second expert annotator.

Locations: Finally, MTurk workers were asked to fill in all the location cells. A location can be "unknown" if it is not specified in the text, or a span of the original paragraph. Five grids for the same paragraph were completed by five different Turkers, with average pairwise inter-annotator agreement of 0.67. The end result was 81,345 annotations over 488 paragraphs about 183 processes. The dataset

[3] For example, science exams pose such questions to test student's understanding of the text in various ways.

[4] We only trace locations in this work, but the representation can be easily extended to store other properties (e.g., temperature) of the participants.

[5] The five patterns are: How are ***structure*** formed? How does ***system*** work? How does ***phenomenon*** occur? How do you use ***device***? What happens during ***process***?

[6] Expert annotators were from our organization, with a college or higher degree.

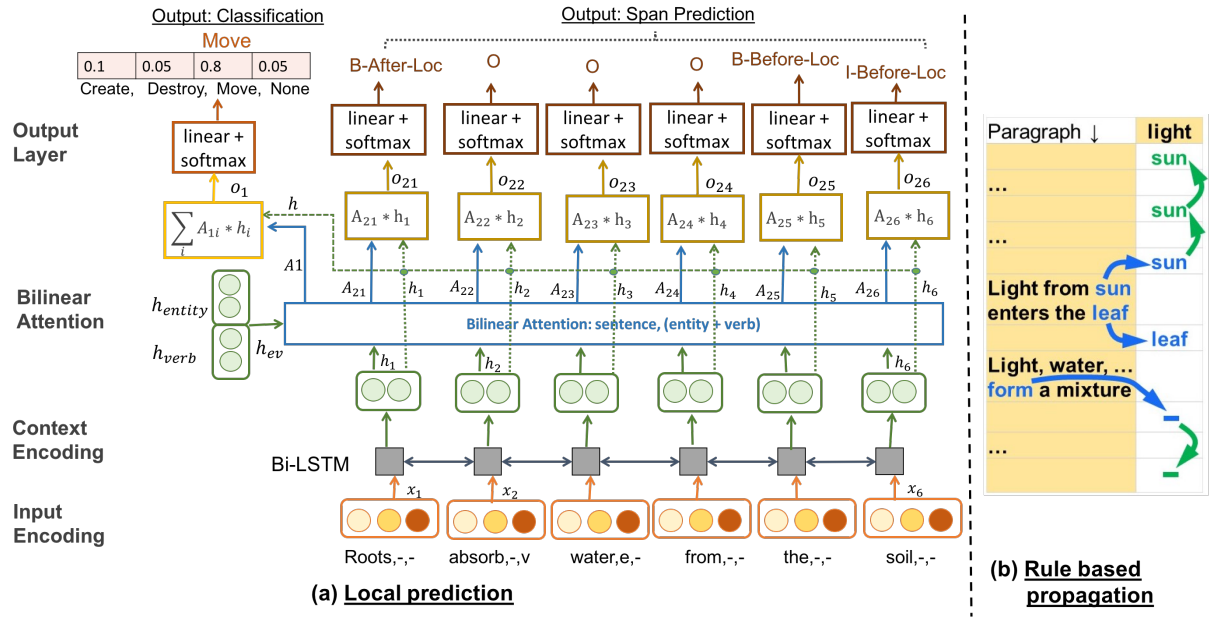


Figure 3: (a) ProLocal uses bidirectional attention to make local predictions about state change type and location (left), and then (b) propagates those changes globally using a persistence rule (right, shown for a single participant (the Light), local predictions shown in blue, propagations via persistence in green).

	bAbI	SCoNE	ProPara
Sentences	Synthetic	Natural	Natural
Questions	templated	templated	templated
# domains	20	3	183
Vocab #words	119	1314	2501
# sentences	131.1k	72.9k	3.3k
# unique sents	3.2k	37.4k	3.2k
Avg words/sent	6.5	10.2	9.0

Table 1: ProPara vs. other procedural datasets.

was then split 80/10/10 into train/dev/test by *process prompt*, ensuring that the test paragraphs were all about processes unseen in train and dev. Table 1 compares our dataset with bAbI and SCoNE.

4 Models

We present two new models for this task. The first, ProLocal, makes local state predictions and then algorithmically propagates them through the process. The second, ProGlobal, is an end-to-end neural model that makes all state predictions using global information.

4.1 ProLocal: A Local Prediction Model

The design of ProLocal consists of two main components: *local prediction* and *commonsense persistence*. The former infers all direct effects of individual sentences and the latter algorithmically propagates known values forwards and backwards to fill in any remaining unknown states.

4.1.1 Local Prediction

The intuition for local prediction is to treat it as a surface-level QA task. BiLSTMs with span prediction have been effective at answering surface-level questions, e.g., Given "Roots absorb water." and "Where is the water?", they can be reliably trained to answer "Roots" (Seo et al., 2017a). We incorporate a similar mechanism here.

Given a sentence (step) and a participant e in it, the local prediction model makes two types of predictions: the change type of e (one of: *no change*, *created*, *destroyed*, *moved*) and the locations of e before and after this step. The change type prediction is a multi-class classification problem, while the location prediction is viewed as a SQuAD-style surface-level QA task with the goal to find a location span in the input sentence. The design of this model is depicted in Figure 3(a), which adapts a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) recurrent neural network architecture (biLSTM) with attention for input encoding. The prediction tasks are handled by two different output layers. We give the detail of these layers below.

Input Encoding: Each word w_i in the input sentence is encoded with a vector $x_i = [v_w : v_e : v_v]$, the concatenation of a pre-trained GloVe (Pennington et al., 2014) word embedding v_w, indicator variables v_e on whether w_i is the specified participant and v_v on whether w_i is a verb (via a POS tagger).

Context Encoding: A biLSTM is used to contextualize the word representations in a given sentence. h_i denotes the concatenated output of the bidirectional LSTM for the embedded word x_i, and encodes the word's meaning in context.

Bilinear Attention: Given the participant and verb, the role of this layer is to identify which contextual word embeddings to attend to for generating the output. We first create h_{ev} by concatenating the contextual embedding of the participant and verb.[7] We then use a bilinear similarity function $sim(h_i, h_{ev}) = (h_i^T * B * h_{ev}) + b$, similar to (Chen et al., 2016), to compute attention weights A_i over each word w_i in the sentence.

For state change type prediction, the words between the verb and participant may be important, while for the location tagging, contextual cues such as "from" and "to" could be more predictive. Hence, we train two sets of attention parameters resulting in weights A_1 and A_2 which are combined with the contextual vectors h_i as described below to produce hidden states o_1 and o_2 that are fed to the output layers. Here, $|step|$ refers to number of words in the given step or sentence.

$$o_1 = \sum_i A_{1i} * h_i$$

$$o_2 = [(A_{21} * h_1) : (A_{22} * h_2) : \ldots : (A_{2|step|} * h_{|step|})]$$

Output 1: State Change Type: We apply a feed-forward network on hidden state o_1 to derive the probabilities of the four state change type categories: Create, Destroy, Move and None.

Output 2: Location Spans: The second output is computed by predicting BIO tags (one of five tags: B-Before-LOC, I-Before-LOC, B-After-LOC, I-After-LOC, O) for each word in the sentence. We apply a feed-forward network on hidden state o_{2i} for $word_i$ to derive the probabilities of these location tags. Notice that if the change type is predicted as "Create" (or "Destroy") then only the "after" (or "before") location prediction is used.

Training: We train the state change type prediction and location tag prediction models jointly, where the loss is the sum of their negative log likelihood losses. We use Adadelta (Zeiler, 2012) with learning rate 0.2 to minimize the total loss.

4.1.2 Commonsense Persistence

The local prediction model will partially fill in the state change grid, showing the direct locational effects of actions (including "not exists" and "unknown location"). To complete the grid, we then algorithmically apply a commonsense rule of persistence that propagates locations forwards and backwards in time where locations are otherwise missing. Figure 3(b) shows an example when applying this rule, where '?' indicates "unknown location". This corresponds to a rule of inertia: things are by default unchanged unless told otherwise. If there is a clash, then the location is predicted as unknown.

4.2 ProGlobal: A Global Prediction Model

Unlike ProLocal, the design principle behind ProGlobal is to model the persistence of state information *within* the neural model itself, rather than as a post-processing step. ProGlobal infers the states of *all* participants at each step, even if they are not mentioned in the current sentence, using: (1) the global context (i.e., previous sentences), and (2) the participant's state from the previous step.

Given a sentence (step) with its context (paragraph) and a participant e, ProGlobal predicts the existence and location of e after this step in two stages. It first determines the state of e as one of the classes ("not exist", "unknown location", "known location"). A follow-up location span prediction is made if the state is classified as "known location".

Figure 4 shows ProGlobal's neural architecture, where the left side is the part for state prediction at each step, and the right side depicts the propagation of hidden states from one step to the next. We discuss the detail of this model below.

Input Encoding: Given a participant e, for each step_i, we take the entire paragraph as input. Each word w in the paragraph is represented with three types of embeddings: the general word embedding v_w, a position embedding v_d which indicates the relative distance to the participant in the paragraph, and a sentence indicator embedding v_s which shows the relative position (*previous, current, following*) of each sentence in terms of the current step i. Both the position embedding and the sentence indicator embedding are of size 50 and are randomly initialized and automatically trained by the model. We concatenate these three types of embeddings to represent each word $x = [v_w : v_d : v_s]$.

Context Encoding: Similar to ProLocal, we use a biLSTM to encode the whole paragraph and use $\tilde{h}$ to denote the biLSTM output for each word.

State Prediction: As discussed earlier, we first predict the location state of a participant e. Let

[7] Multi-word entities/verbs or multiple verbs are represented by the average word vectors.

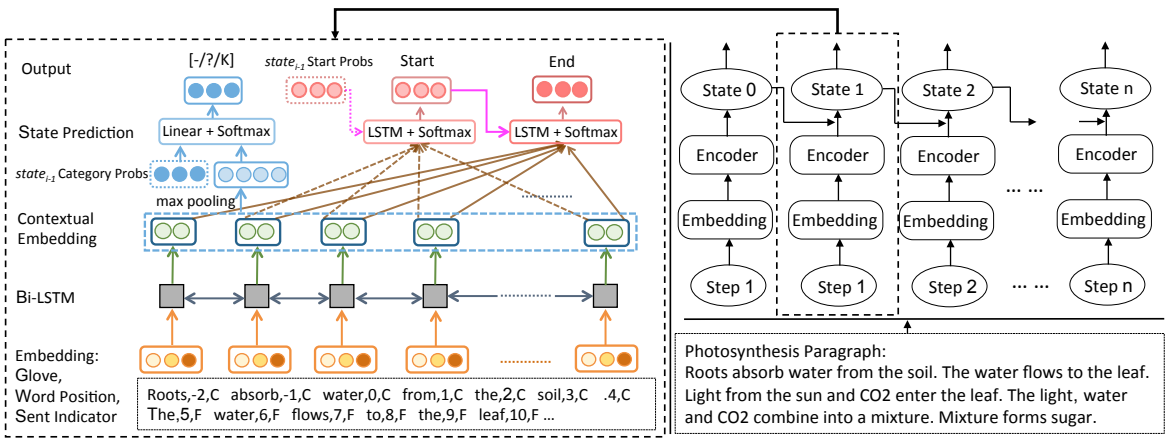


Figure 4: ProGlobal predicts a participant's state (type and location) after a given step using bilinear attention over the entire paragraph, combined with its predictions from the previous step.

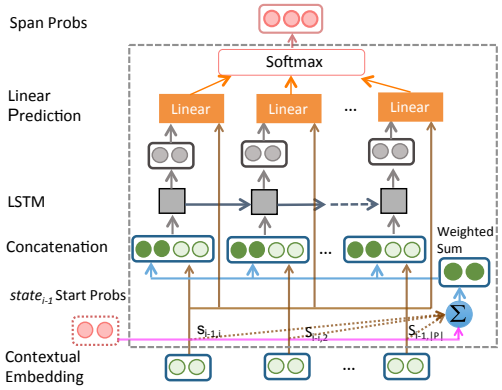


Figure 5: Details of the LSTM+Softmax unit, used for predicting the start/end words of a location.

$\tilde{H}_i^P = [\tilde{h}_i^1, \tilde{h}_i^2, ..., \tilde{h}_i^{|P|}]$ denote the hidden vectors (contextual embeddings) for words in step$_i$ with respect to participant e, where h_i^t denotes the t-th word representation output by the biLSTM layer and P is the whole paragraph. We then apply max pooling to derive a paragraph representation: $\mu_i^P = \max(\tilde{H}_i^P)$. To incorporate the category prediction of the previous step, step$_{i-1}$, we concatenate its probability vector $c_{i-1}^P \in \mathbb{R}^3$ with μ_i^P, and apply a feed-forward network to derive the probabilities of the three categories:

$$c_i^P = \text{softmax}(W_c \cdot [\mu_i^P : c_{i-1}^P] + b_c)$$

Location Span Prediction: (Figure 5). To predict the location span, we predict the start word of the span (by generating a probability distribution over words) and the end word. To predict the location start, we take two types of information as input: the start probability distribution $s_{i-1}^P \in \mathbb{R}^{|P|}$ predicted from step$_{i-1}$, and the contextual embeddings $\tilde{H}_i^P$ of words in the current step$_i$:

$$\tilde{H}_i^* = \sum_{t=1}^{|P|} s_{i-1}^t \cdot \tilde{H}_i^t$$

$$\varphi_i^t = \text{LSTM}([\tilde{H}_i^t : \tilde{H}_i^*])$$

where $\tilde{H}_i^*$ is a sum of word vectors in the paragraph, weighted by the start probabilities from the previous step step$_{i-1}$. φ_i^t is the encoded vector representation for the t-th word in the paragraph. We then concatenate $\tilde{H}_i^P$ and φ_i^P, and apply a feed-forward network to obtain the start probability distribution for step$_i$: $s_i^P = \text{softmax}(W_s \cdot [\tilde{H}_i^P : \varphi_i^P] + b_s)$. Similarly, to predict the end word of the span, we use the start probability distribution s_i^P of $step_i$ and $\tilde{H}_i^P$, and apply another LSTM and feed-forward networks to obtain the probabilities. For state$_0$ (the initial location before any steps), we directly feed the sequence of the vectors from the encoding layer to a linear transformation to predict the location start, and apply the same architecture to predict the location end.

Training: For each participant e of paragraph P, the objective is to optimize the sum of the negative log likelihood of the category classification and location span prediction[8]. We use Adadelta to optimize the models with learning rate 0.5.

5 Experiments and Analysis

5.1 Tasks & Evaluation Metrics

As described in Section 3, the quality of a model is evaluated based on its ability to answer three categories of questions, with respect to a given participant e:

[8]We compute the loss for location span prediction only when the category is annotated as "known location".

	Sentence Encoding	Intermediate State Representn.	Propagation through time	Answer Decoding
EntNet	positional encoding	Dynamic memory blocks	Gated propagation	Softmax classification
QRN	positional encoding	Single latent vector *h*	Gated propagation of *h*	Softmax classification
ProLocal	LSTM	Explicit symbolic	Algorithmic	Span prediction
ProGlobal	LSTM	Distribution over spans	LSTM	Span prediction

Table 2: Design decisions in the four neural models.

(Cat-1) **Is** *e* created (destroyed, moved) in the process?
(Cat-2) **When** (step#) is *e* created (destroyed, moved)?
(Cat-3) **Where** is *e* created (destroyed, moved from/to)?
These questions are answered by simple scans over the state predictions for the whole process. (Cat-1) is asked over all participants, while (Cat-2) and (Cat-3) are asked over just those participants that were created (destroyed, moved). The accuracy of the answers is used as the evaluation metric, except for questions that may have multiple answers (e.g., "*When is e moved?*"). In this case, we compare the predicted and gold answers and use the F_1 score as the "accuracy" of the answer set prediction.[9]

For questions in category (3), an answer is considered correct if the predicted location is identical to, or a sub-phrase of, the labeled location (typically just one or two words), after stop-word removal and lemmatizing.

5.2 Baseline Methods

We compare our models with two top methods inspired by the bAbI dataset, Recurrent Entity Networks (EntNet) and Query Reduction Networks (QRN), described earlier in Section 2. Both models make different use of gated hidden states to propagate state information through time, and generate answers using softmax. The detailed comparisons in their design are shown in Table 2.

We use the released implementations[10] (with default hyper-parameter values), and retrained them on our dataset, adapted to the standard bAbI QA format. Specifically, we create three separate variations of data by adding three bAbI-style questions after each step in a paragraph, respectively:

Q1. "Does *e* exist?" (yes/no)
Q2. "Is the location of *e* known?" (yes/no)
Q3. "Where is *e*?" (span)

The template Q1 is applied to all participants. Q2 will only be present in the training data if Q1 is "yes", and similarly Q3 is only present if Q2 is "yes". These three variations of data are used to train three different models from the same method.

At test time, we cascade the questions (e.g., ask Q2 only if the answer to the Q1 model is "yes"), and combine the model outputs accordingly to answer the questions in our target tasks (Section 5.1).

We also compare against a rule-based baseline and a feature-based baseline. The rule-based method, called ProComp, uses a set of rules that map (a SRL analysis of) each sentence to its effects on the world state, e.g., IF X moves to Y THEN after: at(X,Y). The rules were extracted from VerbNet (Schuler, 2005) and expanded. A full description of ProComp is available at (Clark et al., 2018). The feature-based method uses a Logistic Regression (LR) classifier to predict the state change type (Move, Create, etc.) for each participant + sentence pair, then a NER-style CRF model to predict the from/to locations as spans of the sentence. The LR model uses bag-of-word features from the sentence, along with a discrete feature indicating whether the participant occurs before or after the verb in the given sentence. The CRF model uses standard NER features including capitalization, a verb indicator, the previous 3 words, and the POS-tag of the current and previous word. Similar to our ProLocal model, we apply commonsense persistence rules (Section 4.1.2) to complete the partial state-change grids predicted by both these baselines.

5.3 Results

Parameter settings: Both our models use GloVe embeddings of size 100 pretrained on Wikipedia 2014 and Gigaword 5 corpora[11]. The number of hidden dimensions for the biLSTM are set to 50(ProLocal) and 100(ProGlobal). Dropout rates (Srivastava et al., 2014) for the contextual encoding layer are 0.3(ProLocal) and 0.2(ProGlobal). ProGlobal uses word position and sentence indicator embeddings each of size 50, and span prediction LSTMs with a hidden dimension of 10. The learning rates for Adadelta optimizer were

[9] This approach has been adopted previously for questions with multiple answers (e.g., (Berant et al., 2013)). For questions with only one answer, F_1 is equivalent to accuracy.

[10] https://github.com/siddk/entity-network and https://github.com/uwnlp/qrn

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

Question type	Baseline Models					Our Models		Human
(# questions)	Majority	QRN	EntNet	Rule-based	Feature-based	ProLocal	ProGlobal	Upper Bound
Cat-1 (750)	51.01	52.37	51.62	57.14	58.64	62.65	62.95	91.67
Cat-2 (601)	-	15.51	18.83	20.33	20.82	30.50	36.39	87.66
Cat-3 (823)	-	10.92	7.77	2.4	9.66	10.35	35.9	62.96
macro-avg	-	26.26	26.07	26.62	29.7	34.50	45.08	80.76
micro-avg		26.49	25.96	26.24	29.64	33.96	45.37	79.69

Table 3: Model accuracy on the end task (test partition of *ProPara*). Questions are (Section 5.1): (Cat-1) **Is** e_i created (destroyed, moved)? (Cat-2) **When** is e_i created (...)? (Cat-3) **Where** is e_i created (...)?

0.2(ProLocal) and 0.5(ProGlobal). Our models are trained on the train partition and the parameters tuned on the dev partition.

Table 3 compares the performance of various models on the *ProPara* test partition. For the first category of questions, we also include a simple majority baseline. We aggregate results over the questions in each category, and report both macro and micro averaged accuracy scores.

From Table 3, we can see that EntNet and QRN perform comparably when applied to *ProPara*. However, despite being the top-performing systems for the bAbI task, when predicting whether a participant entity is created, destroyed or moved, their predictions are only slightly better than the majority baseline. Compared to our local model ProLocal, EntNet and QRN are worse in predicting the exact step where a participant is created, destroyed or moved, but better in predicting the location. The weak performance of EntNet and QRN on *ProPara* is understandable: both systems were designed with a different environment in mind, namely a large number of examples from a few conceptual domains (e.g., moving objects around a house), covering a limited vocabulary. As a result, they might not scale well when applied to real procedural text, which justifies the importance of having a real challenge dataset like *ProPara*.

Although the rule-based baseline (Clark et al., 2018) uses rules mapping SRL patterns to state changes, its performance appears limited by the incompleteness and approximations in the rulebase, and by errors by the SRL parser. The feature-based baseline performs slightly better, but its performance is still poor compared to our neural models. This suggests that it has not generalized as well to unseen vocabulary (25% of the test vocabulary is not present in the train/dev partitions of *ProPara*).

When comparing our two models, it is interesting that ProGlobal performs substantially better than ProLocal. One possible cause of this is cascading errors in ProLocal: if a local state prediction is wrong, it may still be propagated to later time steps without any potential for correction, thus amplifying the error. In contrast, ProGlobal makes a state decision for every participant entity at every time-step, taking the global context into account, and thus appears more robust to cascading errors. Furthermore, ProGlobal's gains are mainly in Cat-2 and Cat-3 predictions, which rely more heavily on out-of-sentence cues. For example, 30% of the time the end-location is not explicitly stated in the state-change sentence, meaning ProLocal cannot predict the end-location in these cases (as no sentence span contains the end location). ProGlobal, however, uses the entire paragraph and may identify a likely end-location from earlier sentences.

Finally, we computed a human upper bound for this task (last column of Table 3). During dataset creation, each grid was fully annotated by 5 different Turkers (Section 3). Here, for each grid, we identify the Turker whose annotations result in the best score for the end task with respect to the other Turkers' annotations. The observed upper bound of ~80% suggests that the task is both feasible and well-defined, and that there is still substantial room for creating better models.

5.4 Analysis

To further understand the strengths and weaknesses of our systems, we ran the simplified paragraph in Figure 2 verbatim through the models learned by ProLocal and ProGlobal. The results are shown in Figure 6, with errors highlighted in red.

ProLocal correctly interprets "*Light from the sun and CO2 enters the leaf.*" to imply that the light was at the sun before the event. In addition, as there were no earlier mentions of light, it propagates this location backwards in time, (correctly) concluding the light was initially at the sun. However, it fails to predict that "*combine*" (after state 3) destroys the inputs, resulting in continued prediction of the existence and locations for those inputs. One contributing factor is that ProLocal's predic-

Paragraph (seq. of steps):		ProLocal Predictions: water	light	CO2	mixture	sugar
	state0	soil	sun	?	-	-
Roots absorb water from soil						
	state1	roots	sun	?	-	-
The water flows to the leaf.						
	state2	leaf	sun	?	-	-
Light from the sun and CO2 enter the leaf.						
	state3	leaf	?	?	-	-
The light, water, and CO2 combine into a mixture.						
	state4	leaf	?	?	?	-
Mixture forms sugar.						
	state5	leaf	?	?	-	?

Paragraph (seq. of steps):		ProGlobal Predictions: water	light	CO2	mixture	sugar
	state0	soil	-	-	-	-
Roots absorb water from soil						
	state1	soil	-	-	-	-
The water flows to the leaf.						
	state2	leaf	-	-	-	-
Light from the sun and CO2 enter the leaf.						
	state3	leaf	leaf	leaf	-	-
The light, water, and CO2 combine into a mixture.						
	state4	-	-	-	leaf	-
Mixture forms sugar.						
	state5	-	-	-	-	leaf

Figure 6: ProLocal (top) and ProGlobal (bottom) predictions on a simple paragraph (errors in red).

tions ignore surrounding sentences (context), potentially making it harder to distinguish destructive vs. non-destructive uses of "*combine*".

ProGlobal also makes some errors on this text, most notably not realizing the light and CO2 exist from the start (rather, they magically appear at the leaf). Adding global consistency constraints may help avoid such errors. It is able to predict the sugar is formed at the leaf, illustating its ability to persist and transfer location information from earlier sentences to draw correct conclusions.

We additionally randomly selected 100 prediction errors from the dev set for ProGlobal, and identified four phenomena contributing to errors:

(1) Implicit Creation/Destruction: In 37% of the errors, the information about the creation or destruction of a participant is implicit or missing, which resulted in existence classification errors. For example, in the sentences "*A fuel goes into the generator. The generator converts mechanical energy into electrical energy.*", "*fuel*" is implicitly consumed as the generator converts mechanical energy into electrical energy.

(2) Location Errors: In 27% of the examples, the location spans were not perfectly identified as follows: absolute wrong location span prediction (17%), longer span prediction (6%), and location prediction from different granularity (4%).

(3) Complex Syntax: In 13% of the examples, a moving participant and its target location are separated with a wide context within a sentence, making it harder for the model to locate the location span.

(4) Propagation: ProGlobal tends to propagate the previous location state to next step, which may override locally detected location changes or propagate the error from previous step to next steps. 9% of the errors are caused by poor propagation.

5.5 Future Directions

This analysis suggests several future directions:
Enforcing global consistency constraints: e.g., it does not make sense to create an already-existing entity, or destroy a non-existent entity. Global constraints were found useful in the earlier ProRead system (Berant et al., 2014).
Data augmentation through weak supervision: additional training data can be generated by applying existing models of state change, e.g., from VerbNet (Kipper et al., 2008), to new sentences to create additional sentence+state pairs.
Propagating state information backwards in time: if e_j is at l_{ij} after $step_i$, it is likely to also be there at $step_{i-1}$ given no information to the contrary. ProGlobal, EntNet, and QRNs are inherently unable to learn such a bias, given their forward-propagating architectures.

6 Conclusion

New datasets and models are required to take reading comprehension to a deeper level of machine understanding. As a step in this direction, we have created the ProPara dataset, the first to contain natural text about a changing world along with an annotation of entity states during those changes. We have also shown that this dataset presents new challenges for previous models, and presented new models that exploit ideas from surface-level QA, in particular LSTM input encoding and span prediction, producing performance gains. The dataset and models are available at `http://data.allenai.org/propara`.

Acknowledgements

We are grateful to Paul Allen whose long-term vision continues to inspire our scientific endeavors. We also thank Oren Etzioni, Carissa Schoenick, Mark Neumann, and Isaac Cowhey for their critical contributions to this project.

References

Jun Araki, Dheeraj Rajagopal, Sreecharan Sankaranarayanan, Susan Holm, Yukari Yamakawa, and Teruko Mitamura. 2016. Generating questions and multiple-choice answers using semantic analysis of texts. In *COLING*. pages 1125–1136.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proc. EMNLP'13*.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D Manning. 2014. Modeling biological processes for reading comprehension. In *Proc. EMNLP'14*.

Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. 2018. Simulating action dynamics with neural process networks. *6th International Conference on Learning Representations (ICLR)* .

Tommaso Caselli, Ben Miller, Marieke van Erp, Piek Vossen, Martha Palmer, Eduard Hovy, Teruko Mitamura, and David Caswell. 2017. Proceedings of the events and stories in the news workshop. In *Proceedings of the Events and Stories in the News Workshop*.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. *CoRR* abs/1606.02858.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proc. Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST'14)*. pages 103–111.

Christopher Clark and Matt Gardner. 2017. Simple and effective multi-paragraph reading comprehension. *arXiv preprint arXiv:1710.10723* .

Peter Clark, Bhavana Dalvi Mishra, and Niket Tandon. 2018. What happened? leveraging verbnet to predict the effects of actions in procedural text. *arXiv preprint arXiv:1804.05435* .

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2016. Tracking the world state with recurrent entity networks. *CoRR* abs/1612.03969.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*. pages 1693–1701.

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

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proc. EMNLP'17*.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proc. ACL'17*.

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2008. A large-scale classification of english verbs. *Language Resources and Evaluation* 42(1):21–40.

Reginald Long, Panupong Pasupat, and Percy Liang. 2016. Simpler context-dependent logical forms via model projections. In *ACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pages 1532–1543.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proc. EMNLP'16*.

Karin Kipper Schuler. 2005. *VerbNet: A broad-coverage, comprehensive verb lexicon*. Ph.D. thesis, Univ Colorado at Boulder.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017a. Bidirectional attention flow for machine comprehension. In *Proc. ICLR'17*.

Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2017b. Query-reduction networks for question answering. In *ICLR*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2017. Constructing datasets for multi-hop reading comprehension across documents. *arXiv preprint arXiv:1710.06481* .

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards AI-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698* .

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

Terry Winograd. 1972. Understanding natural language. *Cognitive Psychology* 3(1):1–191.

Matthew Zeiler. 2012. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

Combining Deep Learning and Topic Modeling for Review Understanding in Context-Aware Recommendation

Mingmin Jin[1] **Xin Luo**[1] **Huiling Zhu**[1,2] **Hankz Hankui Zhuo**[1,2*]
[1]School of Data and Computer Science, Sun Yat-Sen University
[2]Guangdong Key Laboratory of Big Data Analysis and Processing
Guangzhou, China
{jinmm[1],luox35[1],zhuhling6[2],zhuohank[2]}@{mail2[1],mail[2]}.sysu.edu.cn

Abstract

With the rise of e-commerce, people are accustomed to writing their reviews after receiving the goods. These comments are so important that a bad review can have a direct impact on others buying. Besides, the abundant information within user reviews is very useful for extracting user preferences and item properties. In this paper, we investigate the approach to effectively utilize review information for recommender systems. The proposed model is named LSTM-Topic matrix factorization (LTMF) which integrates both LSTM and Topic Modeling for review understanding. In the experiments on popular review dataset Amazon , our LTMF model outperforms previous proposed HFT model and ConvMF model in rating prediction. Furthermore, LTMF shows the better ability on making topic clustering than traditional topic model based method, which implies integrating the information from deep learning and topic modeling is a meaningful approach to make a better understanding of reviews.

1 Introduction

Recommender systems (RSs) are widely used in the field of electronic commerce to provide personalized recommendation services for customers. Most popular RSs are based on *Collaborative Filtering* (CF), which makes use of users' explicit ratings or implicit behaviour for recommendations (Koren, 2008). But CF models suffer from data sparsity, which is also called "*cold-start*" problem. Models perform poorly when there is few available data. To alleviate this problem, utilizing user reviews can be a good approach because user reviews can directly reflect users' preferences and items' properties and exactly correspond to the user latent factors and item latent factors in CF models.

*corresponding author

To understand user reviews, previous approaches are mainly based on *topic modeling*, a suite of algorithms that aim to discover the thematic information among documents (Blei, 2012). The simplest and commonly used topic model is *latent dirichlet allocation*(LDA). Recently, as deep learning shows great performance in computer vision (Krizhevsky et al., 2017) and NLP (Kim, 2014), some approaches combining deep learning with CF are proposed to capture latent context features from reviews.

However, we find there are some limitations in existing models. First, the LDA algorithm used in previous models like Hidden Factors as Topics (HFT) (McAuley and Leskovec, 2013) ignores contextual information. If a user writes "***I prefer apple than banana when choosing fruits***" in a review, we can clearly know the user's preference and recommend items including apple. But LDA ignores the structural information and considers the two words as the same since they both appear once in the sentence.

Compared with topic modeling, deep learning methods such as *Convolutional Neural Network* (CNN) and *Recurrent Neural Network* (RNN) are able to retain more context information. CNN uses sliding windows to capture local context and word order. RNN considers a sentence as a word sequence, and the former word information will be reserved and passed back, which gives RNN the ability to retain the whole sentence information.

But these still exist some problems. For CNN, the sizes of sliding windows are often small, which causes CNN model fails to link words in the sentence begin and end. Given the review ***"I prefer apple than google when choosing jobs"***, CNN can not notice the two words *'apple'* and *'jobs'* simultaneously if the windows size is small, so it will meet the ambiguity problem that the word *'apple'* means fruit or company. For RNN, al-

Proceedings of NAACL-HLT 2018, pages 1605–1614
New Orleans, Louisiana, June 1 - 6, 2018.

though it performs better than CNN on persisting former information, the information will still decreases with the length of sentences increasing. So when a review is long, the effect of RNN is limited.

Faced with these problems, we propose to integrate deep learning and topic modeling to extract more global context information and get a deeper understanding of user reviews. Deep learning methods can reserve context information, while topic modeling can provide word co-occurrence relation to make a supplement for information loss.

We use *Long Short-Term Memory* (LSTM) network for the deep learning part, because it is a special type of RNN which has better performance on gradient vanishing and long term dependence problems than vanilla RNN structure. We use LDA for the topic modeling part. Then the two parts are integrated into a matrix factorization framework. The final model is named **LSTM-Topic matrix factorization** (**LTMF**).

Furthermore, as the topic modeling part and deep learning part are connected in our model, the topic clustering results will be influenced by the deep learning information. In experiments, LTMF shows a better topic clustering ability than traditional LDA based HFT model. This gives us some inspiration on using the integrating methods into other tasks like sentiment classification.

In the remainder of the paper, we first review previous work related to our work. Then we address preliminaries and present our models in detail. After that we evaluate our approach by comparing our approach with state-of-the-art algorithms. Finally we conclude the paper with future work.

2 Related Work

There has been some earlier approaches to extract review information for RSs. Wang and Blei (2011) proposed *collaborative topic regression* (CTR) that combined topic modeling and collaborative filtering in a probabilistic model. McAuley and Leskovec (2013) developed a statistical model HFT using a transfer function to combine rating and review tightly. Ling et al. (2014) and Bao et al. (2014) proposed models similar to CTR and HFT with some structural differences.

Recently, several researchers begin to utilize deep learning in RSs. Wang et al. (2015) presented a Bayesian model *collaborative deep learning* (CDL) leveraging SDAE neural networks as a text feature learning component. Bansal et al. (2016) trained a *gated recurrent units* (GRUs) network to encode text sequences into latent vectors. Zhao et al. (2016) trained a deep CNN to discover the abstract representation of movie posters and still frames, and incorporated it into a neighborhood CF model. Kim et al. (2016) utilized CNN to retain contextual information in review, and developed a document context-aware recommendation model (ConvMF). The ConvMF model is a recently proposed model and is shown to outperform PMF and CDL, and we choose it as a baseline in our experiments. Zheng et al. (2017) proposed the *Deep Cooperative Neural Networks* (DeepCoNN) model which constructed two concurrent CNN to simultaneously model user and item reviews and then combined the features into Factorization Machine. Attention in neural networks has been popular in nearly years, Seo et al. (2017) proposed a model using CNN with dual attention for rating prediction. There are some similarity between the D-attn model with our LTMF model for we both want to extract more global information, where they use attention CNN model and we utilize the information from both topic modeling and deep learning. The D-attn model fail to work if there is not enough reviews, while our LTMF model use review information as a supplementary of rating. So it can still work effectively even there are few reviews.

Besides, Diao et al. (2014) proposed a method jointly modeling aspects, sentiments and ratings for movie recommendation. Hu et al. (2015) proposed MR3 model to combine ratings, social relations and reviews together for rating prediction. These hybrid models boost the performance than individual components, which also give us some inspiration on proposing the LTMF framework.

3 Preliminary

3.1 Notations

We use explicit ratings as the training and test data. Suppose there are M users

$$\mathcal{U} = \{u_1, u_2, ..., u_i, ..., u_M\}$$

and N items

$$\mathcal{V} = \{v_1, v_2, ..., v_j, ..., v_N\},$$

where each user and item is represented by a K-dimension latent vector, $u_i \in \mathbb{R}^K$ and $v_j \in \mathbb{R}^K$. The rating sparse matrix is denoted as $R \in \mathbb{R}^{M \times N}$, where r_{ij} is the rating of user u_i on item v_j. D is the review (document) corpus where d_{ij} is the review of user u_i on item v_j.

3.2 PMF: a standard matrix factorization model

Probabilistic Matrix Factorization (PMF) (Mnih and Salakhutdinov, 2008) is an effective recommendation model that uses matrix factorization (MF) technique to find the latent features of users and items from a probabilistic perspective. In PMF, the predicted rating $\hat{R}_{ij}$ is expressed as the inner product of user latent vector u_i and item latent vector v_j : $\hat{R}_{ij} = u_i^T v_j$. To get latent vectors, PMF minimises the following loss function:

$$\mathcal{L} = \sum_i^M \sum_j^N I_{ij}(R_{ij} - u_i^T v_j)^2 + \lambda_u \sum_i^M \| u_i \|_F^2 + \lambda_v \sum_j^N \| v_j \|_F^2, \quad (1)$$

where R_{ij} is the observed rating. The first part of Eq.(1) is the sum-of-squared-error between predicted and observed ratings and the second part is quadratic regularization terms to avoid overfitting. λ_u and λ_v are corresponding regularization parameters. I_{ij} is the indicator function which equals 1 if i-th user rated j-th item, and equals 0 otherwise.

3.3 HFT: understand reviews through topic modeling

Hidden Factors as Topics (HFT) (McAuley and Leskovec, 2013) provides an effective approach to integrates topic modeling into traditional CF models. It utilizes LDA, the simplest topic model which assumes there are k topics $\mathcal{T} = \{t_1, t_2, ..., t_k\}$ in document corpus D. Each document $d \in D$ has a topic distribution θ_d over $\mathcal{T}$ and each topic has a word distribution ϕ over a fixed vocabulary. To connect the document-topic distribution θ and item factors v, HFT proposes a transformation function:

$$\theta_{j,k} = \frac{\exp(\kappa v_{j,k})}{\sum_k \exp(\kappa v_{j,k})}, \quad (2)$$

where $v_{j,k}$ is the k-th latent factor in item vector v_j and $\theta_{j,k}$ is the k-th topic probability in item document-topic distribution θ_j, κ is the parameter controlling the "peakiness" of the transformation. Besides, HFT introduces an additional variable ψ to ensure the word distribution ϕ_k is a stochastic vector which satisfies $\sum_w \phi_{k,w} = 1$, the relation is denoted as follows:

$$\phi_{k,w} = \frac{\exp(\psi_{k,w})}{\sum_w \exp(\psi_{k,w})} \quad s.t. \quad \sum_w \phi_{k,w} = 1. \quad (3)$$

The final loss function is :

$$\mathcal{L} = \sum_i^M \sum_j^N I_{ij}(R_{ij} - \hat{R}_{ij})^2 - \lambda_t \sum_d^N \sum_{n \in N_d} \log \theta_{d,z_{d,n}} \phi_{z_{d,n},w_{d,n}}, \quad (4)$$

where $\hat{R}_{ij}$ is predicted ratings, θ and ϕ are the topic and word distribution respectively, $w_{d,n}$ is the n-th word in document d and $z_{d,n}$ is the word's corresponding topic, λ_t is a regularization parameters.

4 The LTMF Model

We propose the LSTM-Topic matrix factorization (LTMF) model, which integrates LSTM and topic modeling for recommendation. The model utilizes both rating and review information. For the rating part, we use probabilistic matrix factorization to extract rating latent vectors. For the review part, we use LDA (following the way of HFT) to extract topic latent vectors and adopt an LSTM architecture to generate document latent vectors . Then we combine the three vectors into a unified model. The overview of LTMF model is shown in Figure 1.

4.1 Parameter Relation

The left of Figure 1 is the parameters relations in LTMF model, which can be divided into three parts: $\Theta = \{\mathcal{U}, \mathcal{V}\}$ is the parameters associated with rating MF, $\Phi = \{\theta, \phi\}$ is the parameters associated with topic model, $\Omega = \{W, l\}$ is the parameters associated with LSTM. The shaded nodes are data (R:rating, D: reviews) where the others are parameters. Single connection lines represent there are constraint relationship between the two nodes. Double connections (e.g. $\mathcal{V}$ and θ) mean the relationship is bidirectional so they can affect each other's results.

4.2 LSTM Architecture

The right of Figure 1 is the LSTM architecture used in our models. For the j-th item, we concatenate all of its reviews as one document se-

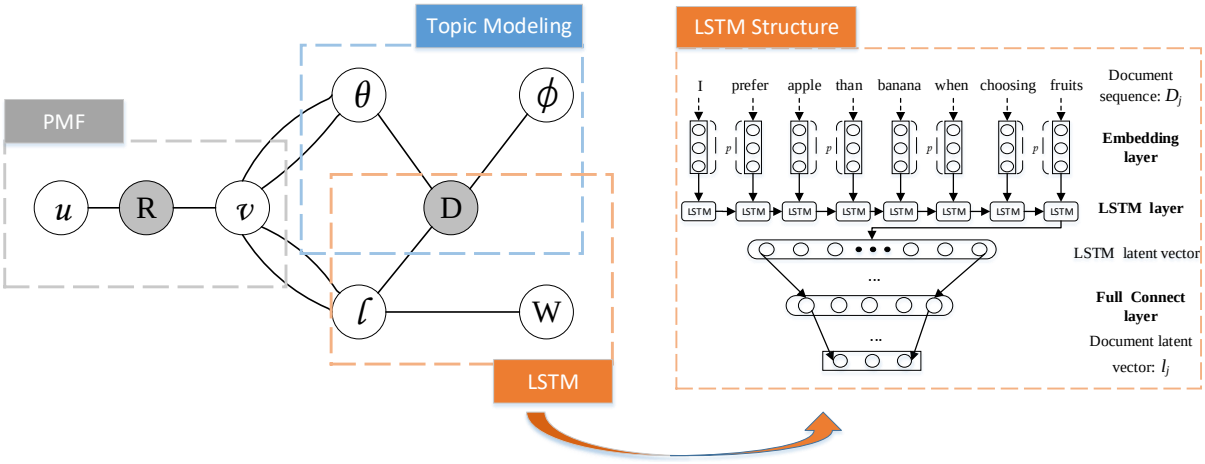


Figure 1: The overview of LTMF model. Left:Parameters relations of LTMF. Right:the detailed LSTM architecture.

quence D_j. Every word in the document sequence $D_j = (w_1, w_2, ..., w_{nj})$ will firstly be embedded into a p dimension vector. Next, word vectors are sent into LSTM network according to the word order in D_j and produces a latent vector. Finally, the latent vector is sent to a full connect layer whose output is the document latent vector l_j. The above process can be written as:

$$l_j = LSTM(D_j, W), \tag{5}$$

where D_j is the input document sequence, W represents weights and bias variables in LSTM network.

4.3 Probabilistic Prior

Gaussian distribution is the basic prior hypothesis in our model. We place zero-mean spherical Gaussian priors on user latent features u, LSTM weights W and observed ratings R. For item vector v, we place the Gaussian prior on its difference with LSTM output l_j :

$$p(v|l_j, \sigma_v^2) = \prod_{j=1}^{N} \mathcal{N}(v_j - l_j|0, \sigma_v^2 I)$$
$$= \prod_{j=1}^{N} \mathcal{N}(v_j|l_j, \sigma_v^2 I).$$

The function is important for connecting ratings and reviews. Although document vector l_j is closed to item feature vector v_j for they both reflect item's properties. There still exists some discrepancies. For example, when writing reviews, users usually write more about appearance and only briefly mention price. So in review based document vector l_j, the weight of "appearance" will be larger than rating based latent vector v_j. To preserve the discrepancy between v_j and l_j , we import the Gaussian noise vector σ_v as the offset.

4.4 Objective Function

Finally, we maximize the log-posterior of the three parts and get the objective function as follows:

$$\begin{aligned}\mathcal{L} = &\sum_{i}^{M}\sum_{j}^{N} I_{ij}(R_{ij} - u_i^T v_j)^2 \\ &- \lambda_t \sum_{d}^{N}\sum_{n\in N_d} \log \theta_{z_{d,n}} \phi_{z_{d,n},w_{d,n}} \\ &+ \lambda_u \sum_{i}^{M} \| u_i \|_F^2 + \lambda_v \sum_{j}^{N} \| v_j - l_j \|_F^2 \\ &+ \lambda_W \sum_{k}^{N_k} \| W_k \|_F^2,\end{aligned} \tag{6}$$

where N_k is the number of weighs in LSTM network, $\lambda_u, \lambda_v, \lambda_W$ are regularization parameters. z is the topic assignment for each word, λ_t is the regularization parameters to control the proportion of topic part.

The objective function of LTMF can be considered as an extended PMF model where the information from topic modeling and LSTM is included as regular terms. In the next section, we will explain how LTMF leverages the information from topic modeling and LSTM, and why LTMF can combine the information of the two parts.

4.5 The Effectiveness of LTMF

As shown in Figure 1, item vectors $\mathcal{V}$ connect with both topic part and LSTM part, which means the information from the two part will both affect the

result of item vectors. If we take partial derivative of Eq.(6) with respect to v_j, the constraint relationship can be clearer:

$$\frac{\partial \mathcal{L}}{\partial v_j} = \sum_{i=1}^{M} 2I_{ij}(R_{ij} - u_i^T v_j)u_i + 2\lambda_v(v_j - l_j) - \lambda_t \kappa \sum_{k=1}^{K}(n_{j,k} - N_j \frac{\exp(\kappa v_{j,k})}{\sum \exp(\kappa v_{j,k})}), \quad (7)$$

In Eq.(7), the optimization direction of v_j is subject to two regular terms. The former one is controlled by LSTM vector l_j. The latter one is controlled by topic parameters (κ, $n_{j,k}$, N_j). Hence, we can leverages the information from both LSTM and topic modeling for recommendation.

Besides, note the double connections between item vector $\mathcal{V}$ and topic distribution θ in Figure 1. They mean the information from topic modeling can affect the result of $\mathcal{V}$, while the change in $\mathcal{V}$ can also be passed to topic part and affect the review understanding result of topic modeling by Eq.2. For $\mathcal{V}$ and LSTM vector l, the analysis is the same. Indeed, item vectors $\mathcal{V}$ plays the role of transporter to connect LSTM part and topic modeling part. This is why LTMF can combine the information of topic modeling and LSTM to make a deeper understanding of user reviews.

Furthermore, LTMF provides an effective framework to integrate topic model with deep learning networks for recommendation. In experiments, we replace the LSTM part with CNN to make a comparison model. Experiments show both models boost the rating prediction accuracy.

4.6 Optimization

Our objective is to search:

$$\arg\min_{\Theta,\Phi,z,\kappa,\Omega} \mathcal{L}(\Theta, \Phi, z, \kappa, \Omega). \quad (8)$$

Recall that Θ is the parameters associated with ratings MF, Φ is the parameters associated with topic modeling, z is the topic assignment for each word, κ is the peakiness parameter to control the transformation between item vector v and topic distribution θ, Ω is the parameters associated with LSTM.

For v_j is coupled with the parameters of topic modeling and LSTM vector, we cannot optimize these parameters independently. We adopt a procedure that alternates between two steps. In each step we fix some parameters and optimize the others. The optimization process is shown below:

1. solve the objective by fixing z^t and Ω^t:
$$\arg\min_{\Theta,\Phi,\kappa} \mathcal{L}(\Theta, \Phi, z^t, \kappa, \Omega^t)$$
to update Θ^{t+1}, Φ^{t+1}, κ^{t+1}.

2. (a) update Ω^{t+1} with fixing ${v_j}^{t+1}$ and document sequence D_j.

   (b) sample $z_{d,j}^{t+1}$ with probability
$$p(z_{d,j}^{t+1} = k) = \phi_{k,w_{d,j}}^{t+1}.$$

In the step 1, we fix z and Ω to update remaining terms Θ, Φ, κ by L-BFGS algorithm. In the step 2, we fix Θ, Φ and κ to update LSTM parameters Ω and topic model parameters z. Since LSTM part and topic part are independent when item vectors $\mathcal{V}$ are certain, we can update the two term respectively. In step 2(a), we update Ω by back propagation algorithm. With fixing the other parameters, the objective function of W can be seen as a weighted squared error function ($\| v_j - l_j \|_F^2$) with L_2 regularized terms ($\| W \|_F^2$), which means we can use D_j as the input and v_j is the label to run the back propagation process. In step 2(b), we iterates through all documents and each word within to update $z_{d,j}$ via Gibbs Sampling. The reason why we do not divide the process into three steps is that the step 2(a) and 2(b) are independent with step 1 finished, which means we can parallelize the two steps.

Finally, we repeat these two steps until convergence. In practice, we run the step 1 with 5 gradient iterations using LBFGS, then we iterate the LSTM part 5 times. At the same time, we update the topic model part once. The whole process is called a cycle, and it usually takes 30 cycles to reach a local optimum.

In addition to the gradient of v_j, the gradients of other parameters used in step 1 are listed as follows:

$$\frac{\partial \mathcal{L}}{\partial u_i} = \sum_{j=1}^{N} 2I_{ij}(R_{ij} - u_i^T v_j)v_j + 2\lambda_u u_i. \quad (9)$$

$$\frac{\partial \mathcal{L}}{\partial \psi} = -\lambda_t \sum_{w=1}^{N_w} \sum_{k=1}^{K} \left(n_{k,w} - N_k \frac{\exp(\psi_{k,w})}{z_w} \right) \quad (10)$$

$$\frac{\partial \mathcal{L}}{\partial \kappa} = -\lambda_t \sum_{j=1}^{N} \sum_{k=1}^{K} v_{j,k} \left(n_{jk} - N_j \frac{\exp(\kappa v_{j,k})}{z_j} \right). \quad (11)$$

where ψ is used to determine word distribution ϕ by Eq.(3); $n_{k,w}$ is the number of times that word

Dataset	users	items	ratings	av. words per item	density
Amazon Instant Video (AIV)	29753	15147	135167	86.69	0.0030%
Apps for Android (AFA)	240931	51599	1322839	65.80	0.0106%
Baby (BB)	71812	42515	379969	81.83	0.0124%
Musical Instruments (MI)	29005	48751	150526	86.44	0.0106%
Office Product (OP)	59844	60628	286521	82.41	0.0079%
Pet Supplies (PS)	93270	70063	477859	78.86	0.0073%
Grocery Gourmet Food (GGF)	86389	108448	508676	76.97	0.0054%
Video Games (VG)	84257	39422	476546	92.55	0.0143%
Patio Lawn and Garden (PLG)	54167	57826	242944	82.22	0.0078%
Digital Music (DM)	56812	156496	351769	38.79	0.0040%

Table 1: Statistics of datasets.

w occurs in topic k; N_w is the word vocabulary size of the document corpus; N_k is the number of words in topic k; $n_{j,k}$ is the number of times when topic k occurs in the document of item j; N_j is the total number of words in document j; z_w and z_j are the corresponding normalizers:

$$z_w = \sum_{k=1}^{K} \exp(\psi_{k,w}), \quad z_j = \sum_{k=1}^{K} \exp(\kappa v_{j,k}).$$

5 Experiment

5.1 Datasets

We use the real-world Amazon dataset[1] (collected by McAuley et al. (2015)) for our experiments. For the original dataset is too large, we choose 10 sub datasets in experiments. To increase data density, we remove users which have less than 3 ratings. For raw review texts, we adopt the same preprocessing methods as ConvMF[2]: set the maximum length of a item document to 300; remove common stop words and document specific words which have document frequency higher than 0.5; choose top 8000 distinct words as the vocabulary; remove all non-vocabulary words to construct input document sequences. After preprocessing, the statistics of datasets are listed in Table 1, where the abbreviations of datasets are shown in parentheses.

5.2 Evaluation Procedure

5.2.1 Baseline

The baselines used in our experiments are listed as follows:

- PMF: Probabilistic Matrix Factorization (PMF) (Mnih and Salakhutdinov, 2008) is a standard matrix factorization model for RSs. It only uses rating information.

- HFT: This is a state-of-art method that combines reviews with ratings (McAuley and Leskovec, 2013). It utilizes LDA to capture unstructured textual information in reviews.

- ConvMF: Convolutional Matrix Factorization (ConvMF) (Kim et al., 2016) is a recently proposed recommendation model. It utilizes CNN to capture contextual information of item reviews.

- LMF: LSTM Matrix Factorization (LMF) is a submodel of LTMF without the topic part. We can compare it with ConvMF to show the effectiveness of LSTM than CNN on review understanding.

- CTMF: We modify the LTMF model by replacing the LSTM part with CNN (following the structure of ConvMF) and construct the comparison model CNN-Topic Matrix Factorization (CTMF). CTMF can be used to evaluate the effectiveness of combining deep learning and topic modeling.

In experiments, we randomly split one dataset into training set, test set, validation set under proportions of 80%, 10%, 10%, where each user and item appears at least once in the training set. We use Mean Square Error (MSE) as metric to evaluate various models.

[1] http://jmcauley.ucsd.edu/data/amazon/
[2] http://dm.postech.ac.kr/ cartopy/ConvMF/

Dataset	(a) PMF	(b) HFT	(c) ConvMF	(d) CTMF	(e) LMF	(f) LTMF
AIV	1.436 (0.02)	1.368 (0.02)	1.388 (0.02)	1.350 (0.03)	1.321 (0.02)	**1.309** (0.02)
AFA	1.673 (0.01)	1.649 (0.01)	1.651 (0.01)	1.648 (0.01)	1.635 (0.01)	**1.629** (0.01)
BB	1.643 (0.01)	1.577 (0.01)	1.556 (0.02)	1.531 (0.01)	1.513 (0.02)	**1.499** (0.01)
MI	1.555 (0.03)	1.423 (0.02)	1.399 (0.02)	1.367 (0.02)	1.317 (0.02)	**1.302** (0.02)
OP	1.622 (0.02)	1.547 (0.02)	1.501 (0.02)	1.466 (0.02)	1.432 (0.02)	**1.420** (0.02)
PS	1.796 (0.01)	1.736 (0.01)	1.698 (0.02)	1.680 (0.02)	1.646 (0.02)	**1.626** (0.01)
GGF	1.585 (0.01)	1.539 (0.01)	1.478 (0.01)	1.446 (0.02)	1.393 (0.01)	**1.386** (0.01)
VG	1.510 (0.02)	1.468 (0.01)	1.463 (0.01)	1.448 (0.01)	1.423 (0.01)	**1.409** (0.01)
PLG	1.854 (0.02)	1.779 (0.02)	1.710 (0.02)	1.678 (0.02)	1.628 (0.02)	**1.608** (0.02)
DM	1.197 (0.01)	1.171 (0.01)	1.032 (0.01)	0.990 (0.01)	0.968 (0.01)	**0.965** (0.01)

Table 2: MSE results of various models (K=5). The best results are highlighted in bold. The standard deviations of MSE results are shown in parenthesis.

5.2.2 Implementation Details

For all models, we set the dimension of user and item latent vectors $K = 5$, and initialize the vectors randomly between 0 and 1. Topic number and the dimension of document latent vector l are also set to 5. For methods using deep learning, we initialized word latent vectors randomly with the embedding dimension $p = 200$. The optimization algorithm used in back propagation is *rmsprop* and the activation function used in fully connected layer is *tanh* . In LSTM network, we set the output dimension to 128 and dropout rate 0.2. For CTMF, we adopt the same setting as ConvMF where the sliding window sizes is $\{3, 4, 5\}$ and the shared weights per window size is 100.

Hyper parameters are set as follows. For PMF, $\lambda_u = \lambda_v = 0.1$. For HFT, we select $\lambda_t \in \{1, 5\}$ which gives better result in each experiment. For LMF and ConvMF, we set $\lambda_u = 0.1$ and $\lambda_v = 5$. For LTMF and CTMF, we select $\lambda_t \in \{0.05, 0.1, 0.5\}$ which gives the lowest validation set error.

5.3 Quantitative analysis of rating prediction

We evaluate these models and report the lowest test set error on each dataset. The MSE results are shown in Table 2 where the best result of each dataset is highlighted in bold and the standard deviations of corresponding MSE are recorded in parenthesis.

We can see that the **LTMF model consistently outperform these baselines on all datasets** . This clearly confirms the effectiveness of our proposed method. To make a more intuitive comparison, the improvement histograms of these models are shown in Figure 2.

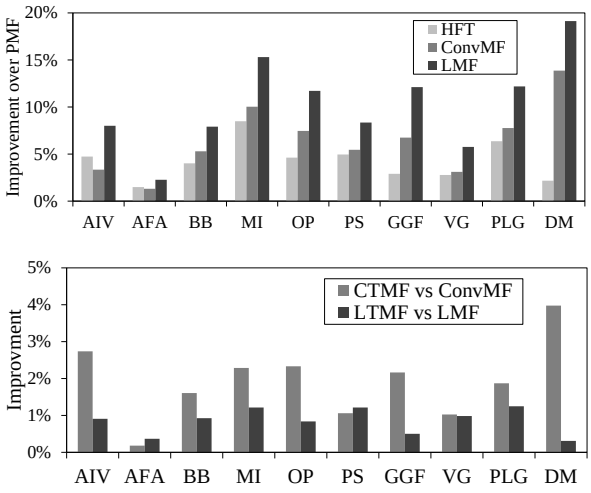


Figure 2: Above: Improvements of HFT, ConvMF and LMF, compared with PMF on different datasets. Below: Improvements of CTMF and LTMF, compared with ConvMF and LMF respectively.

The figure above are the improvements of HFT, ConvMF and LMF compared with PMF on different datasets, where PMF only uses rating information and the other three use both rating and review information with different approaches. We observe that all three methods make significant improvements over PMF, which indicates review information is helpful to model user and item features as well as improve recommendation results. Compared with HFT, LMF makes over 3% improvement on 9 out of the 10 datasets. ConvMF performs better than HFT while LMF still obtains over 3% improvement than ConvMF on 7 datasets. The differences between HFT, ConvMF and LMF can be attributed to their individual methods for re-

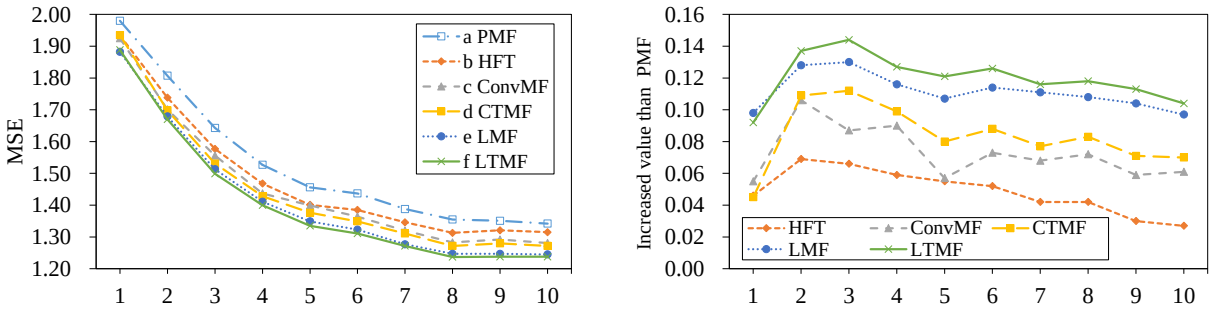


Figure 3: Results for recommendation within limited ratings and reviews. Left: the MSE values of all models. Right: the incerase compared with PMF.

views understanding. As mentioned in Section 1, Topic Modeling based HFT only considers the co-existence of words in texts and ignores structural context information. CNN based ConvMF lacks the ability to capture global context information due to the size limitation of sliding windows. This is exactly what LSTM possesses and why LSTM based LMF model outperforms ConvMF.

The figure below is the comparison of two integrated models (LTMF and CTMF) that import topic information with two original models that only use deep learning (LMF and ConvMF). We can see that both integrated models outperform the original models, which confirms our conjecture that **recommendation results can be improved by combining structural and unstructured information**. For CTMF model, it makes over 2% improvement on 5 out of 10 datasets compared with ConvMF. As to LTMF model, it achieves nearly 1% improvements that LMF on 7 out of 10 datasets.

The reason why LTMF gains less promotion can be explained from two sides. Numerically, for the comparison model LMF is already a strong baseline proposed by ourselves, it's more difficult to make a significant improvement. Theoretically, since LSTM can persist enough global information when the input sentence is relatively short, the supplements of topic information in LTMF are not so remarkable. As an illustration, we can compare the results on datasets "DM" and "VG". For the dataset "DM", as shown in Table 1, it has the fewest words per item (38.79) and the improvement of LTMF is minimum. But for the dataset "VG", it has the most words per item (92.55). The global context information obtained by LSTM will still decrease with such long sentences, and the topic information can make an effective supplement. So the improvement of LTMF on "VG" is greater and comparable with CTMF.

5.4 Recommendation with different data sparsity

Rating data and review data are always sparse in RSs. To compare these models on making recommendation in different data sparsity, especially for new users who only have limited ratings, we choose the dataset "Baby" and refilter it to make sure every user has at least N ratings (N varies from 1 to 10). A greater N means the user has rated more items, so the data sparsity problem is weaker. We test all models on the 10 subsets of "Baby" with the same dataset split ratio and text preprocessing. The final results are shown in Figure 3, where the left one is the MSE values of all models, and the right one is the increase of the other models compared with PMF.

We can observe that all models gain better recommendation accuracy with the increment of user rating number N. In other words, user and item latent features can be better extracted with more useful information. When N is small, especially when $N = \{2, 3\}$, the models which utilize both review and rating information achieve biggest improvements over PMF. It suggests that **review information can provide effective supplement when rating data is scarce**. With the increase of N, the improvements of all review used models become smaller. This is because models can extract more features from gradually dense ratings data, and the effectiveness of review data begins to decrease. Same as the previous experiment, our LTMF model achieve the best results in the comparison with other models.

5.5 Qualitative Analysis

In HFT, the result of topic words only depends on the information from Topic Modeling. But in our

Office Product (OP)				
topic1	topic2	topic3	topic4	topic5
envelope	markers	pins	wallet	planner
erasers	**compatible**	scale	notebooks	keyboard
needs	lead	***huge***	window	tab
numbers	mail	credit	notebook	remove
letters	***nice***	**document**	cardboard	stickers
christmas	camera	***attach***	plug	clips

Table 3: Top topic words discovered by HFT

Office Product (OP)				
topic1	topic2	topic3	topic4	topic5
bands	bags	scale	wallet	folder
drum	camera	**document**	clock	folders
remote	cabinet	magnets	coins	binder
chalk	**compatible**	monitors	notebooks	stickers
presentation	tray	pins	shredder	remove
buttons	party	fax	bookmark	head

Table 4: Top topic words discovered by LTMF

proposed LTMF framework, the information extracted by LSTM and Topic Modeling will both affect the final word clustering results. So, we can compare the topic words discovered by HFT and LTMF to evaluate whether combing LSTM and Topic Modeling is able to make a better understanding of user reviews.

We choose the dataset "Office Product" (OP) and show the top topic words of HFT and LTMF in Table 3 and Table 4. As we can see, there are many words existed in both tables (e.g. "wallet", "notebooks", "document"). These words are closely related to the category of dataset "Office Product", which implies both models can get a good interpretation of user reviews.

However, when we carefully compare the two tables, there exists some differences. In Table 3, there are some adjectives and verbs which have little help for topic clustering (e.g. "nice", "huge", "attach"), but they still get large weights and appear in the front of topic words list. Obviously, HFT misinterprets these words for they usually appear together with the real topic words. In Table 4, we are not able to find them in top words list, because extra information from LSTM makes a timely supplement. Besides, similar situations also occur on words "document" and "compatible". The word "document" is an apparent topic word, so LTMF gives it a larger weight in topic words list. For the word "compatible", as an adjectives, it can provide less topic information than nouns, so LTMF decreases its weight and put "camera" in the second place. From the above analysis we can see LTMF shows the better topic clustering ability than HFT.

6 Conclusion and Future Work

In this paper, we investigate the approach to effectively utilize review information for RSs. We propose the LTMF model which integrates both LSTM and Topic modeling in context aware recommendation. In the experiments, our LTMF model outperforms HFT and ConvMF in rating prediction especially when the data is sparse. Furthermore, LTMF shows better ability on making topic clustering than traditional topic model based method HFT, which implies integrating the information from deep learning and topic modeling is a meaningful approach to make a better understanding of reviews. In the future, we plan to evaluate more complex networks for recommendation tasks under the framework proposed by LTMF. Besides, we are interested to apply the method of combing topic model and deep learning into some traditional NLP tasks.

Acknowledgments

We thank the National Key Research and Development Program of China (2016YFB0201900), National Natural Science Foundation of China (U1611262), Guangdong Natural Science Funds for Distinguished Young Scholar (2017A030306028), Pearl River Science and Technology New Star of Guangzhou, and Guangdong Province Key Laboratory of Big Data Analysis and Processing for the support of this research.

References

Trapit Bansal, David Belanger, and Andrew McCallum. 2016. Ask the gru: Multi-task learning for deep text recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, New York, NY, USA, RecSys '16, pages 107–114. https://doi.org/10.1145/2959100.2959180.

Yang Bao, Hui Fang, and Jie Zhang. 2014. Topicmf: Simultaneously exploiting ratings and reviews for recommendation. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. AAAI Press, AAAI'14, pages 2–8. http://dl.acm.org/citation.cfm?id=2893873.2893874.

David M. Blei. 2012. Probabilistic topic models. *Commun. ACM* 55(4):77–84. https://doi.org/10.1145/2133806.2133826.

Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J. Smola, Jing Jiang, and Chong Wang. 2014. Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA, KDD '14, pages 193–202. https://doi.org/10.1145/2623330.2623758.

Guang-Neng Hu, Xin-Yu Dai, Yunya Song, Shu-Jian Huang, and Jia-Jun Chen. 2015. A synthetic approach for recommendation: Combining ratings, social relations, and reviews. In *Proceedings of the 24th International Conference on Artificial Intelligence*. AAAI Press, IJCAI'15, pages 1756–1762. http://dl.acm.org/citation.cfm?id=2832415.2832493.

Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. 2016. Convolutional matrix factorization for document context-aware recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, New York, NY, USA, RecSys '16, pages 233–240. https://doi.org/10.1145/2959100.2959165.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 1746–1751. https://doi.org/10.3115/v1/D14-1181.

Yehuda Koren. 2008. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA, KDD '08, pages 426–434. https://doi.org/10.1145/1401890.1401944.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. Imagenet classification with deep convolutional neural networks. *Commun. ACM* 60(6):84–90. https://doi.org/10.1145/3065386.

Guang Ling, Michael R. Lyu, and Irwin King. 2014. Ratings meet reviews, a combined approach to recommend. In *Proceedings of the 8th ACM Conference on Recommender Systems*. ACM, New York, NY, USA, RecSys '14, pages 105–112. https://doi.org/10.1145/2645710.2645728.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems*. ACM, New York, NY, USA, RecSys '13, pages 165–172. https://doi.org/10.1145/2507157.2507163.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, USA, SIGIR '15, pages 43–52. https://doi.org/10.1145/2766462.2767755.

Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic matrix factorization. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, Curran Associates, Inc., pages 1257–1264.

Sungyong Seo, Jing Huang, Hao Yang, and Yan Liu. 2017. Interpretable convolutional neural networks with dual local and global attention for review rating prediction. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, New York, NY, USA, RecSys '17, pages 297–305. https://doi.org/10.1145/3109859.3109890.

Chong Wang and David M. Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA, KDD '11, pages 448–456. https://doi.org/10.1145/2020408.2020480.

Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA, KDD '15, pages 1235–1244. https://doi.org/10.1145/2783258.2783273.

Lili Zhao, Zhongqi Lu, Sinno Jialin Plan, and Qiang Yang. 2016. Matrix factorization+ for movie recommendation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. AAAI Press, IJCAI'16, pages 3945–3951. http://dl.acm.org/citation.cfm?id=3061053.3061171.

Lei Zheng, Vahid Noroozi, and Philip S. Yu. 2017. Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, New York, NY, USA, WSDM '17, pages 425–434. https://doi.org/10.1145/3018661.3018665.

Deconfounded Lexicon Induction for Interpretable Social Science

Reid Pryzant[*], **Kelly Shen**[*], **Dan Jurafsky**[+†], **Stefan Wager**[‡]
[*+]Department of Computer Science
[†]Department of Linguistics
[‡]Graduate School of Business
Stanford University
{rpryzant,kshen21,jurafsky,swager}@stanford.edu

Abstract

NLP algorithms are increasingly used in computational social science to take linguistic observations and predict outcomes like human preferences or actions. Making these social models transparent and interpretable often requires identifying features in the input that predict outcomes while also controlling for potential confounds. We formalize this need as a new task: inducing a lexicon that is predictive of a set of target variables yet uncorrelated to a set of confounding variables. We introduce two deep learning algorithms for the task. The first uses a bifurcated architecture to separate the explanatory power of the text and confounds. The second uses an adversarial discriminator to force confound-invariant text encodings. Both elicit lexicons from learned weights and attentional scores. We use them to induce lexicons that are predictive of timely responses to consumer complaints (controlling for product), enrollment from course descriptions (controlling for subject), and sales from product descriptions (controlling for seller). In each domain our algorithms pick words that are associated with *narrative persuasion*; more predictive and less confound-related than those of standard feature weighting and lexicon induction techniques like regression and log odds.

1 Introduction

Applications of NLP to computational social science and data science increasingly use *lexical features* (words, prefixes, etc) to help predict non-linguistic outcomes like sales, stock prices, hospital readmissions, and other human actions or preferences. Lexical features are useful beyond predictive performance. They enhance interpretability in machine learning because practitioners know why their system works. Lexical features can also be used to understand the subjective properties of a text.

For social models, we need to be able to select lexical features that predict the desired outcome(s) while also controlling for potential confounders. For example, we might want to know which words in a product description lead to greater sales, regardless of the item's price. Words in a description like "luxury" or "bargain" might increase sales but also interact with our confound (price). Such words don't reflect the *unique* part of text's effect on sales and should not be selected. Similarly, we might want to know which words in a consumer complaint lead to speedy administrative action, regardless of the product being complained about; which words in a course description lead to higher student enrollment, regardless of the course topic. These instances are associated with *narrative persuasion*: language that is responsible for altering cognitive responses or attitudes (Spence, 1983; Van Laer et al., 2013).

In general, we want words which are predictive of their targets yet decorrelated from confounding information. The lexicons constituted by these words are useful in their own right (to develop causal domain theories or for linguistic analysis) but also as interpretable features for down-stream modeling. Such work could help widely in applications of NLP to tasks like linking text to sales figures (Ho and Wu, 1999), to voter preference (Luntz, 2007; Ansolabehere and Iyengar, 1995), to moral belief (Giles et al., 2008; Keele et al., 2009), to police respect (Voigt et al., 2017), to financial outlooks (Grinblatt and Keloharju, 2001; Chatelain and Ralf, 2012), to stock prices (Lee et al., 2014), and even to restaurant health inspections (Kang et al., 2013).

Identifying linguistic features that are indicative of such outcomes and decorrelated with confounds is a common activity among social scientists, data scientists, and other machine learning practitioners. Indeed, it is essential for developing transpar-

Proceedings of NAACL-HLT 2018, pages 1615–1625
New Orleans, Louisiana, June 1 - 6, 2018.

ent and interpretable machine learning NLP models. Yet there is no generally accepted and rigorously evaluated procedure for the activity. Practitioners have conducted it on a largely ad-hoc basis, applying various forms of logistic and linear regression, confound-matching, or association quantifiers like mutual information or log-odds to achieve their aims, all of which have known drawbacks (Imai and Kim, 2016; Gelman and Loken, 2014; Wurm and Fisicaro, 2014; Estévez et al., 2009; Szumilas, 2010).

We propose to overcome these drawbacks via two new algorithms that consider the causal structure of the problem. The first uses its architecture to learn the part of the text's effect which the confounds cannot explain. The second uses an adversarial objective function to match text encoding distributions regardless of confound treatment. Both elicit lexicons by considering learned weights or attentional scores. In summary, we

1. Formalize the problem into a new task.
2. Propose a pair of well-performing neural network based algorithms.
3. Conduct the first systematic comparison of algorithms in the space, spanning three domains: consumer complaints, course enrollments, and e-commerce product descriptions.

The techniques presented in this paper will help scientists (1) better interpret the relationship between words and real-world phenomena, and (2) render their NLP models more interpretable[1].

2 Deconfounded Lexicon Induction

We begin by formalizing this language processing activity into a task. We have access to text(s) T, target variable(s) Y, and confounding variable(s) C. The goal is to pick a lexicon L such that when words in T belonging to L are selected, the resulting set $L(T)$ is related to Y but not C. There are two types of signal at play: the part of Y that T can explain, and that explainable by C. These signals often overlap because language reflects circumstance, but we are interested in the part of T's explanatory power which is *unique* to T, and hope to choose L accordingly.

So if $\text{Var}\left[\mathbb{E}\left[Y|L(T), C\right]\right]$ is the information in Y explainable by both $L(T)$ and C, then our goal is to choose L such that this variance is maximized *after* C has been fixed. With this in mind, **we formalize the task of deconfounded lexicon induction as finding a lexicon L that maximizes an *informativeness coefficient***,

$$\mathcal{I}(L) = \mathbb{E}\left[\text{Var}\left[\mathbb{E}\left[Y\middle|L(T),\, C\right]\middle|C\right]\right], \quad (1)$$

which measures the explanatory power of the lexicon beyond the information already contained in the confounders C. Thus, highly informative lexicons cannot simply collect words that reflect the confounds. Importantly, this coefficient is only valid for comparing different lexicons of the same size, because in terms of maximizing this criterion, using the entire text will trivially make for the best possible lexicon.

Our coefficient $\mathcal{I}(L)$ can also be motivated via connections to the causal inference literature: in Section 7, we show that—under assumptions often used to analyze causal effects in observational studies—the coefficient $\mathcal{I}(L)$ can correspond exactly to the strength of T's causal effects on Y.

Finally, note that by expanding out an ANOVA decomposition for Y, we can re-write this criterion as

$$\begin{aligned}\mathcal{I}(L) = \mathbb{E}\left[\left(Y - \mathbb{E}\left[Y\middle|C,\, L(T)\right]\right)^2\right] \\ - \mathbb{E}\left[\left(Y - \mathbb{E}\left[Y\middle|C\right]\right)^2\right],\end{aligned} \quad (2)$$

i.e., $\mathcal{I}(L)$ measures the performance improvement $L(T)$ affords to optimal predictive models that already have access to C. We use this fact for evaluation in Section 4.

3 Proposed Algorithms

We continue by describing the pair of novel algorithms we are proposing for deconfounded lexicon induction problems.

3.1 Deep Residualization (DR)

Motivation. Our first method is directly motivated by the setup from Section 2. Recall that $\mathcal{I}(L)$ measures the amount by which $L(T)$ can improve predictions of Y made from the confounders C. We accordingly build a neural network architecture that first predicts Y directly from C as well as possible, and then seeks to fine-tune those predictions using T.

Description. First we pass the confounds through a feed-forward neural network (FFNN) to obtain

[1]Code, hyperparameters, and instructions for practitioners are online at https://nlp.stanford.edu/projects/deconfounded-lexicon-induction/

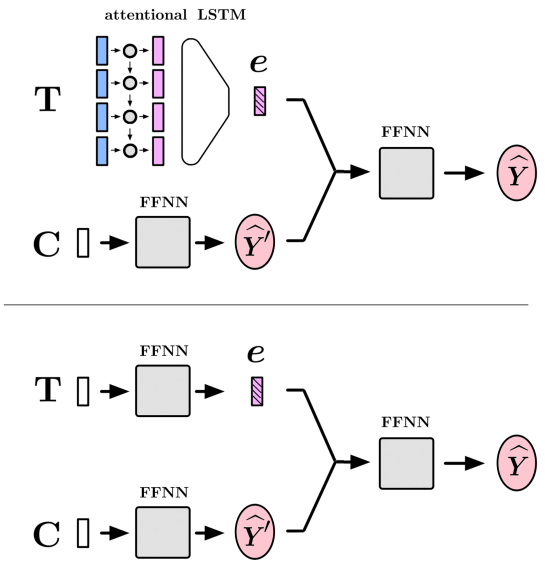


Figure 1: The Deep Residualization (DR) selector. Values which are used to calculate losses are enclosed in red ovals. *Top*: DR+ATTN, which represents text as a sequence of word embeddings. *Bottom*: DR+BOW, which represents text as a vector of word frequencies.

preliminary predictions $\hat{Y}'$. We also encode the text into a continuous vector $\boldsymbol{e} \in \mathcal{R}^d$ via two alternative mechanisms:

1. DR+ATTN: the text is converted into a sequence of embeddings and fed into Long Short-Term Memory (LSTM) cell(s) (Hochreiter and Schmidhuber, 1997) followed by an attention mechanism inspired by Bahdanau et al. (2015). If the words of a text have been embedded as vectors $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, ..., $\boldsymbol{x}_n$ then $\boldsymbol{e}$ is calculated as a weighted average of hidden states, where the weights are decided by a FFNN whose parameters are shared across timesteps:

$$\begin{aligned}
\boldsymbol{h}_0 &= \vec{0} \\
\boldsymbol{h}_t &= LSTM(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}) \\
l_t &= ReLU(\boldsymbol{W}^{attn}\boldsymbol{h}_t) \cdot \boldsymbol{v}^{attn} \\
p_t &= \frac{exp(l_t)}{\sum exp(l_i)} \\
\boldsymbol{e} &= \sum p_i \boldsymbol{h}_i
\end{aligned}$$

2. DR+BOW: the text is converted into a vector of word frequencies, which is compressed with a two-layer feedforward neural network (FFNN):

$$\begin{aligned}
\boldsymbol{t} &= [freq_1, freq_2, ..., freq_k] \\
\boldsymbol{h} &= ReLU(\boldsymbol{W}^{hidden}\boldsymbol{t}) \\
\boldsymbol{e} &= ReLU(\boldsymbol{W}^{output}\boldsymbol{t})
\end{aligned}$$

We then concatenate $\boldsymbol{e}$ with $\hat{Y}'$ and feed the result through another neural network to generate final predictions $\hat{Y}$. If Y is continuous we compute loss with

$$\mathcal{L}_{continuous} = ||\hat{Y} - Y||_2$$

If Y is categorical we compute loss with

$$\mathcal{L}_{categorical} = -p^* \log \widehat{p^*}$$

Where $\widehat{p^*}$ corresponds to the predicted probability of the correct class. The errors from $\hat{Y}$ are propagated through the whole model, but the errors from $\hat{Y}'$ are only used to train its progenitor (Figure 1).

Note the similarities between this model and the popular residualizing regression (RR) technique (Jaeger et al., 2009; Baayen et al., 2010, inter alia). Both use the text to improve an estimate generated from the confounds. RR treats this as two separate regression tasks, by regressing the confounds against the variables of interest, and then using the residuals as features, while our model introduces the capacity for nonlinear interactions by backpropagating between RR's steps.

Lexicon Induction. We elicit lexicons from +ATTN style models by (1) running inference on a test set, but rather than saving those predictions, saving the attentional distribution over each source text, and (2) mapping each word to its average attentional score and selecting the k highest-scoring words.

For +BOW style models, we take the matrix that compresses the text's word frequency vector, then score each word by computing the l_1 norm of the column that multiplies it, with the intuition that important words are dotted with big vectors in order to be a large component of $\boldsymbol{e}$.

3.2 Adversarial Selector (A)

Motivation. We begin by observing that a desirable L can explain Y, but is unrelated to C, which implies it should should struggle to predict C. The Adversarial Selector draws inspiration from this.

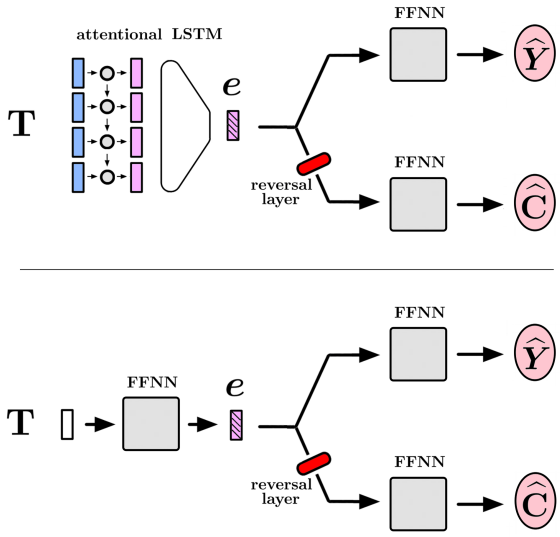


Figure 2: The Adversarial (A) selector. Values which are used to calculate losses are enclosed in red ovals. *Top*: A+ATTN, which represents text as a sequence of word embeddings. *Bottom*: A+BOW, which represents text as a vector of word frequencies.

It learns adversarial encodings of T which are useful for predicting Y, but *not* useful for predicting C. It is depicted in Figure 2.

Description. First, we encode T into $\boldsymbol{e} \in \mathcal{R}^d$ via the same mechanisms as the Deep Residualizer of Section 3.1. $\boldsymbol{e}$ is then passed to a series of FFNNs ("prediction heads") which are trained to predict each target and confound with the same loss functions as that of Section 3.1. As gradients backpropagate from the confound prediction heads to the encoder, we pass them through a *gradient reversal layer* in the style of Ganin et al. (2016) and Britz et al. (2017), which multiplies gradients by -1. If the cumulative loss of the target variables is $\mathcal{L}_t$ and that of the confounds is $\mathcal{L}_c$, then the loss which is implicitly used to train the encoder is $\mathcal{L}_e = \mathcal{L}_t - \mathcal{L}_c$, thereby encouraging the encoder to learn representations of the text which are not useful for predicting the confounds.

Lexicons are elicited from this model via the same mechanism as the Deep Residualizer of Section 3.1.

4 Experiments

We evaluate the approaches described in Sections 3 and 5 by generating and evaluating deconfounded lexicons in three domains: financial complaints, e-commerce product descriptions, and course descriptions. In each case the goal is to find words which can *always* help someone net a positive outcome (fulfillment, sales, enrollment), regardless of their situation. This involves finding words associated with *narrative persuasion*: predictive of human decisions or preferences but decorrelated from non-linguistic information which could also explain things. We analyze the resulting lexicons, especially with respect to the classic Aristotelian modes of persuasion: logos, pathos, and ethos.

We compare the following algorithms: **R**egression (R), **R**egression with **C**onfound features (RC), **M**ixed effects Regression (M), **R**esidualizing **R**egressions (RR), Log-**O**dds **R**atio (OR), **M**utual **I**nformation (MI), and MI/OR with regresssion (R+MI and R+OR). See Section 5 for a discussion of these baselines, and the online supplementary information for implementation details. We also compare the proposed algorithms: **D**eep **R**esidualization using word frequencies (DR+BOW) and embeddings (DR+ATTN), and **A**dversarial Selection using word frequencies (A+BOW) and embeddings (A+ATTN).

In Section 2 we observed that $\mathcal{I}(L)$ measures the improvement in predictive power that $L(T)$ affords a model already having access to C. Thus, we evaluate each algorithm by (1) regressing C on Y, (2) drawing a lexicon L, (3) regressing $C + L(T)$ on Y, and (4) measuring the size of gap in test prediction error between the models of step (1) and (3). For classification problems, we measured error with cross-entropy (XE):

$$XE = -\sum_i p_i \log \hat{p}_i$$

$$\text{performance} = XE_C - XE_{L(T),C}$$

And for regression, we computed the mean squared error (MSE):

$$MSE = \frac{1}{n}\sum_i (\hat{Y}_i - Y_i)^2$$

$$\text{performance} = MSE_C - MSE_{L(T),C}$$

Because we fix lexicon size but vary lexicon content, lexicons with good words will score highly under this metric, yielding the large performance improvements when combined with C.

We also report the average strength of association between words in L and C. For categorical confounds, we measure Cramer's V (V) (Cramér, 2016), and for continuous confounds, we use the

point-biserial correlation coefficient (r_{pb}) (Glass and Hopkins, 1970). Note that r_{pb} is mathematically equivalent to Pearson correlation in bivariate settings. Here the best lexicons will score the lowest.

We implemented neural models with the Tensorflow framework (Abadi et al., 2016) and optimized using Adam (Kingma and Ba, 2014). We implemented linear models with the scikit learn package (Pedregosa et al., 2011). We implemented mixed models with the lme4 R package (Bates et al., 2014). We refer to the online supplementary materials for per-experiment hyperparameters.

For each dataset, we constructed vocabularies from the 10,000 most frequently occurring tokens, and randomly selected 2,000 examples for evaluation. We then conducted a wide hyperparameter search and used lexicon performance on the evaluation set to select final model parameters. We then used these parameters to induce lexicons from 500 random train/test splits. Significance is estimated with a bootstrap procedure: we counted the number of trials each algorithm "won" (i.e. had the largest $error_C - error_{L(T),C}$). We also report the average performance and correlation of all the lexicons generated from each split. We ran these experiments using lexicon sizes of k = 50, 150, 250, and 500 and observed similar behavior. The results reported in the following sections are for $k = 150$, and the words in Tables 1, and 2, 3 are from randomly selected lexicons (other lexicons had similar characteristics).

4.1 Consumer Financial Protection Bureau (CFPB) Complaints

Setup. We consider 189,486 financial complaints publicly filed with the Consumer Financial Protection Bureau (CFPB)[2]. The CFPB is a product of Dodd-Frank legislation which solicits and addresses complaints from consumers regarding a variety of financial products: mortgages, credit reports, etc. Some submissions are handled on a timely basis (< 15 days) while others languish.

We are interested in identifying salient words which help push submissions through the bureaucracy and obtain timely responses, regardless of the specific nature of the complaint. Thus, our target variable is a binary indicator of whether the complaint obtained a timely response. Our confounds are twofold, (1) a categorical variable tracking the type of issue (131 categories), and (2) a categorical variable tracking the financial product (18 categories). For the proposed DR+BOW, DR+ATTN, A+BOW, and A+ATTN models, we set $|\boldsymbol{e}|$ to 1, 64, 1, and 256, respectively.

Results. In general, this seems to be a tractable classification problem, and the confounds alone are moderately predictive of timely response (XE_C = 1.06). The proposed methods appear to perform the best, and DR+BOW achieved the largest performance/correlation ratio (Figure 3).

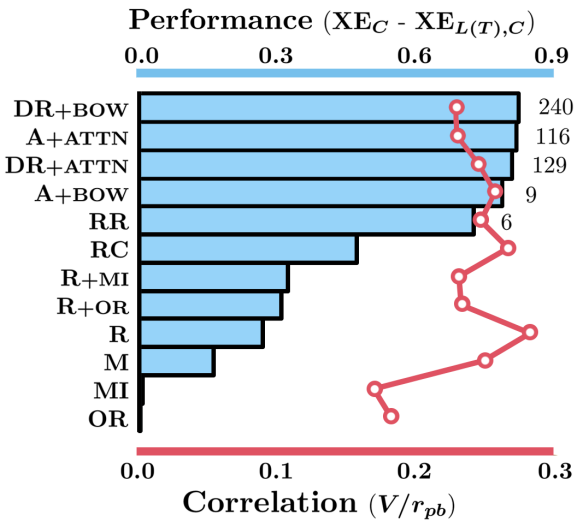


Figure 3: Predictive performance ($XE_C - XE_{L(T),C}$) and average confound correlation (V/r_{pb}) of lexicons generated via our proposed algorithms and a variety of methods in current use. The numbers to the right of each bar indicate the number of winning bootstrap trials.

DR+BOW	MI	RR	R
.	secondly	being	100
ma'am	forget	6	fargo
multiple	focus	issued	wells
guide	questions	agreement	.
submitted	battle	starting	fdcpa
'nt	vs	150.00	angry
honor	certainly	question	owe
,	contained	in	hipaa
xx/xx/xxxx	the	.	file
ago	be	agreement	across

Table 1: The ten highest-scoring words in lexicons generated by Deep Residualization + BOW (DR+BOW), Mutual Information (MI), Residialized Regression (RR), and regression (R).

[2]These data can be obtained from https://www.consumerfinance.gov/data-research/consumer-complaints/

We obtain further evidence upon examining the lexicons selected by four representative algorithms: proposed (DR+BOW), a well-performing baseline (RR), and two naive baselines (R, MI) (Table 1). MI's words appear unrelated to the confounds, but don't seem very persuasive, and our results corroborate this: these words failed to add predictive power over the confounds (Figure 3). On the opposite end of the spectrum, R's words appear somewhat predictive of the timely response, but are confound-related: they include the FDCPA (Fair Debt Collection Practices Act) and HIPAA (Health Insurance Portability and Accountability Act), which are directly related to the confound of financial product.

The top-scoring words in RR's lexicon include numbers ("6", "150.00") and words that suggest that the issue is ongoing ("being", "starting"). On the other hand, the words of DR+BOW draw on the rhetorical devices of ethos by respecting the reader's authority ("ma'am", "honor"), and logos by suggesting that the writer has been proactive about solving the issue ("multiple", "submitted", "xx/xx/xxx", "ago"). These are narrative qualities that align with two of the persuasion literature's "weapons of influence": reciprocation and commitment (Kenrick et al., 2005). Several algorithms implicitly favored longer (presumably more detailed) complaints by selecting common punctuation.

4.2 University Course Descriptions

Setup. We consider 141,753 undergraduate and graduate course offerings over a 6-year period (2010 - 2016) at Stanford University. We are interested in how the writing style of a description convinces students to enroll. We therefore choose log(enrollment) as our target variable and control for non-linguistic information which students also use when making enrollment decisions: course subject (227 categories), course level (26), number of requirements satisfied (7), whether there is a final (3), the start time, and the combination of days the class meets (26). All except start time are modeled as categorical variables. For the proposed DR+BOW, DR+ATTN, A+BOW, and A+ATTN models, we set $|e|$ to 1, 100, 16, and 64, respectively.

Results. This appears to be a tractable regression problem; the confounds alone are highly predictive of course enrollment (MSE_C = 3.67). (Figure 4). A+ATTN performed the best, and in general, the proposed techniques produced the most-predictive and least-correlated lexicons. Interestingly, Residualization (RR) and Regression with Confounds (RC) appear to outperform the Deep Residualization selector.

A+ATTN	R	OR
future	programming	summer
instructor	required	interpretation
eating	prerequisites	stability
or	computer	attitude
doing	management	optimization
guest	introduction	completion
sexual	chemical	during
culture	applications	labor
research	you	production
project	clinical	background

Table 2: The ten highest-scoring words in lexicons generated by Adversarial + ATTN (A+ATTN), Regression (R), and Log-Odds Ratio (OR).

In Table 2 we observe stark differences between the highest-scoring words of a proposed technique (A+ATTN) and two baselines with opposing characteristics (R, OR) (Table 2). Words chosen via Regression (R) appear predictive of enrollment, but also related to the confounds of subject ("programming", "computer", "management", "chemical", "clinical") and level ("required", "prerequisites", "introduction").

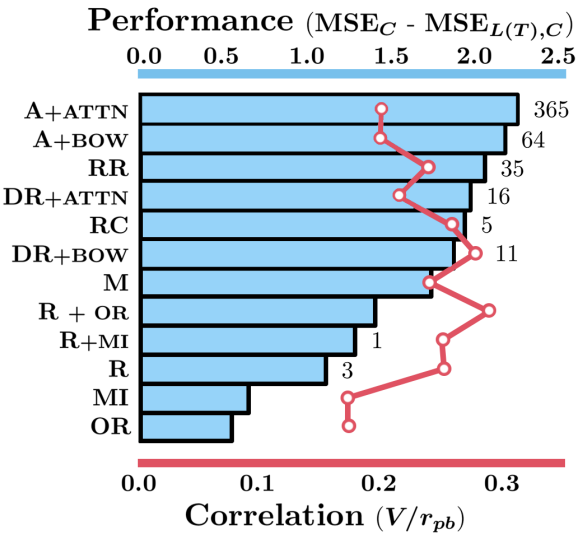


Figure 4: Course description comparative performance.

Log-Odds Ratio (OR) selected words which

A+BOW			RR		
word	transliteration	translation	word	transliteration	translation
ます	*masu*	polite suffix	７５		
プロテイン	*purotein*	protein	５		
お	*oh*	polite prefix	ニチバン	*nichiban*	adhesive company
粒	*tsubu*	grain	４		
栄養	*eiyō*	nutrition	群	*gun*	group
ご	*go*	polite prefix	サイズ	*saizu*	size
配合	*haigō*	formulation	摂取	*sesshu*	intake
デザート	*dezāto*	dessert	枚	*mai*	sheet
錠	*jō*	tablet	化学	*kagaku*	chemical
大豆	*daizu*	soy	ミニ	*mini*	mini

Table 3: The ten highest-scoring words in lexicons generated by Adversarial Selection + BOW (A+BOW) and Residualization (RR).

appear unrelated to both the confounds and enrollment. The Adversarial Selector (A+ATTN) selected words which are both confound-decorrelated and predictive of enrollment. Its words appeal to the concept of variety ("or", "guest"), and to pathos, in the form of universal student interests ("future", "eating", "sexual"). Notably, the A+ATTN words are also shorter (mean length of 6.2) than those of R (9.3) and OR (9.0), which coincides with intuition (students often skim descriptions) and prior research (short words are known to be more persuasive in some settings (Pratkanis et al., 1988)). The lexicon also suggests that students prefer courses with research project components ("research", "project").

4.3 eCommerce Descriptions

Setup. We consider 59,487 health product listings on the Japanese e-commerce website Rakuten[3]. These data originate from a December 2012 snapshot of the Rakuten marketplace. They were tokenized with the JUMAN morphological analyzer (Kurohashi and Nagao, 1999).

We are interested in identifying words which advertisers could use to increase their sales, regardless of the nature of the product. Therefore, we set log(sales) as our target variable, and control for an item's price (continuous) and seller (207 categories). The category of an item (i.e. toothbrush vs. supplement) is not included in these data. In practice, sellers specialize in particular product types, so this may be indirectly accounted for. For the proposed DR+BOW, DR+ATTN, A+BOW, and A+ATTN models, we set $|e|$ to 4, 64, 4, and 30, respectively.

[3]These data can be obtained from https://rit.rakuten.co.jp/data_release/

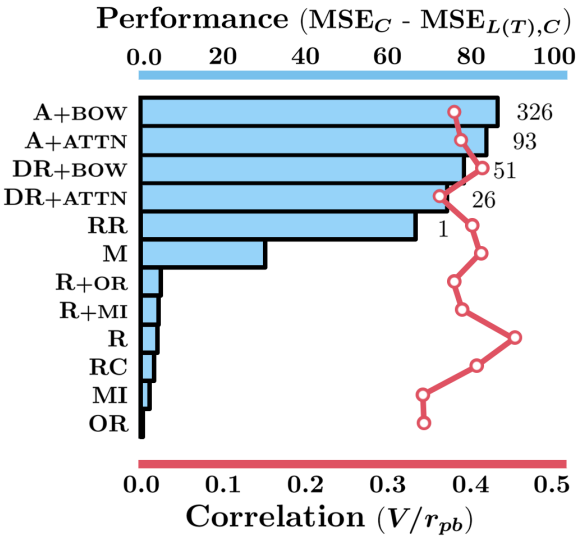


Figure 5: E-commerce comparative performance.

Results. This appears to be a more difficult prediction task, and the confounds are only slightly predictive of sales (MSE_C = 116.34) (Figure 5). Again, lexicons obtained via the proposed methods were the most successful, achieving the highest performance with the lowest correlation (Table 3). When comparing the words selected by A+BOW (proposed) and RR (widely used and well performing), we find that both draw on the rhetorical element of logos and demonstrate informativeness ("nutrition", "size", etc.). A+BOW also draws on ethos by identifying word stems associated with politeness. This quality draws on the authority of shared cultural values, and has been shown to appeal to Japanese shoppers (Pryzant et al., 2017). On the other hand, RR selected sev-

eral numbers and failed to avoid brand indicators: "nichiban", a large company which specializes in medical adhesives, is one of the highest-scoring words.

5 Related Work

There are three areas of related work which we draw on. We address these in turn.

Lexicon induction. Some work in lexicon induction is intended to help interpret the subjective properties of a text or make make machine learning models more interpretable, i.e. so that practitioners can know why their system works. For example, Taboada et al. (2011); Hamilton et al. (2016) induce sentiment lexicons, and Mohammad and Turney (2010); Hu et al. (2009) induce emotion lexicons. Practitioners often get these words by considering the high-scoring features of regressions trained to predict an outcome (McFarland et al., 2013; Chahuneau et al., 2012; Ranganath et al., 2013; Kang et al., 2013). They account for confounds through manual inspection, residualizing (Jaeger et al., 2009; Baayen et al., 2010), hierarchical modeling (Bates, 2010; Gustarini, 2016; Schillebeeckx et al., 2016), log-odds (Szumilas, 2010; Monroe et al., 2008), mutual information (Berg, 2004), or matching (Tan et al., 2014; DiNardo, 2010). Many of these methods are manual processes or have known limitations, mostly due to multicollinearity (Imai and Kim, 2016; Chatelain and Ralf, 2012; Wurm and Fisicaro, 2014). Furthermore, these methods have not been tested in a comparative setting: this work is the first to offer an experimental analysis of their abilities.

Causal inference. Our methods for lexicon induction have connections to recent advances in the causal inference literature. In particular, Johansson et al. (2016) and Shalit et al. (2016) propose an algorithm for counterfactual inference which bear similarities to our Adversarial Selector (Section 3.2), Imai et al. (2013) advocate a lasso-based method related to our Deep Residualization (DR) method (Section 3.1), and Egami et al. (2017) explore how to make causal inferences from text through careful data splitting. Unlike us, these papers are largely unconcerned with the underlying features and algorithmic interpretability. Athey (2017) has a recent survey of machine learning problems where causal modeling is important.

Persuasion. Our experiments touch on the mechanism of *persuasion*, which has been widely studied. Most of this prior work uses lexical, syntactic, discourse, and dialog interactive features (Stab and Gurevych, 2014; Habernal and Gurevych, 2016; Wei et al., 2016), power dynamics (Rosenthal and Mckeown, 2017; Moore, 2012), or diction (Wei et al., 2016) to study *discourse* persuasion as manifested in argument. We study *narrative* persuasion as manifested in everyday decisions. This important mode of persuasion is understudied because researchers have struggled to isolate the "active ingredient" of persuasive narratives (Green, 2008; De Graaf et al., 2012), a problem that the formal framework of deconfounded lexicon induction (Section 2) may help alleviate.

6 Conclusion

Computational social scientists frequently develop algorithms to find words that are related to some information but not other information. We encoded this problem into a formal task, proposed two novel methods for it, and conducted the first principled comparison of algorithms in the space. Our results suggest the proposed algorithms offer better performance than those which are currently in use. Upon linguistic analysis, we also find the proposed algorithms' words better reflect the classic Aristotelian modes of persuasion: logos, pathos, and ethos.

This is a promising new direction for NLP research, one that we hope will help computational (and non-computational!) social scientists better interpret linguistic variables and their relation to outcomes. There are many directions for future work. This includes algorithmic innovation, theoretical bounds for performance, and investigating rich social questions with these powerful new techniques.

7 Appendix: Causal Interpretation of the Informativeness Coefficient

Recall the definition of $\mathcal{I}(L)$:

$$\mathcal{I}(L) = \mathbb{E}\left[\text{Var}\left[\mathbb{E}\left[Y \middle| L(T),\, C\right] \middle| C\right]\right]$$

Here, we discuss how under standard (albeit strong) assumptions that are often made to identify causal effects in observational studies, we can interpret $\mathcal{I}(L)$ with $L(T) = T$ as a measure of the strength of the text's causal effect on Y.

Following the potential outcomes model of Rubin (1974) we start by imagining potential out-

comes $Y(t)$ corresponding to the outcome we would have observed given text t for any possible text $t \in \mathcal{T}$; then we actually observe $Y = Y(T)$. With this formalism, the causal effect of the text is clear, e.g., the effect of using text t' versus t is simply $Y(t') - Y(t)$.

Suppose that T, our observed text, takes on values in $\mathcal{T}$ with a distribution that depends on C. Let's also assume that the observed text T is independent of the potential outcomes $\{Y(t)\}_{t\in\mathcal{T}}$, conditioned on the confounders C (Rosenbaum and Rubin, 1983). So we know what would happen with any given text, but don't yet know which text will get selected (because T is a random variable). Now if we fix C and there is any variance remaining in $Y(T)$ (i.e. $\mathbb{E}\left[\text{Var}\left[Y(T)\middle|C, \{Y(t)\}_{t\in\mathcal{T}}\right]\right] > 0$) then the text has a causal effect on Y.

Now we assume that $Y(t) = f_c(t) + \epsilon$, meaning that the difference in effects of one text t relative to another text t' is always the same given fixed confounders. For example, in a bag of words model, this would imply that switching from using the word "eating" versus "homework" in a course description would always have the same impact on enrollment (conditionally on confounders). With this assumption in hand, then the causal effects of T, $\mathbb{E}\left[\text{Var}\left[Y(T)\middle|C, \{Y(t)\}_{t\in\mathcal{T}}\right]\right]$, matches $\mathcal{I}(L)$ as described in equation (1) (Imbens and Rubin, 2015). In other words, given the same assumptions often made in observational studies, the informativeness coefficient of the full, uncompressed text in fact corresponds to the amount of variation in Y due to the causal effects of T.

8 Acknowledgements

We gratefully acknowledge support from NSF Award IIS-1514268. We thank Youngjoo Chung for her invaluable assistance, advice, and the Rakuten data. We also thank Will Hamilton for his advice and direction while writing.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* .

Stephen Ansolabehere and Shanto Iyengar. 1995. *Going Negative: How Attack Ads Shrinks and Polarize the Electorate*. New York: Free Press.

Susan Athey. 2017. Beyond prediction: Using big data for policy problems. *Science* 355(6324):483–485.

R. Harald Baayen, Victor Kuperman, and Raymond Bertram. 2010. Frequency effects in compound processing. In *Compounding*, Benjamins, pages 257–270.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *4th International Conference on Learning Representations (ICLR)* .

Douglas Bates, Martin Maechler, Ben Bolker, and Steven Walker. 2014. lme4: Linear mixed-effects models using eigen and s4. *R package version* 1(7):1–23.

Douglas M Bates. 2010. lme4: Mixed-effects modeling with r.

Bruce L. Berg. 2004. *Methods for the social sciences*. Pearson Education Inc, United States of America.

Denny Britz, Reid Pryzant, and Quoc V. Le. 2017. Effective domain mixing for neural machine translation. In *Second Conference on Machine Translation (WMT)*.

V. Chahuneau, K. Gimpel, B. R. Routledge, L. Scherlis, and N. A. Smith. 2012. Word salad: Relating food prices and descriptions. In *Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

Jean-Bernard Chatelain and Kirsten Ralf. 2012. Fallacious liaisons: Near multicolinearity and "classical suppressors," aid policies, and growth. *Revue économique* 63(3):557–567.

Harald Cramér. 2016. *Mathematical Methods of Statistics (PMS-9)*, volume 9. Princeton university press.

Anneke De Graaf, Hans Hoeken, José Sanders, and Johannes WJ Beentjes. 2012. Identification as a mechanism of narrative persuasion. *Communication Research* 39(6):802–823.

John DiNardo. 2010. Natural experiments and quasi-natural experiments. In *Microeconometrics*, Springer, pages 139–153.

Naoki Egami, Christian J. Fong, Justin Grimmer, Margaret E. Roberts, and Brandon M. Stewart. 2017. How to make causal inferences using texts .

Pablo A. Estévez, Michel Tesmer, Claudio A. Perez, and Jacek M. Zurada. 2009. Normalized mutual information feature selection. *IEEE Transactions on Neural Networks* 20(2):189–201.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research* 17(59):1–35.

Andrew Gelman and Eric Loken. 2014. The statistical crisis in science data-dependent analysis—a "garden of forking paths"—explains why many statistically significant comparisons don't hold up. *American Scientist* 102(6):460.

Micheal W. Giles, Bethany Blackstone, and Richard L. Vining Jr. 2008. The Supreme Court in American democracy: Unraveling the linkages between public opinion and judicial decision making. *The Journal of Politics* 70(2):293–306.

Gene V. Glass and Kenneth D. Hopkins. 1970. *Statistical methods in education and psychology*. Prentice-Hall Englewood Cliffs, NJ.

Melanie C. Green. 2008. Research challenges: Research challenges in narrative persuasion. *Information Design Journal* 16(1):47–52.

Mark Grinblatt and Matti Keloharju. 2001. How distance, language, and culture influence stockholdings and trades. *The Journal of Finance* 56(3):1053–1073.

Mattia Gustarini. 2016. *Analysing smartphone users "inner-self": the perception of intimacy and smartphone usage changes*. Ph.D. thesis, University of Geneva.

Ivan Habernal and Iryna Gurevych. 2016. Which argument is more convincing? Analyzing and predicting convincingness of web arguments using bidirectional lstm. In *54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

William L. Hamilton, Kevin Clark, Jure Leskovec, and Dan Jurafsky. 2016. Inducing domain-specific sentiment lexicons from unlabeled corpora. *2016 Conference on Empirical Methods on Natural Language Processing (EMNLP)* .

Chin-Fu Ho and Wen-Hsiung Wu. 1999. Antecedents of customer satisfaction on the internet: An empirical study of online shopping. In *Systems Sciences, 1999. HICSS-32. Proceedings of the 32nd Annual Hawaii International Conference on*. IEEE, pages 9–pp.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural computation* 9(8):1735–1780.

Yajie Hu, Xiaoou Chen, and Deshun Yang. 2009. Lyric-based song emotion detection with affective lexicon and fuzzy clustering method. In *The International Society of Music Information Retrieval (ISMIR)*.

Kosuke Imai and In Song Kim. 2016. *When Should We Use Linear Fixed Effects Regression Models for Causal Inference with Longitudinal Data?*. Ph.D. thesis, Working paper, Princeton University, Princeton, NJ.

Kosuke Imai, Marc Ratkovic, et al. 2013. Estimating treatment effect heterogeneity in randomized program evaluation. *The Annals of Applied Statistics* 7(1):443–470.

Guido W. Imbens and Donald B. Rubin. 2015. *Causal inference in statistics, social, and biomedical sciences*. Cambridge University Press.

T. Florian Jaeger, Victor Kuperman, and Austin Frank. 2009. Issues and solutions in fitting, evaluating, and interpreting regression models. In *Talk given at WOMM presession to the 22nd CUNY Conference on Sentence Processing*.

Fredrik Johansson, Uri Shalit, and David Sontag. 2016. Learning representations for counterfactual inference. In *International Conference on Machine Learning (ICLR)*.

Jun Seok Kang, Polina Kuznetsova, Michael Luca, and Yejin Choi. 2013. Where not to eat? Improving public policy by predicting hygiene inspections using online reviews. In *Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

Denise M. Keele, Robert W. Malmsheimer, Donald W. Floyd, and Lianjun Zhang. 2009. An analysis of ideological effects in published versus unpublished judicial opinions. *Journal of Empirical Legal Studies* 6(1):213–239.

Douglas T. Kenrick, Steven L. Neuberg, and Robert B. Cialdini. 2005. *Social psychology: Unraveling the mystery*. Pearson Education New Zealand.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *3rd International Conference for Learning Representations (ICLR)* .

Sadao Kurohashi and Makoto Nagao. 1999. Japanese morphological analysis system juman version 3.61. *Department of Informatics, Kyoto University* .

Heeyoung Lee, Mihai Surdeanu, Bill MacCartney, and Dan Jurafsky. 2014. On the importance of text analysis for stock price prediction. In *International Conference on Language Resources and Evaluation (LREC)*.

Frank Luntz. 2007. *Words that work: It's not what you say, it's what people hear*. Hachette Books.

Daniel A. McFarland, Dan Jurafsky, and Craig Rawlings. 2013. Making the connection: Social bonding in courtship situations. *American journal of sociology* 118(6):1596–1649.

Saif M. Mohammad and Peter D. Turney. 2010. Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon. In *Proceedings of the NAACL HLT 2010 workshop on computational approaches to analysis and generation of emotion in text*. Association for Computational Linguistics, pages 26–34.

Burt L. Monroe, Michael P. Colaresi, and Kevin M. Quinn. 2008. Fightin'words: Lexical feature selection and evaluation for identifying the content of political conflict. *Political Analysis* 16(4):372–403.

Brian C.J. Moore. 2012. *An introduction to the psychology of hearing*. Brill.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12(Oct):2825–2830.

Anthony R. Pratkanis, Anthony G. Greenwald, Michael R. Leippe, and Michael H. Baumgardner. 1988. In search of reliable persuasion effects: III. The sleeper effect is dead: Long live the sleeper effect. *Journal of personality and social psychology* 54(2):203.

Reid Pryzant, Young-joo Chung, and Dan Jurafsky. 2017. Predicting sales from the language of product descriptions. In *Special Interest Group on Information Retrieval (SIGR) eCommerce Workshop*.

Rajesh Ranganath, Dan Jurafsky, and Daniel A. McFarland. 2013. Detecting friendly, flirtatious, awkward, and assertive speech in speed-dates. *Computer Speech & Language* 27(1):89–115.

Paul R. Rosenbaum and Donald B. Rubin. 1983. The central role of the propensity score in observational studies for causal effects. *Biometrika* 70(1):41–55.

Sara Rosenthal and Kathleen Mckeown. 2017. Detecting influencers in multiple online genres. *ACM Transactions on Internet Technology (TOIT)* 17(2):12.

Donald B. Rubin. 1974. Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of Educational Psychology* 66(5):688.

Simon J.D. Schillebeeckx, Sankalp Chaturvedi, Gerard George, and Zella King. 2016. What do I want? The effects of individual aspiration and relational capability on collaboration preferences. *Strategic Management Journal* 37(7):1493–1506.

Uri Shalit, Fredrik Johansson, and David Sontag. 2016. Estimating individual treatment effect: Generalization bounds and algorithms. *34th International Conference on Machine Learning (ICML)* .

Donald P. Spence. 1983. Narrative persuasion. *Psychoanalysis & Contemporary Thought* .

Christian Stab and Iryna Gurevych. 2014. Identifying argumentative discourse structures in persuasive essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Magdalena Szumilas. 2010. Explaining odds ratios. *Journal of the Canadian Academy of Child and Adolescent Psychiatry* 19(3):227.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational linguistics* 37(2):267–307.

Chenhao Tan, Lillian Lee, and Bo Pang. 2014. The effect of wording on message propagation: Topic- and author-controlled natural experiments on Twitter. *Annual Meeting of the Association for Computational Linguistics (ACL)* .

Tom Van Laer, Ko De Ruyter, Luca M. Visconti, and Martin Wetzels. 2013. The extended transportation-imagery model: A meta-analysis of the antecedents and consequences of consumers' narrative transportation. *Journal of Consumer research* 40(5):797–817.

Rob Voigt, Nicholas P Camp, Vinodkumar Prabhakaran, William L. Hamilton, Rebecca C. Hetey, Camilla M. Griffiths, David Jurgens, Dan Jurafsky, and Jennifer L. Eberhardt. 2017. Language from police body camera footage shows racial disparities in officer respect. *Proceedings of the National Academy of Sciences* .

Zhongyu Wei, Yang Liu, and Yi Li. 2016. Is this post persuasive? Ranking argumentative comments in the online forum. In *The 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Lee H. Wurm and Sebastiano A. Fisicaro. 2014. What residualizing predictors in regression analyses does (and what it does not do). *Journal of Memory and Language* 72:37–48.

Detecting Denial-of-Service Attacks from Social Media Text: Applying NLP to Computer Security

Nathanael Chambers and **Ben Fry** and **James McMasters**
Department of Computer Science
United States Naval Academy
nchamber@usna.edu

Abstract

This paper describes a novel application of NLP models to detect denial of service attacks using only social media as evidence. Individual networks are often slow in reporting attacks, so a detection system from public data could better assist a response to a broad attack across multiple services. We explore NLP methods to use social media as an *indirect* measure of network service status. We describe two learning frameworks for this task: a feed-forward neural network and a partially labeled LDA model. Both models outperform previous work by significant margins (20% F1 score). We further show that the topic-based model enables the first fine-grained analysis of how the public reacts to ongoing network attacks, discovering multiple "stages" of observation. This is the first model that both detects network attacks (with best performance) and provides an analysis of when and how the public interprets service outages. We describe the models, present experiments on the largest twitter DDoS corpus to date, and conclude with an analysis of public reactions based on the learned model's output.

1 Introduction

Distributed Denial of Service (DDoS) attacks have become more frequent and more severe in their impact. Coordinated attacks across several services are now common, yet there are fewer methods to detect multi-network events. Research into detecting and preventing single attacks focuses on *direct* evidence based on characteristics of a network itself, such as monitoring abnormal traffic. This paper instead investigates an aytpical source for multiple attacks with *indirect* evidence: social media text. Do users of attacked systems post on social media? What can be learned from comments? Can NLP learning models extract enough information from user posts to detect attacks? Previous work on attack detection with social media is sparse, and focused on detecting trending words. This paper is the first to learn models of language without 'attack' dictionaries and seed words. The goal is the real-time detection of attacks without network data. Our secondary goal is to illustrate NLP applications to computer security topics.

Research on information extraction from social media has shown that many types of events in the world can be reliably detected from the language that users post. Several approaches have been shown effective in identifying events like earthquakes (Sakaki et al., 2010), concerts and product releases (Ritter et al., 2012), and other natural disasters (Neubig et al., 2011). Detecting DDoS attacks is not too dissimilar from these goals. An attack is a real event in the world, and it takes a community by surprise. This paper thus adopts ideas from NLP, but applies them to the unique application of DDoS detection.

Social media is obviously not the only way (nor the most direct) to monitor network services and attacks. There are several commercial services that directly measure outages, such as norsecorp[1]. These perform *direct* monitoring of network response. We do not propose social media as a better alternative, but rather as an alternative that enhances direct monitoring. Social media also brings its own unique benefits. For instance, social media does not require a priori knowledge of which networks should be monitored. It can also help detect "soft" outages like slowdowns and account blockages, things that direct monitoring cannot always detect. Therefore, this paper is not suggesting a replacement, but rather a new source of valuable information. It is a monitoring architecture that is not constrained by a predefined list of services.

Our goal in using social media is driven by the

[1] http://map.norsecorp.com

Proceedings of NAACL-HLT 2018, pages 1626–1635
New Orleans, Louisiana, June 1 - 6, 2018.

hypothesis that as a network attack unfolds, its users go through a series of observational stages that can be automatically learned and detected. The first stage is a state of confusion and basic symptom observation, as seen in the following real tweets from Twitter:

> *hey linode what's happening? I can't login, my servers are down and you don't reply on mails?*
>
> *is xbox live experiencing some issues?*

These tweets don't discuss an attack even though that is what was occurring. Later stages then develop into direct commentary as the community coelesces to a belief that an attack is the cause:

> *Breaking: Band of America website rumored to be under DDoS attack.*
>
> *Citi Bank & BofA under Massive DDoS attack*

We show that our proposed LDA-based model can effectively identify these stages. There is very little previous work in this area, and that which exists focuses entirely on the second stage. Ritter et al. (2015) proposed models that include hard-coded keywords like 'DDoS' and phrases like '<entity> is down'. Their work helped identify attacks with social media, but these identifications tend to be *after* the news has already reported it. Early symptoms that are discussed don't use words like 'DDoS' because the conclusion has not yet been drawn. We thus propose the first learning models that identify *early* attack discussions: the first is a neural network, and the second is a broader topic model that provides better insight into the evolution of an attack.

Finally, our last goal is to model the themes and topics that users notice during network attacks. This is a somewhat subjective analysis, but it is backed by an empirical model trained from real-world data. Not only do we empirically produce state-of-the-art results, 25% gains over previous work, we also show that our detection system learns topics of discussion previously uninvestigated in the security field.

The core contributions of this paper are as follows: (1) a *25% improvement* over previous work on attack detection, (2) we present the first neural network results on detecting network attacks from social media, (3) we present a partially labeled LDA model for detecting network attacks with state-of-the-art results, (4) the PLDA enables the first analysis of the evolution of an attack as seen through its users, and (5) we make available the largest list of historical DDoS attacks to date.

2 Previous Work

The most relevant line of research to this paper is *event extraction* from social media. Space prohibits describing all work; the major approaches vary in levels of supervision. Ritter et al. (2012) used a Latent Dirichlet Allocation model to identify events in text *without* labeled data. They showed you can cluster and extract events like concerts, movies, and performances into a calendar. General event detection from social media has continued in several threads (Benson et al., 2011; Popescu et al., 2011; Anantharam et al., 2015; Wei, 2016; Zhou et al., 2017). Guo et al. (2013) link tweets to news stories using an annotated dataset. Sakaki et al. (2010) detect earthquake events by monitoring tweets with keywords like 'earthquake'. This is similar in goal to our paper, but different in approach and brittle in its application. We crucially do *not assume* that users use known keywords and phrases.

We take inspiration from the thread of work on flu detection (Lamb et al., 2013; Broniatowski et al., 2013). Their work leverages mentions of an event ('caught', 'sick', 'flu'), and then uses human annotators to label these mentions as relevant to the desired event (flu). We also identify mentions of an event, but we crucially differ by *not knowing* event words a priori. We believe a typical user does not know what a DDoS attack is, so we cannot assume certain language will be used. A major contribution of this work is the first analysis of how the (perhaps uninformed) public perceives DDoS attacks as they occur.

The first work (to our knowledge) on attack detection from Twitter was Motoyama et al. (2010). They tracked a single phrase "X is down" and experimented with whether outages could be detected from its counts. They use a trend detection formula to notice increases of this one phrase to trigger an alert. We compare against this strong baseline later. The work by Kergl et al. (2016) uses social media to identify users who discuss zero-day exploits. While not directly related to the work in this paper, its success reinforces the hypothesis that social media contains useful data for computer security monitoring.

The main thread in this area is the learning model from Ritter et al. (Ritter et al., 2015) and follow-on work (Kergl, 2015; Chang et al., 2016). They proposed a weakly supervised learner to identify cybersecurity events from Twitter. They

Attacked Services (dd-mm-yy)			
Ancestry.com	16-06-14	Lib. Congress	18-07-16
BBC Website	14-03-15	Newsweek	29-09-16
Call of Duty	20-09-14	Planned Parent.	29-07-15
DNS	21-10-16	Reddit	19-04-13
Github	27-03-15	Spamhaus	18-03-13

Table 1: A sample of 10 DDoS events in our dataset. A 20 day span is collected around each attack date.

collected tweets that contain the word *'DDoS'*, and then collected a set of known network attack days. The known days provided a training set from which they trained this weakly-supervised classifier on the 'DDoS' tweets. An important constraint in their approach, similar to flu research, is the need to use a seed word(s). Seed words enable the collection of a very relevant training set, but it limits the system because it depends on social media posts to use these words, and more importantly, to actually know that a DDoS is happening. We instead hypothesize that attacks are preceded by users who first observe symptoms of the attack, and don't directly discuss a DDoS or use related attack words. Our analysis shows we match several orders of magnitude more tweets.

3 Datasets

We manually created a dataset of historical DDoS attacks that include the entity attacked and the date of attack. Most past attacks are difficult to identify hour ranges, so we used a full 24-hour day as our granularity. We included 6 attacks with sufficient volume from previous work (Ritter et al., 2015), but we grew this set to 50 attacks based on our own investigations into recent years, mostly through web search results for 'DDoS attacks'. Table 1 lists *some* of these for illustration of its diversity. The full list is at `www.usna.edu/Users/cs/nchamber/data/ddos/`

For each of these known attacks, we collected tweets that contained the attacked entity's name in a 20-day period: 17 days prior to the attack, 1 day on the attack, and 2 days following. We wanted a sufficient lead up to the attack to include previous work's trending model (Motoyama et al., 2010), and to provide non-attack days for evaluation. The days surrounding the known attack date are labeled NOT-ATTACK, and the attack day itself as ATTACK. Sometimes an attack lasted longer than a single day, in which case the days following were also labeled as ATTACK, as appropriate.

The historical attacks span the years 2012-2016. We split the data so that years 2012, 2015, and 2016 comprise the training set, 2013 is the development set, and the year 2014 is the test set only used for computing final experiment numbers. Splitting on years (rather than months or entities) guards against test set pollution into our training set. The evaluation on the 2014 test set is thus an unbiased experiment because nothing from the entire year is included in training. For the experiments, we use the union of training and development to train the final models that are then used to evaluate on the test year 2014. There are 200 test days in 2014, 50 in dev, and ˜500 in training.

The full dataset consists of 50 attack days over approximately 800 days and 2 million tweets. The only previous work on this area used seed words to pull out around 9-10 thousand tweets. Our dataset is *more than 2 orders of magnitude larger*. The reason is due to the larger number of attacks we collected, but notably our tweets are more diverse and varied because we don't require hard-coded target words and phrases to match.

Formally, the dataset is 800 labeled datums:

$$d_i = (Entity, Date, Tweets, Label) \quad (1)$$

where $d_i \in D$ and D is the set of all days. $Entity$ is the attacked network service, $Date$ is the calendar date, and $Tweets$ are all tweets on that date mentioning that entity. $Label$ is a binary variable: ATTACK or NOT-ATTACK. Even though the day following an attack often includes attack discussion, it is still labeled NOT-ATTACK. Only if the attack was ongoing is the next day labeled ATTACK.

4 Models

Two primary goals motivate the models we propose and evaluate. The first goal is the automatic classification of attack and non-attack events. We propose the first neural network for this task, and move on to a generative model based on topic models. We evaluate their relative performance and compare against baselines from prior work.

The second goal is a model that enables analysis of user behavior during the evolution of an attack. What do people notice? What do people focus on? These are important questions for the security community that NLP models can help answer. We present a brief subjective study using the generative model, show how learned topics change over time, and discuss the data's implications.

4.1 Task Formulation

As discussed in Section 3, our input is labeled datums: $d_i = (Entity, Date, Tweets, Label)$. Each datum in the training set has a known label of ATTACK or NOTATTACK based on our historical knowledge of which entities were attacked on which days. We thus formulate the task as a binary classification over 24 hour days. We train models with the labeled training set, and report final numbers on test. In order to tune parameters, we use the development set to run grid searches over the models' parameters. The test set was always excluded from these until the final experiments.

4.2 Logistic Regression

Our first baseline model is logistic regression with word-based features. The following were used:

Unigrams. All words in the tweets were lowercased and punctuation stripped.
Bigrams. All bigrams are included & lowercased. Start and stop symbols are used for tweet boundaries, and punctuation included as separate tokens.
Bigram/Trigram Patterns. Since we know the entity, we parameterize the entity's mention in each tweet, and build bigrams and trigrams around them. For instance, the phrase "reddit is slow" is included as a trigram feature "X is slow". This allows learning across instances, so "spamhaus is slow" is included as the same feature.

We use the Stanford CoreNLP toolkit with default settings to train the model. We removed all features that occurred only once. This model is referred to as **LogisticReg** below.

4.3 Neural Network Models

Neural networks have made significant advancements in many NLP areas. Two of the main reasons for this are (1) improved representation of the features, and (2) stacking of hidden layers provides a better data fit.

We experimented with two feed-forward neural networks using word embeddings. We first trained a simple one-layer neural network that is similar to logistic regression, but with embeddings as input (instead of frequency counts). This is the **Neural-1** model. We then trained a two-layer network with hidden layer h of size m, and a softmax output layer to the binary label task. This is the **Neural-2** model.

The input to both of these models is as a Continuous Bag of Words (CBOW) model (Mikolov et al., 2013). Unlike logistic regression, the only features input to the network are unigrams (a tweet's individual tokens). Each unigram u has a word embedding x_u of length n, and they are all input as a weighted average. The reader is referred to Mikolov (2013) for more CBOW background.

We do not use pre-trained word embeddings, but instead learn them from our data. The embedding values are initialized randomly $[0, 1]$ from the uniform distribution. We used DyNet as our modeling toolkit (Neubig et al., 2017).

Overfitting is often a problem with neural networks, and we quickly found our models doing so. We thus applied 0.5 dropout for regularization (Srivastava et al., 2014). We experimented with other dropout values but did not see reliable gains or losses, so kept it at the typical 0.5 value.

We trained other networks without word embeddings, but instead "one-hot vectors" where the vector is the size of the vocabulary. This model did not perform as well and required more memory, so we do not report its results. Additional hidden layers did not improve either, as expected from the observed overfitting.

4.4 Constrained Topic Modeling

While the neural models above improve over previous work and baselines, they are difficult to interpret what is actually learned. One of the applications of this paper is to analyze what people discuss during network attacks. The hidden layers and word embeddings are opaque and difficult from which to draw conclusions.

In contrast, a generative model that represents words explicitly as probability distributions allows for easier post-analysis. It also may generalize better to this task because training data is more sparse and noisy. While we have 2 million tweets, orders of magnitude more than previous work, this is still modest in size with 800 days. To make matters worse, the dataset is biased toward NOTATTACK. 95% of the training set is NOTATTACK, leaving few training instances that are actually labeled as ATTACK. As shown in the next section, the neural models tend to overfit to these small signals. Further, we observed that online discussions go through different stages (Section 5.4), and the neural model merges stages to its detriment.

We thus propose a model inspired by Latent Dirichlet Allocation (LDA) (Blei et al., 2003), but a model carefully designed to the unique applica-

tion at hand. For readers unfamiliar with LDA, the model can be thought of as a clustering algorithm, and an overview of LDA and its variants can be found in Blei's survey (Blei, 2012).

4.4.1 LDA for Security Events

A traditional LDA model can learn general topics on our dataset with the hope that attack topics bubble up. Our initial experiments found this to be insufficient and the non-attack days were full of distracting topics.

For the goal of analyzing attack discussion, we need to encourage the LDA model to learn attack-specific topics. We draw heavily from Labeled LDA (Ramage et al., 2009). Each word is assigned a topic as in standard LDA, but topics can have a known label from the document. This is relevant to this paper because we know which days are attacks (in training). Thus, when a tweet is on an attack day, we assign the tweet a label ATTACK, and bias the Labeled LDA learner to assign its words to an attack-related topic.

What labels do we have in our data? ATTACK and NOTATTACK labels are first, but we also know which entities are mentioned in tweets, providing labels to learn entity-specific topics. We can label a tweet about reddit as REDDIT, and bias the Labeled LDA algorithm to assign a reddit-specific topic. The following tweet is an example:

reddit isn't responding maybe DNS is wrong

This tweet mentions two entities (reddit and dns), and it occurs on a known attack day for reddit in training. This tweet thus has 3 labels (attack, reddit, dns). The tweet's tokens can draw from 1 of 3 topics, which is good but a bit constraining. One of the premises of this paper is that people discuss attacks on social media in a variety of ways (not just one topic). They might discuss hackers, the DDoS attack itself, or just general downtime. The vanilla Labeled LDA (Ramage et al., 2009) is then too strict, but there is a multi-topic extension in the Partially Labeled Dirichlet Allocation (PLDA) (Ramage et al., 2011). PLDA is a version that instead of having one topic per label, it learns N_l topics for each label l. For our example tweet, tokens can now be labeled with 1 of $\sum_l N_l$ topics. We use $N_{attack} = 5$ and $N_{reddit} = N_{dns} = 5$ in our experiments[2], so this tweet would sample from 15 topics.

[2]This is about reddit, but each company has its own 5 topics. Experiments have 40 companies for total 200 topics.

Formally, let a tweet be defined as a document d with words $w \in W_d$. Each document has a set of labels Λ. This set Λ always contains the BACKGROUND label to capture general twitter conversations. Further, if a network or company is mentioned in the document, Λ also contains the company's label (e.g., MICROSOFT). Finally, the label ATTACK is added to Λ if d is an attack day and W_d includes the attacked network's name. Each word $w \in d$ has a latent label l and a latent topic z_l. For readers familiar with plate diagrams, this diagram is shown in Figure 1. Readers will notice its similarity to Ramage et al. (2009) with the addition of a new β parameter and the important change that the attack label is *observed* in training, but *unobserved* in test. When observed (in training), we favor assigning words to attack topics. When unobserved, we want to dissuade but still allow for it when the text strongly favors attack topics. To this end, our PLDA differs from standard use in that ψ is generated from a non-symmetric dirichlet with hyperparameters $\boldsymbol{v}$ (a vector of length $\sum_l N_l$) defined as:

$$v_i = \begin{cases} \alpha, & \text{if } i \notin AttackTopics \\ \beta, & \text{if } i \in AttackTopics \ \ \& \ \ attack \notin \Lambda \\ \beta * 10, & \text{if } i \in AttackTopics \ \ \& \ \ attack \in \Lambda \end{cases}$$

This is a non-symmetric dirichlet prior that enables attack labels to be chosen (β) without an observed attack day. Every tweet must be able to sample from attack topics because we need to label future unknown (unlabeled) attacks. The PLDA in the literature assumes full labeling at all times, but our task is more difficult. When ATTACK is observed, its smoothing parameter's value is $\beta * 10$ because of our heightened certainty, rather than simply β when unobserved.

The number of attack topics N_a and background topics N_b was chosen empirically from dev set performance. For simplicity and to avoid overfitting, we chose a single number $M = 5$ of company topics $\forall_c \, N_c = M$ that is the same across all companies and also $N_{attack} = M$. Only N_b was varied in our parameter tuning stage to discover how many background topics were necessary.

Space prohibits a full mathematical description of PLDA, so we direct the reader to Ramage et al. (2011) for details. Those unfamiliar with the above formalities can think of it as a soft clustering of words that is accomplished through sampling.

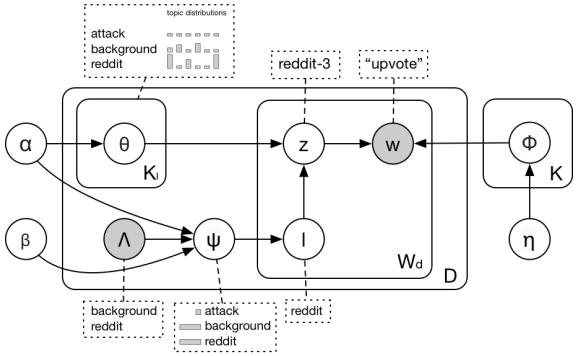


Figure 1: Plate diagram of the partially labeled topic model (PLDAttack). Dotted boxes are example values of a non-attack tweet discussing Reddit.

4.4.2 Inference and Classification

Inference in this model is performed with collapsed Gibbs sampling, sampling l and z_l in turn while holding all other variables constant. A single iteration requires looping over the entire dataset and assigning labels (topics) to each token on each day. We repeat this process until convergence of the joint probability of the model. After convergence, we hold distributions θ, ψ constant and run 20 more sampling iterations. Each word is then assigned the topic that was sampled the most in the 20 iterations.

Once sampling completes, this PLDAttack model provides us two very useful tools. First, the assigned topics enables us to use it as a classifier for our target task: DDoS detection from social media. Second, the topics themselves allow us to create timelines of discussions about DDoS attacks. This provides a higher-level analysis of what people say (Learned Topics).

With all words labeled, we want the model to make a prediction about an entity e on a given day d. Was the entity attacked[3]? We compute the probability of an attack using the labels themselves without any modification:

$$P(attack|d) = \frac{\sum_{w \in W_d} 1\{z_w \in Attack\}}{|W_d|} \quad (2)$$

where $|W_d|$ is the number of words in all tweets on day d and $Attack$ is the set of attack topics. $1\{x\}$ is the indicator function. If this probability is greater than a threshold, the entity/day is labeled as an attack. Otherwise, it is not an attack.

The cutoff threshold depends on a typical probability that is assigned to tweets, and how frequent an entity is actually mentioned on a given day. We use the development set to identify the optimal cutoff to maximize our F1 score.

[3]Or, is the entity currently under attack? This paper is an early detection attempt, leaving live-tracking to future work.

5 Evaluation and Results

All experiments are conducted on the dataset described in Datasets. The task is a binary classification of ATTACK or NOTATTACK given a day of tweets. All parameters are optimized on the development set: we treat attack days as known on training days, but hidden from the development and test days. We calculate F1 score on the development attack days, and optimize parameters using a basic grid search. For the final reported results, we combine train+dev into one observed training set, and the test set is now included in sampling, but with unobserved attack days. Since the PLDAttack model is probabilistic, all reported numbers are an average of 10 independent runs.

We use ATTACK F1 as the main evaluation target; the harmonic mean between precision and recall. Applications overly concerned with missing attacks would optimize to recall R. We chose $F1$ as a happy balance between a quality classifier (good precision P) and a useful classifier (good recall R). We report all three scores for both the ATTACK and NOTATTACK labels, but optimize to F1 during parameter search on the development set.

5.1 Trending Baselines

Entity Trending: This baseline follows the hypothesis that a website under attack is mentioned more than usual, and language analysis is not required. There is credence to this idea. Much of our data includes a spike in discussion on the attack day (however, some non-attack days show similar frequency spikes). We model frequency trending with an exponential decay function similar to that in Motoyama et al. (2010). It uses an Exponentially Weighted Moving Average:

$$A_t = \alpha * n_t + (1 - \alpha) * A_{t-1} \quad (3)$$

where A_t is the EWMA of day t, n_t is the number of tweets on day t, and α determines how the current day's count affects the moving average. We then need a threshold T_t to determine when n_t is trending. This is based on a moving deviation σ^2:

$$D_t = n_t - A_{t-1} \quad (4)$$

	Non-Attack			Attack		
	P	R	F1	P	R	F1
Freq Baseline	.99	.87	.92	.29	**.83**	.43
Motoyama'10	.97	.94	.95	.35	.58	.44
LogisticReg	.97	.75	.85	.14	.67	.24
Neural-1	.96	.97	**.96**	.54	.47	.49
Neural-2	.97	.96	**.96**	.55	.53	.53
PLDAttack	.96	.96	**.96**	**.61**	.52	**.55**

Table 2: Results on the held-out test set of 200 test datums. Motoyama'10 is a trending phrase baseline.

$$\sigma_t^2 = \beta * D_t^2 + (1 - \beta) * \sigma_{n-1}^2 \quad (5)$$

Given this deviation, the threshold is then:

$$T_t = M_{t-1} + \epsilon * \sigma_{t-1} \quad (6)$$

If $n_t > T_t$ for a day, we signal an ATTACK.

Pattern Trending: This modified baseline exactly duplicates Motoyama et al. (2010). Their approach looks for trending mentions that match the pattern, *'X is down'*. The X is substituted with the company's name. We use the same equation 6, but frequency n_t is defined as how many tweets contain the pattern (instead of just 'X').

5.2 Experiment Results

The test set results for baselines and models are shown in Table 2. All improvements are statistically significant as indicated using McNemar's two-tailed test. The trending baselines have high recall. When an attack is happening, the network does indeed trend on social media. Precision is low, however, because non-security events also cause discussions. The neural models outperform the baselines, and a hidden layer (Neural-2) is definitely needed for increased detection. The training set of 500 documents is still small for neural training, though. Neural models have many parameters, and they overfit to our training set despite regularization with dropout, reducing dimensions, and removing hidden layers. Even still, we improved over the Motoyama baseline by 20% relative F1.

PLDA (PLDAttack) showed the highest precision when classifying an ATTACK. Since its recall was similar to the neural models, it produced the best F1 score. This is a 25% relative improvement over previous work. PLDAttack generalizes to the dataset slightly better than the neural models.

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
issues	after	attack	down	down
working	service	ddos	servers	anonymous
having	#news	under	site	megaupload
issue	hit	twitter	hacked	takes
email	attack	dns	website	sites
update	website	attacks	goes	music
services	hackers	massive	taking	claims
problem	#tech	services	web	doj

Table 3: The top words in each of 5 attack topics learned from the entire train/test dataset.

We now have two good approaches for detecting attacks: neural models and topic modeling. The remaining question is to analyze what people are actually discussing, and it is here that topic modeling further shines.

5.3 Learned Topics

The generative model is attractive because we can use its learned distributions to produce insight into what people discuss. It is more precise in our experiments, and the neural models are simply too difficult to analyze.

Table 3 shows some of the attack topics that were learned on one of our model's runs (results are an average of training runs). As can be seen, though topics are similar, they capture subtle differences in what people discuss during an attack. The first topic represents tweets about news surrounding the event. These often contain links, and show up after the attack is made known to news agencies. In contrast, topic 3 is more general about servers down and specific services such as email. The fourth topic captures discussion about Anonymous and the claims that the group makes about taking sites down. This was obviously learned as an artifact of our data which contains several Anonymous-related events.

One of the most useful analyses we can do with this type of model is track topic evolution over time. Figure 2 illustrates one attack day and the dramatic jump of the attack topics. For simplicity, we plot the 5 attack topics, and hide the others as they are generally flatter across the bottom. This shows that social media became aware on the 9th hour, and only took one more hour to reach peak intensity. What is perhaps most useful with a timeline is understanding the impact of an attack on its users. There is a fair bit of chatter the day following the event, showing that people do not easily forget such attacks and depending on the entity, this could have effects on how people engage. We

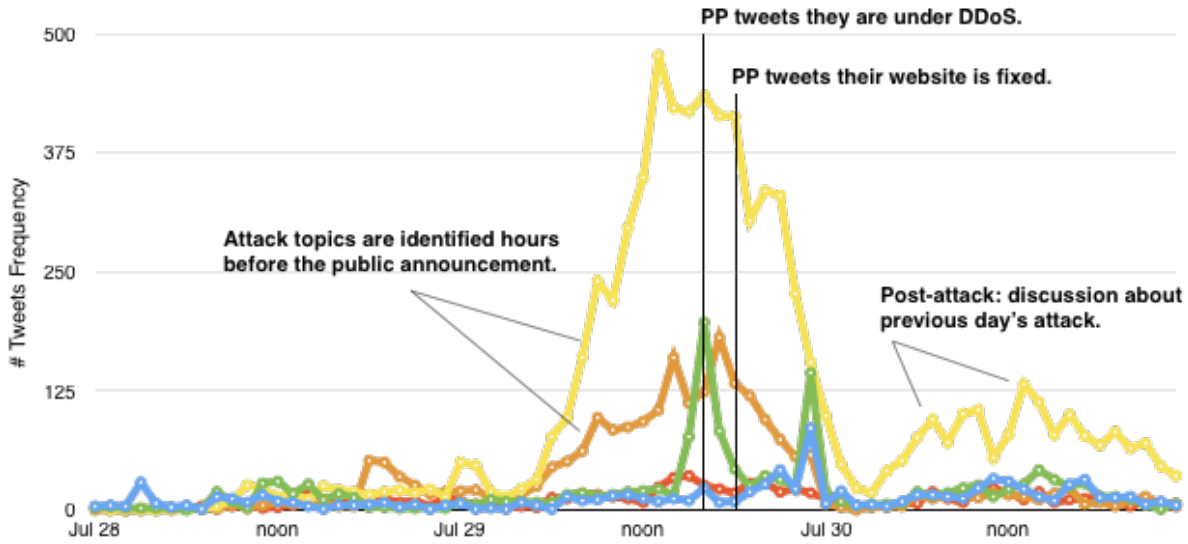


Figure 2: Attack topics during a 3-day period around a DDoS on the Planned Parenthood (PP) website. The attack was announced at 5pm on July 29. The green topic spikes with PP's first tweet about the DDoS. The yellow & brown topics rise hours **before** the announcement. They decrease after the attack, similar to the red 'news' topic.

also see that Planned Parenthood (in this example) delayed in announcing the attack. Whether this is tactical or simply how long it took to realize the event, PLDAttack offers a natural way to discover just how soon patrons (or the public in general) became aware of the issue without an announcement. Decision making around these events might be guided by helpful NLP tools such as this.

Finally, we note previous work tracked tweets with 'DDoS' in it. There were ˜50 such tweets, but our models instead matched *tens of thousands*. Previous seed-based work cannot produce this type of analysis.

5.4 Attack Stages in Online Discussions

To analyze the PLDAttack model's strengths, we split attack days into "spiking" chunks to identify common stages of online chatter. We use topic frequency spikes for Reddit to provide diversity in analysis from the Planned Parenthood Figure 2. Reddit is a community based website attacked on April 19, 2013. We identified four distinct stages of a DDoS attack on social media: (1) Symptom, (2) Inference, (3) Confirmation, and (4) Resumption. Figure 3 shows examples from each stage.

The **Symptom Stage** is the earliest sign of a problem with user observations of the network service. These aren't comments about malicious attacks, but statements about authentication problems and unresponsive websites. This is the most difficult stage for a learner (**false positives**). Services can have trouble for a variety of reasons, not necessarily DDoS attacks. Some of our evaluation data includes inoccuous problems, and these caused a decrease in precision.

The **Inference Stage** includes guesses about the cause of the previous stage's symptoms. These can and do intermix with the Symptom Stage. As seen in the reddit examples in Figure 3, some of the users wonder if they broke reddit rather than a malicious act occurring. We also see an example of someone guessing that it is a DDoS attack, but without actual knowledge of it.

The **Confirmation Stage** occurs when the website publicly announces an attack. Not all attacks have a public announcement. Our error analysis revealed this to be the cause of several **false negatives**. When the public is not directly informed, the learning algorithms must rely on symptoms and inferences only. Previous work largely isolated itself to attacks with a Confirmation Stage, for instance, relying on the 'DDoS' keyword to be present (Ritter et al., 2015).

Finally, the **Resumption Stage** is when the network service is restored. The reddit examples show people commenting on the resumption, and making jokes about the previous situation. Similar to the Symptom Stage, this stage contributes to **false positives** because it also occurs with normal routine network problems, not just malicious acts.

5.5 Error Analysis

Identifying the four stages above led to a natural method of studying errors in our model. We

Symptom Stage
Please come back Reddit! I'm bored.
Reddit won't authenticate me. #lifeisover
Reddit is calling me a robot and won't let me use it

Inference Stage
There is a DDoS attack on Reddit right now?
i broke reddit? wat? I think we crashed Reddit #wow

Confirmation Stage
wow turns out reddit is being DDoS attacked right now
Reddit is experiencing a malicious DDoS attack
Reddit's reward for the Boston bombing? DDOS attacks.

Resumption Stage
@JpDeathBlade Reddit is back and in full force.
Reddit may be returning.
It's ok, Reddit is back up. Go home, nothing to see here.

Figure 3: Examples from the four stages of a social media response to DDoS attacks.

chose a sample of false positives and false negatives, and manually looked at these incorrect decisions to align common mistakes with how they related to the 4 stages. Looking at the false positives, the majority are from the Symptom and Inference Stages. Looking at false negatives, we found attacks where the network did **not** make a public statement, so the Confirmation Stage was missing.

These stages of course do not account for all of the mistakes that are made. Precision is at 61% in our best model, leaving room for improvement. Other reasons for errors included distractor events. For example, the Boston Bombing occurred near the Reddit DDoS. The preceding days included thousands of tweets talking about the *attack* in Boston. This is obviously a different type of attack, and the machine learners were led astray.

5.6 Robustness

A danger in many stochastic processes is finding one good run and only reporting on those results. We thus compare our our model across runs and found the topics to be somewhat robust and steady. We chose five random runs of the best performing model (the one from Figure 2) and focused on the largest attack topic. Is this topic learned in all runs? Not only was the same topic subjectively learned in each run, we graphed the observed frequency of this largest attack topic from 5 of the 10 runs. Not only did it maintain the same frequency, but also the same general shape across the runs. Space prohibits more illustration, but the graph can be found on our data website: `www.usna.edu/Users/cs/nchamber/data/ddos/`

6 Discussion

The core conclusion from our experiments is that social media does indeed contain signals to identify DDoS attacks. Our proposed neural network outperformed previous work (Motoyama et al., 2010) by 20% F1, a very large margin. Even though online users are an *indirect* source of evidence, the 53% F1 from the neural network shows that useful information can be extracted from text.

We further improved results with the generative PLDAttack model based on topic modeling, achieving a smaller 4% increase over the neural net but 25% over the prior trending approach. Although neural networks have significant advantages over LDA-based models, PLDAttack offers advantages by enabling deeper analysis of what people say, what topics are discussed, and how attack discussions evolve over time on Twitter. For instance, it enabled Figure 2 to illustrate the different topics that people discuss during such an event.

Can these results be used in a DDoS detection framework? We believe it can. PLDAttack recall may not be as high as desired, but it can be increased by adjusting the prediction cutoff probability λ. We empirically set the cutoff based on dev set performance to optimize F1. However, a detection system may desire to optimize recall at the expense of precision, thus choosing a lower λ and forcing the system to predict attacks more often. This would increase false positives, but with a human in the loop, it is manageable to monitor.

This paper thus proposed two NLP models for learning to identify DDoS attacks from social media without network data. They leverage *indirect* evidence described by users when they post online about service availability. By identifying the *early topics* before public announcements, we see this as an important step toward a broad-scale monitoring system not dependent on individual network reporting. We hope our datasets and models encourage further efforts in NLP and Computer Security. Models and data are available online: *www.usna.edu/Users/cs/nchamber/data/ddos/*

7 Acknowledgments

This work was supported in part by a grant from the Office of Naval Research. We also thank the support of the DoD HPC Modernization Office for enabling our undergraduate education and research. Finally, thanks to EdinburghNLP for hosting me while wrapping up this work. Slàinte!

References

Pramod Anantharam, Payam Barnaghi, Krishnaprasad Thirunarayan, and Amit Sheth. 2015. Extracting city traffic events from social streams. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(4):43.

Edward Benson, Aria Haghighi, and Regina Barzilay. 2011. Event discovery in social media feeds. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 389–398. Association for Computational Linguistics.

David M Blei. 2012. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

David A Broniatowski, Michael J Paul, and Mark Dredze. 2013. National and local influenza surveillance through twitter: an analysis of the 2012-2013 influenza epidemic. *PloS one*, 8(12):e83672.

Ching-Yun Chang, Zhiyang Teng, and Yue Zhang. 2016. Expectation-regulated neural model for event mention extraction. In *Proceedings of NAACL-HLT*, pages 400–410.

Weiwei Guo, Hao Li, Heng Ji, and Mona T Diab. 2013. Linking tweets to news: A framework to enrich short text data in social media. In *ACL (1)*, pages 239–249. Citeseer.

Dennis Kergl. 2015. Enhancing network security by software vulnerability detection using social media analysis extended abstract. In *Data Mining Workshop (ICDMW), 2015 IEEE International Conference on*, pages 1532–1533. IEEE.

Dennis Kergl, Robert Roedler, and Gabi Dreo Rodosek. 2016. Detection of zero day exploits using real-time social media streams. In *Advances in Nature and Biologically Inspired Computing*, pages 405–416. Springer.

Alex Lamb, Michael J Paul, and Mark Dredze. 2013. Separating fact from fear: Tracking flu infections on twitter. In *Proceedings of HLT-NAACL*, pages 789–795.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Marti Motoyama, Brendan Meeder, Kirill Levchenko, Geoffrey M Voelker, and Stefan Savage. 2010. Measuring online service availability using twitter. *WOSN*, 10:13–13.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Graham Neubig, Yuichiroh Matsubayashi, Masato Hagiwara, and Koji Murakami. 2011. Safety information mining-what can nlp do in a disaster-. In *IJCNLP*, volume 11, pages 965–973.

Ana-Maria Popescu, Marco Pennacchiotti, and Deepa Paranjpe. 2011. Extracting events and event descriptions from twitter. In *Proceedings of the 20th international conference companion on World wide web*, pages 105–106. ACM.

Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D Manning. 2009. Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 248–256. Association for Computational Linguistics.

Daniel Ramage, Christopher D. Manning, and Susan Dumais. 2011. Partially labeled topic models for interpretable text mining. In *Proceedings of KDD*.

Alan Ritter, Oren Etzioni, Sam Clark, et al. 2012. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1104–1112. ACM.

Alan Ritter, Evan Wright, William Casey, and Tom Mitchell. 2015. Weakly supervised extraction of computer security events from twitter. In *Proceedings of the 24th International World Wide Web Conference (WWW)*.

Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, pages 851–860. ACM.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Wei Wei. 2016. *Probabilistic Models of Topics and Social Events*. Ph.D. thesis, Arizona State University.

Deyu Zhou, Xuan Zhang, and Yulan He. 2017. Event extraction from twitter using non-parametric bayesian mixture model with word embeddings. In *European Association for Computational Linguistics (EACL)*.

The Importance of Calibration for Estimating Proportions from Annotations

Dallas Card
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA, 15213, USA
dcard@cmu.edu

Noah A. Smith
Paul G. Allen School of CSE
University of Washington
Seattle, WA, 98195, USA
nasmith@cs.washington.edu

Abstract

Estimating label proportions in a target corpus is a type of measurement that is useful for answering certain types of social-scientific questions. While past work has described a number of relevant approaches, nearly all are based on an assumption which we argue is invalid for many problems, particularly when dealing with human annotations. In this paper, we identify and differentiate between two relevant data generating scenarios (intrinsic vs. extrinsic labels), introduce a simple but novel method which emphasizes the importance of calibration, and then analyze and experimentally validate the appropriateness of various methods for each of the two scenarios.

1 Introduction

A methodological tool often used in the social sciences and humanities (and practical settings like journalism) is *content analysis* – the manual categorization of pieces of text into a set of categories which have been developed to answer a substantive research question (Krippendorff, 2012). *Automated* content analysis holds great promise for augmenting the efforts of human annotators (O'Connor et al., 2011; Grimmer and Stewart, 2013). While this task bears similarity to text categorization problems such as sentiment analysis, the quantity of real interest is often the *proportion* of documents in a dataset that should receive each label (Hopkins and King, 2010). This paper tackles the problem of estimating label proportions in a target corpus based on a small sample of human annotated data.

As an example, consider the hypothetical question (not explored in this work) of whether hate speech is increasingly prevalent in social media posts in recent years. "Hate speech" is a difficult-to-define category only revealed (at least initially) through human judgments (Davidson et al., 2017). Note that the goal would not be to identify individual instances, but rather to estimate a proportion, as a way of measuring the prevalence of a social phenomenon. Although we assume that trained annotators could recognize this phenomenon with some acceptable level of agreement, relying solely on manual annotation would restrict the number of messages that could be considered, and would limit the analysis to the messages available at the time of annotation.[1]

We thus treat proportion estimation as a *measurement* problem, and seek a way to train an instrument from a limited number of human annotations to measure label proportions in an unannotated target corpus.

This problem can be cast within a supervised learning framework, and past work has demonstrated that it is possible to improve upon a naïve classification-based approach, even without access to any labeled data from the target corpus (Forman, 2005, 2008; Bella et al., 2010; Hopkins and King, 2010; Esuli and Sebastiani, 2015). However, as we argue (§2), most of this work is based on a set of assumptions that we believe are invalid in a significant portion of text-based research projects in the social sciences and humanities.

Our contributions in this paper include:

- identifying two different data-generating scenarios for text data (*intrinsic* vs. *extrinsic* labels) and and establishing their importance to the problem of estimating proportions (§2);
- analyzing which methods are suitable for each setting, and proposing a simple alternative approach for extrinsic labels (§3); and
- an empirical comparison of methods that validates our analysis (§4).

[1] For additional examples see Grimmer et al. (2012), Hopkins and King (2010), and references therein.

Proceedings of NAACL-HLT 2018, pages 1636–1646
New Orleans, Louisiana, June 1 - 6, 2018.

Complicating matters somewhat is the fact that annotation may take place before the entire collection is available, so that the subset of instances that are manually annotated may represent a biased sample (§2). Because this is so frequently the case, all of the results in this paper assume that we must confront the challenges of *transfer learning* or *domain adaptation*. (The simpler case, where we can sample from the true population of interest, is revisited in §5.)

2 Problem Definition

Our setup is similar to that faced in transfer learning, and we use similar terminology (Pan and Yang, 2010; Weiss et al., 2016). We assume that we have a source and a target corpus, comprised of N_S and N_T documents respectively, the latter of which are not available for annotation. We will represent each corpus as a set of documents, i.e., $\boldsymbol{X}^{(S)} = \langle \boldsymbol{x}_1^{(S)}, ..., \boldsymbol{x}_{N_s}^{(S)} \rangle$, and similarly for $\boldsymbol{X}^{(T)}$.

We further assume that we have a set of K mutually exclusive categories, $\mathcal{Y} = \{1, \ldots, K\}$, and that we wish to estimate the proportion of documents in the target corpus that belong to each category. These would typically correspond to a quantity we wish to measure, such as what fraction of news articles frame a policy issue in a particular way, what fraction of product reviews are considered helpful, or what fraction of social media messages convey positive sentiment. Generally speaking, these categories will be designed based on theoretical assumptions, an understanding of the design of the platform that produced the data, and/or initial exploration of the data itself.

In idealized text classification scenarios, it is conventional to assume training data with already-assigned gold-standard labels. Here, we are interested in scenarios where we must generate our labels via an annotation process.[2] Specifically, assume that we have some annotation function, $\mathcal{A}$, which produces a distribution over the K mutually exclusive labels, conditional on text. Given a document, $\boldsymbol{x}_i$, the annotation process samples a label from the annotation function, defined as:

$$\mathcal{A}(\boldsymbol{x}_i, k) \triangleq p(y_i = k \mid \boldsymbol{x}_i). \tag{1}$$

Typically, the annotation function would represent the behavior of a human annotator (or group of annotators), but it could also represent a less controlled real-world process, such as users rating a review's helpfulness. Note that our setup *does* include the special case in which true gold-standard labels are available for each instance (such as the authors of documents in an authorship attribution problem). In such a case, $\mathcal{A}$ is deterministic (assuming unique inputs).

Given that our objective is to mimic the annotation process, we seek to estimate the proportion of documents in the target corpus expected to be categorized into each of the K categories, if we had an unlimited budget and full access to the target corpus at the time of annotation. That is, we wish to estimate $q^{(T)}$, which we define as:

$$q(y = k \mid \boldsymbol{X}^{(T)}) \triangleq \tfrac{1}{N_T} \textstyle\sum_{i=1}^{N_T} p(y_i = k \mid \boldsymbol{x}_i^{(T)}). \tag{2}$$

Given a set of documents sampled from the *source* corpus and L applications of the annotation function, we can obtain, at some cost, a labeled training corpus of L documents, i.e., $D^{(\text{train})} = \langle (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_L, y_L) \rangle$. Because the source and target corpora are not in general drawn from the same distribution, we seek to make explicit our assumptions about how they differ.[3] Past literature on transfer learning has identified several patterns of dataset shift (Storkey, 2009). Here we focus on two particularly important cases, linking them to the relevant data generating processes, and analyze their relevance to estimating proportions.

Two kinds of distributional shift. There are two natural assumptions we could make about what is constant between the two corpora. We could assume that there is no change in the distribution of text given a document's label, that is $p^{(S)}(\boldsymbol{x} \mid y) = p^{(T)}(\boldsymbol{x} \mid y)$. Alternately, we could assume that there is no change in the distribution of labels given text, i.e., $p^{(S)}(y \mid \boldsymbol{x}) = p^{(T)}(y \mid \boldsymbol{x})$. The former is assumed in the case of *prior probability shift*, where we assume that $p(y)$ differs but $p(\boldsymbol{x} \mid y)$ is constant, and the later is assumed in the case of *covariate shift*, where we assume that $p(\boldsymbol{x})$ differs but $p(y \mid \boldsymbol{x})$ is constant (Storkey, 2009).

These two assumptions correspond to two fundamentally different types of scenarios that we need to consider, which are summarized in Table 1. The first is where we are dealing with what we

[2] This could include gathering multiple independent annotations per instance, but we will typically assume only one.

[3] Clearly, if we make no assumptions about how the source and target distributions are related, there is no guarantee that supervised learning will work (Ben-David et al., 2012).

Label type	Intrinsic	Extrinsic
Data generating process	$\boldsymbol{x} \sim p(\boldsymbol{x} \mid y)$	$y \sim p(y \mid \boldsymbol{x})$
Assumed to differ across domains	$p(y)$	$p(\boldsymbol{x})$
Assumed constant across domains	$p(\boldsymbol{x} \mid y)$	$p(y \mid \boldsymbol{x})$
Corresponding distributional shift	Prior probability shift	Covariate shift

Table 1: Data generating scenarios and corresponding distributional properties.

will call *intrinsic* labels, that is labels which are inherent to each instance, and which in some sense precede and predict the generation of the text of that instance. A classic example of this scenario is the case of authorship attribution (e.g., Mosteller and Wallace, 1964), in which different authors are assumed to have different propensities to use different styles and vocabularies. The identity of the author of a document is arguably an intrinsic property of that document, and it is easy to see a text as having been generated conditional on its author.

The contrasting scenario is what we will refer to as *extrinsic* labels; this scenario is our primary interest. We assume here that the labels are not inherent in the documents, but rather have been externally generated, conditional on the text as a stimulus to some behavioral process.[4] We argue that this is the relevant assumption for most annotation-based projects in the social sciences, where the categories of interest do not correspond to pre-existing categories that might have existed in the minds of authors before writing, or affected the writing process. Rather, these are theorized categories that have been developed specifically to analyze or measure some aspect of the document's *effect* that is of interest to the researcher.

We won't always know the true distributional properties of our datasets, but distinguishing between intrinsic and extrinsic labels provides a guide. The critical point is that these two different labeling scenarios have different implications for robustness to distributional shift. In the case of extrinsic labels, especially when working with trained annotators, it is reasonable to assume that the behavior of the annotation function is determined purely by the text, such that $p(y \mid \boldsymbol{x})$ is unchanged between source and target, and any change in label proportions is explained by a change in the underlying distribution of text, $p(\boldsymbol{x})$. With intrinsic labels, by contrast, it *may* be the case that $p(\boldsymbol{x} \mid y)$ is the same for the source and the target, assuming there are no additional factors influencing the generation of text. In that case, a shift in the distribution of features would be fully explained by a difference in the underlying label proportions.

The idea that there are different data generating processes is obviously not new.[5] What is novel here, however, is asking how these different assumptions affect the estimation of proportions. Virtually all past work on estimating proportions has only considered prior probability shift, assuming that $p(\boldsymbol{x} \mid y)$ is constant.[6] Existing methods take advantage of this assumption, and can be shown empirically to work well when it is satisfied (e.g., through artificial modification of real datasets to alter label proportions in a corpus). We expect them to fail, however, in the case of extrinsic annotations, as there is no reason to think that the required assumption should necessarily hold.

By contrast, the problem of covariate shift is in some sense less of a problem because we directly observe $\boldsymbol{X}^{(T)}$. Since the annotation function is assumed to be unchanging, we could perfectly predict the expected label proportions in the target corpus if we could learn the annotation function using labeled data from the source corpus. The problem thus becomes how to learn a well-calibrated approximation of the annotation function from a limited amount of labeled data.

3 Methods

Given a labeled training set and a target corpus, the naïve approach is to train a classifier through any conventional means, predict labels on the target corpus, and return the relative prevalence of predicted labels. Following Forman (2005), we refer to this approach as **classify and count** (CC). If using a probabilistic classifier, averaging the predicted posterior probabilities rather than predicted labels will be referred to as **probabilistic classify and count** (PCC; Bella et al., 2010).

Both approaches can fail, however. In the case of intrinsic labels, this is because these approaches will not account for the shift in prior label prob-

[4] Fong and Grimmer (2016) also consider this process in attempting to identify the causal effects of texts.

[5] Peters et al. (2014) describe these, somewhat confusingly, as *causal* and *anti-causal* problems.

[6] For example, Hopkins and King (2010) argue that bloggers first decide on the sentiment they wish to convey and then write a blog post conditional on that sentiment.

ability, $p(y)$, which is assumed to have occurred (Hopkins and King, 2010). In the case of covariate shift, the difference in $p(\boldsymbol{x})$ will result in a model that is not optimal (in terms of classification performance) for the target domain. In both cases, there is also the problem of classifier bias or miscalibration. Particularly in the case of unbalanced labels, a standard classifier is likely to be biased, overestimating the probability of the more common labels, and vice versa (Zhao et al., 2017). Here we present a simple but novel method for extrinsic labels, followed by a number of baseline approaches against which we will compare. (See supplementary material for additional details.)

3.1 Proposed method: calibrated probabilistic classify and count (PCC$^{\text{cal}}$)

One simple solution, which we propose here, is to attempt to train a well-calibrated classifier. To be clear, calibration refers to the long-run accuracy of predicted probabilities. That is, a probabilistic classifier, $h_\theta(\boldsymbol{x})$, is well calibrated at the level μ if, among all instances for which the classifier predicts class k with a probability of μ, the proportion that are truly assigned to class k is also equal to μ.[7]

It has previously been shown (DeGroot and Fienberg, 1983; Bröcker, 2009) that any proper scoring rule (e.g., cross entropy, Brier score, etc.) can be factored into two components representing *calibration* and *refinement*, the later of which effectively measures how close predicted probabilities are to zero or one. Minimizing a corresponding loss function thus involves a trade-off between these two components.

Optimizing *only* for calibration is not helpful, as a trivial solution is to simply predict a probability distribution equal to the observed label proportions in the training data for all instances (which is perfectly calibrated on the labeled sample). The alternative we propose here is to train a classifier using a typical objective (here, regularized log loss) but use *calibration on held-out data* as a criterion for *model selection*, i.e., when we tune hyperparameters via cross validation. We refer to this method as **calibrated PCC** (PCC$^{\text{cal}}$). Specifically, we select regularization strength via grid search, choosing the value that leads to the lowest average calibration error across training / held-out splits. Of course, other hyperparameters could be included in model selection as well.

To estimate calibration error (CE) during cross-validation, we use an approximation due to Nguyen and O'Connor (2015), adaptive binning. In the case of binary labels, this is computed as:

$$\text{CE} \triangleq \frac{1}{B} \sum_{j=1}^{B} \left(\frac{1}{|\mathcal{B}_j|} \sum_{i \in \mathcal{B}_j} y_i - p_\theta(\boldsymbol{x}_i) \right)^2, \quad (3)$$

using B bins, where bin $\mathcal{B}_j$ contains instances for which $p_\theta(\boldsymbol{x}_i)$ are in the j^{th} quantile, where $p_\theta(\boldsymbol{x}_i)$ is the predicted probability of a positive label for instance i. For added robustness, we take the average of CE for $B \in \{3, 4, 5, 6, 7\}$.

In our experiments, we consider two variants of PCC: the first, PCC$^{\text{F}_1}$, which is a baseline, is tuned conventionally for classification performance, whereas the other (PCC$^{\text{cal}}$) is tuned for calibration, as measured using CE, but is otherwise identically trained. As a base classifier we make use of l_1-regularized logistic regression, operating on n-gram features.[8]

3.2 Existing methods appropriate for extrinsic labels

The idea of extrinsic labels has not been previously considered by past work on estimating proportions, but it is closely related to the problems of calibration and covariate shift. Here we briefly summarize two representative methods, which we consider as baselines (see supplementary material for details).

Platt scaling. One approach to calibration is to train a model using conventional methods and to then learn a secondary calibration model. One of the most common and successful variations on this approach is Platt scaling, which learns a logistic regression classifier on held-out training data, taking the scores from the primary classifier as input. This model is then applied to the scores returned by the primary classifier on the target corpus (Platt, 1999). To estimate proportions, the predicted probabilities are then averaged, as in PCC.

Reweighting for covariate shift. Although they are not typically thought of in the context of estimating proportions, several methods have been proposed to deal directly with the problem of covariate shift, including kernel mean matching and

[7] For example, a weather forecaster will be well-calibrated if it rains on 60% of days for which the forecaster predicted a 60% chance of rain, etc.

[8] More complex models could be considered, but we use logistic regression because it is a well-understood and widely applicable model that has been shown to be relatively well-calibrated in general (Niculescu-Mizil and Caruana, 2005).

its extensions (Huang et al., 2006; Sugiyama et al., 2011). Here, we consider the two-stage method from Bickel et al. (2009), which uses a logistic regression model to distinguish between source and target domains, and then uses the probabilities from this model to re-weight labeled training instances, to more heavily favor those that are representative of the target domain. The appeal of this method is that all unlabeled data can be used to estimate this shift.

3.3 Existing methods appropriate for intrinsic labels

As previously mentioned, virtually all of the past work on estimating proportions makes the assumption that $p(\boldsymbol{x} \mid y)$ is constant between source and target. Under this assumption, it can be shown that $p(y^{(\theta)} = j \mid y = k)$ is also constant for all j and k, where $y^{(\theta)}$ is the predicted label from h_θ, and y is the true (intrinsic) label. If these values were known, then the label proportions in the target corpus could be found by taking the model's estimate of label proportions in the target corpus, (CC), and then solving a linear system of equations as a post-classification correction. Although a number of variations on this model have been proposed, all are based on the same assumption, thus we take a method known as **adjusted classify and count** (ACC) as an exemplar, which directly estimates the relevant quantities using a confusion matrix (Forman, 2005). In the case of binary classification, this reduces to:

$$\hat{q}_{\text{ACC}}(y = 1 \mid \boldsymbol{X}^{(T)}) = \frac{\frac{1}{N_T}\sum_{i=1}^{N_T} y_i^{(\theta)} - \text{FPR}}{\text{TPR} - \text{FPR}}, \quad (4)$$

where $\text{FPR} = \hat{p}(y^{(\theta)} = 1 \mid y = 0)$ and $\text{TPR} = \hat{p}(y^{(\theta)} = 1 \mid y = 1)$ are both estimated using held-out data.

4 Experiments

For our experiments, we focus on the case of binary classification where the difference between the source and target corpora results from a difference in time—that is, the training documents are sampled from one time period, and the goal is to estimate label proportions on documents from a future time period. We include examples of both intrinsic and extrinsic labels to demonstrate the importance of this distinction to the effectiveness of different methods.

As described below, we create multiple subtasks from each dataset by using different partitions of the data. In all cases, we report absolute error (AE) on the proportion of positive instances, averaged across the subtasks of each dataset.

Although we do not have access to the true annotation function, we approximate the expected label proportions in the target corpus by averaging the available labels, which should be a very close approximation when the number of available labels is large (which informed our choice of datasets for these experiments). For a single subtask, the absolute error is thus evaluated as

$$\text{AE} = \left| \hat{q}(y = 1 \mid \boldsymbol{X}^{(T)}) - \tfrac{1}{N_T}\textstyle\sum_{i=1}^{N_T} y_i^{(T)} \right|. \quad (5)$$

For all experiments, we also report the AE we would obtain from using the observed label proportions in the training sample as a prediction (labeled "Train"). Although this does not correspond to an interesting prediction (as it only says the future will always look exactly like the past), it does represent a fundamental baseline. If a method is unable to do better than this, it suggests that the method has too much measurement error to be useful.

To test for statistically significant differences between methods, we use an omnibus application of the Wilcoxon signed-rank test to compare one method against all others, including a Bonferroni correction for the total number of tests per hypothesis. With 4 datasets, each with 2 sample sizes, comparing against 6 other methods this results in a significance threshold of approximately 0.001.

Finally, in order to connect this work with past literature on estimating proportions, we also include a side experiment with one intrinsically-labeled dataset where we have artificially modified the label proportions in the target corpus by dropping positive or negatively-labeled instances in order to simulate a large prior probability shift between the source and target domains.

4.1 Datasets

We briefly describe the datasets we have used here and provide additional details in the supplementary material. Note that although this work is primarily focused on applications in which the amount of human-annotated data is likely to be small, fair evaluation of these methods requires datasets that are large enough that we can approximate the expected label proportion in the target

corpus using the available labels; as such, the following datasets were chosen so as to have a representative sample of sufficiently large intrinsically and extrinsically-labeled data, where documents were time-stamped, with label proportions that differ between time periods.

Media Frames Corpus (MFC): As a primary example of extrinsic labels, we use a dataset of several thousand news articles that have been annotated in terms of a set of broad-coverage framing dimensions (such as economics, morality, etc.). We treat annotations as indicating the presence or absence of each dimension, and consider each one as a separate sub-task. As with all datasets, we create a source and target corpus by dividing the datasets by year. Particularly for this dataset, it seems reasonable to posit that the annotation function was relatively constant between source and target, as the annotators worked without explicit knowledge of the article's date (Card et al., 2015).

Amazon reviews: As a secondary example of extrinsic labels, we make use of a subset of Amazon reviews for five different product categories, each of which has tens of thousands of reviews. For this dataset, we ignore the star rating associated with the review, and instead focus on predicting the proportion of people that would rate the review as helpful. Here we create separate subtasks for each product category by considering each pair of adjacent years as a source and target corpus, respectively (McAuley et al., 2015).

Yelp reviews: As a primary example of a large dataset with intrinsic labels, we make use of the Yelp10 dataset, treating the source location of the review as the label of interest. Specifically, we create binary classification tasks by choosing pairs of cities with approximately the same number of reviews, and again use year of publication to divide the data into source and target corpora, creating multiple subtasks per pair of cities.

Twitter sentiment: Finally, we include a Twitter sentiment analysis dataset which was collected and automatically labeled, using the presence of certain emoticons as implicit labels indicating positive or negative sentiment (with the emoticons then removed from the text). Because of the way this data was collected, and the relatively narrow time coverage, it seems plausible to treat the sentiment as an intrinsic label. As with the above datasets, we create subtasks by considering all pairs of temporally adjacent days with sufficient tweets, and treating them as a paired source and target corpora, respectively. (Go et al., 2009).

4.2 Results

The results on the datasets with extrinsic and intrinsic labels are presented in Figures 1 and 2, respectively.

As expected, the results differ in important ways between intrinsically and extrinsically labeled datasets, although there are some results which hold in all cases. In all settings, CC is worse on average than predicting the observed proportions in the training data (significantly worse for the Amazon and Twitter datasets), reinforcing the idea that averaging the predictions from a classifier will lead to a biased estimate of label proportions. This same finding holds for PCC^{F_1} when the amount of labeled data is small ($L = 500$), suggesting that simply averaging the predicted probabilities is not reliable without a sufficiently large labeled dataset.

For the datasets with *extrinsic* labels, PCC^{cal} performs best on average in all settings. For the MFC dataset, PCC^{cal} is significantly better than all methods except Platt scaling when $L = 500$ and significantly better than all methods except reweighting and PCC^{F_1} when $L = 2000$ (after a Bonferroni correction, as in all cases). As expected, ACC is actually worse on average than CC on the extrinsic datasets, presumably because of the mismatched assumptions. Reweighting for covariate shift offers mediocre performance in all settings, perhaps because, while it attempts to account for covariate shift, it may still suffer from miscalibration.

On the datasets with *intrinsic* labels, by contrast, no one method dominates the others. As expected, ACC does poorly when the amount of labeled data is small ($L = 500$); it does improve upon CC when $L = 4000$, but not by enough to do significantly better than other methods, perhaps calling into question the validity of the assumption that $p(\boldsymbol{x} \mid y)$ is constant in these datasets.

Surprisingly, both Platt scaling and PCC^{cal} also offer competitive performance in the experiments with intrinsic labels. However, this is likely the case in part because the change in label proportions is relatively small from year to year (or day

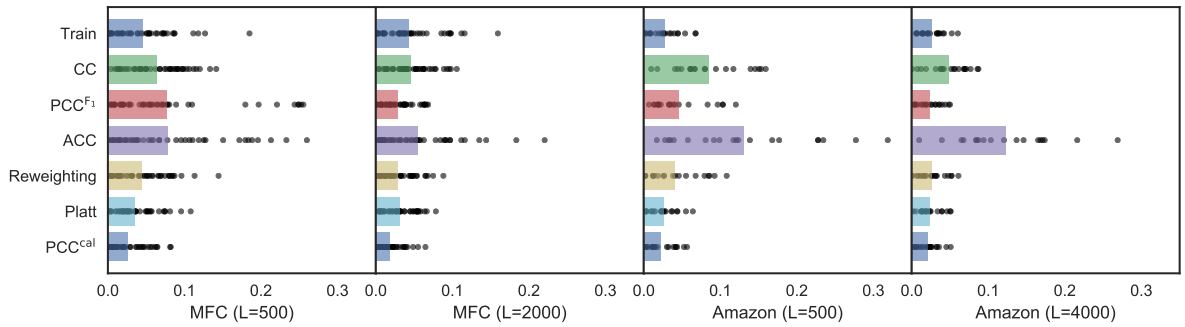


Figure 1: Absolute error (AE) on datasets with extrinsic labels. Each dot represents the result for a single subtask, and bars show the mean. PCC^{cal} (bottom row) performs best on average in all cases and is significantly better than most other methods on MFC.

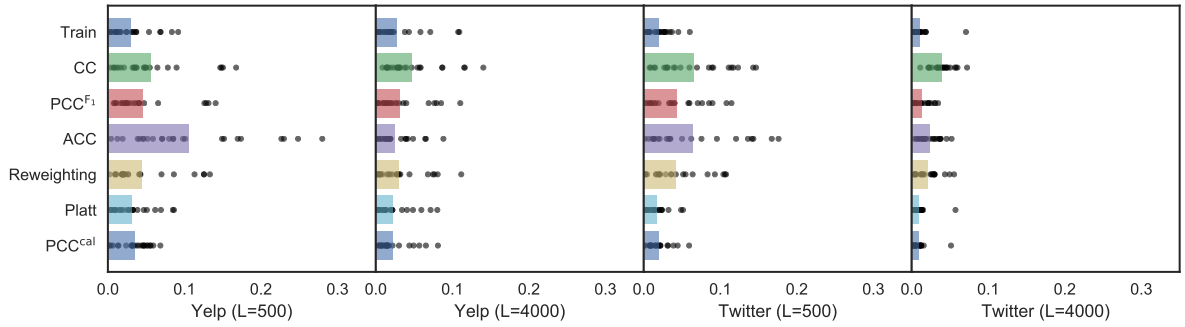


Figure 2: Absolute error (AE) on datasets with intrinsic labels. No method is significantly better than all others.

to day in the case of Twitter). This is illustrated by Figure 3, which presents the results of the side-experiment with artificially modified (intrinsic) label proportions using a subset of the Twitter data. These results confirm past findings, and show that ACC drastically outperforms other methods such as PCC^{F_1}, *if* we selectively drop instances so as to enforce a large difference in label proportions between source and target. This is the expected result, as ACC is the only method tailored to deal with prior probability shift (which is being artificially simulated). Unfortunately, its advantage is not maintained when the difference between source and target is small, which is the case for all of the naturally-occurring differences we found in the Yelp and Twitter datasets. Although past work has relied heavily on these sorts of simulated differences and artificial experiments, it is unclear whether they are a good substitute for real-world data, given that we mostly observed relatively small differences in practice.

Finally, we also tested the effect of using l_2 instead of l_1 regularization, but found that it tended to produce significantly worse estimates of proportions using CC and PCC^{F_1} on the datasets with

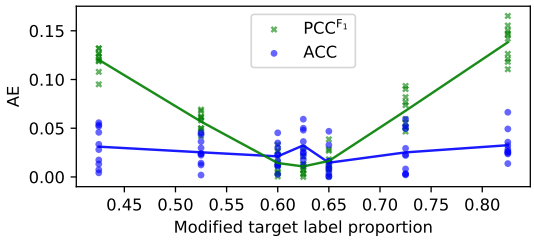


Figure 3: Absolute error (AE) for predictions on one day of Twitter data ($L = 5000$) when artificially modifying target proportions. The proportion of positive labels in the source corpus is 0.625. ACC performs significantly better given an large artificially-created difference in label proportions between source and target, but not when the difference is small.

extrinsic labels, and statistically indistinguishable results using other methods, suggesting that either type of regularization could serve as a basis for PCC^{cal} or Platt scaling.

5 Discussion

As anyone who has worked with human annotations can attest, the process of collecting annotations is messy and time-consuming, and tends to

involve large numbers of disagreements (Artstein and Poesio, 2008). Although it is conventional to treat disagreements as *errors* on the behalf of some subset of annotators, this paper provides an alternative way of understanding these. By treating annotation as a stochastic process, conditional on text, we can explain not only the disagreements between annotators, but also the lack of self-consistency that is also sometimes observed. Although the assumption that $p(y \mid \boldsymbol{x})$ does not change is clearly a simplification, it seems reasonable when working with trained annotators. Certainly this assumption seems much better justified than the conventional assumption that $p(\boldsymbol{x} \mid y)$ is constant, since the latter does not account for differences in the distribution of text arising from differences in subject matter, etc.

Although we have demonstrated that using a method that is appropriate to the data generating process is beneficial, it is important to note that all methods presented here can still result in relatively large errors in the worst cases. In part this is due to the difficulty of learning a conditional distribution involving high-dimensional data (such as text) with only a limited number of annotations. Even with much more annotated data, however, previously unseen features could still have a potentially large impact on future annotations. Ultimately, we should be cautious about all such predictions, and always validate where possible, by eventually sampling and annotating data from the target corpus.

What if we can sample from the target corpus? Although there are many situations in which domain adaptation is unavoidable (such as predicting public opinion from Twitter in real time with models trained on the past), at least some research projects in the humanities and social sciences might reasonably have access to all data of interest from the beginning of the project, such as when working with a historical corpus. Although a full proof is beyond the scope of this paper, in this case, the best approach is almost certainly to simply sample a random set of documents, label them using the annotation function, and report the relative prevalence of each label (Hopkins and King, 2010).

Although this *simple random sampling* (SRS) approach ignores the text, it is an unbiased estimator with variance that can easily calculated, at least

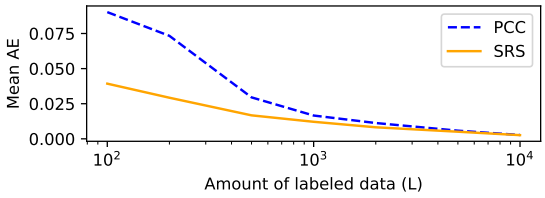


Figure 4: Comparison of SRS and PCC in simulation when we know the true model and sample from the target corpus (averaged over 200 repetitions).

in approximation.[9] More importantly, because it is independent of the dimensionality of the data, it works well on high-dimensional data, such as text, whereas classification-based approaches will struggle. We can illustrate this by comparing SRS and PCC in simulation. Figure 4 shows the mean AE (averaged over 200 trials) for a case in which we know the true model (including the prior on the weights, and thus the appropriate amount of regularization) and only need to learn the values of the weights. Even in this idealized scenario, SRS remains better than PCC for all values of L. (See supplementary material for details).

Depending on the level of accuracy required, simply sampling a few hundred documents and labeling them should be sufficient to get a reasonably reliable estimate of the overall label proportions, along with an approximate confidence interval. Unfortunately, this option is only available when we have full access to the target corpus at the time of annotation.

Additional related work. There is a small literature on the problem of estimating proportions in a target dataset (see §1); as we have emphasized, almost all of it makes the assumption that $p(\boldsymbol{x} \mid y)$ is the same for both source and target. Moreover, most of the methods that have been proposed have been tested using relatively small datasets, or datasets where the target corpus has been artificially modified by altering the label proportions in the target corpus (as we did in the side experiment reported in Figure 3). It

[9] If we were sampling with replacement, the variance in the binary case would be given by the standard formula $\mathbb{V}[\hat{q}^{\text{SRS}}] = \frac{\bar{p}(1-\bar{p})}{L}$, where $\bar{p} = \frac{1}{N_T}\sum_{i=1}^{N_T} p(y_i = 1 \mid \boldsymbol{x}_i)$. This may not be possible, however, as annotators seeing a document for the second or third time would likely be affected by their own past decisions. Nevertheless, using this as the basis for a plug-in estimator should still be a reasonable approximation when the target corpus is large. Please refer to supplementary material for additional details.

seems unclear that this is a good simulation of the kind of shift in distribution that one is likely to encounter in practice. An exception to this is Esuli and Sebastiani (2015), who test their method on the RCV1-v2 corpus, also splitting by time. They perform a large number of experiments, but unfortunately, nearly all of their experiments involve only a very small difference in label proportions between the source and target (with the vast majority < 0.01), which limits the generalizability of their findings. Additional methods for calibration could also be considered, such as the isotonic regression approach of Zadrozny and Elkan (2002), but in practice we would expect the results to be very similar to Platt scaling.

Another line of work has approached the problem of aggregating labels from multiple annotators (Raykar et al., 2009; Hovy et al., 2013; Yan et al., 2013). That is, if we believe that some annotators are more reliable than others, it might make sense to try to determine this in an unsupervised manner, and give more weight to the annotations from the reliable annotators. This seems particularly appropriate when dealing with uncooperative annotators, as might be encountered, for example, in crowdsourcing (Snow et al., 2008; Zhang et al., 2016). However, with a team of trained annotators, we believe that honest disagreements could contain valuable information better not ignored.

Finally, this work also relates to the problem of *active learning*, where the goal is to interactively choose instances to be labeled, in a way that maximizes accuracy while minimizing the total cost of annotation (Beygelzimer et al., 2009; Baldridge and Osborne, 2004; Rai et al., 2010; Settles, 2012). This is an interesting area that might be productively combined with the ideas in this paper. In general, however, the use of active learning involves additional logistical complications and does not always work better than random sampling in practice (Attenberg and Provost, 2011).

6 Conclusions

When estimating proportions in a target corpus, it is important to take seriously the data generating process. We have argued that in the case of data annotated by humans in terms of categories designed to help answer social-scientific research questions, labels should be treated as *extrinsic*, generated probabilistically conditional on text, rather than as a combination of correct and incorrect judgements about a label *intrinsic* to the document. Moreover, it is reasonable to assume in this case that $p(y \mid \boldsymbol{x})$ is unchanging between source and target, and methods that aim to learn a well-calibrated classifier, such as PCC^{cal}, are likely to perform best. By contrast, if $p(\boldsymbol{x} \mid y)$ is unchanging between source and target, then various correction methods from the literature on estimating proportions, such as ACC, can perform well, especially when differences are large. Ultimately, any of these methods can still result in large errors in the worst cases. As such, validation remains important when treating the estimation of proportions as a type of measurement.

Acknowledgements We would like to thank Philip Resnik, Brendan O'Connor, anonymous reviewers, and all members of Noah's ARK for helpful comments and discussion, as well as XSEDE and Microsoft Azure for grants of computational resources used for this work.

References

Ron Artstein and Massimo Poesio. 2008. Inter-Coder Agreement for Computational Linguistics. *Computational Linguistics* 34(4):555–596. https://doi.org/10.1162/coli.07-034-R2.

Josh Attenberg and Foster Provost. 2011. Inactive learning?: Difficulties employing active learning in practice. *SIGKDD Explorations Newsletter* 12(2):36–41. https://doi.org/10.1145/1964897.1964906.

Jason Baldridge and Miles Osborne. 2004. Active learning and the total cost of annotation. In *Proceedings of EMNLP*.

Antonio Bella, Maria Jose Ramirez-Quintana, Jose Hernandez-Orallo, and Cesar Ferri. 2010. Quantification via probability estimators. In *IEEE International Conference on Data Mining*. https://doi.org/10.1109/ICDM.2010.75.

Shai Ben-David, Shai Shalev-Shwartz, and Ruth Urner. 2012. Domain adaptation – can quantity compensate for quality? *Annals of Mathematics and Artificial Intelligence* 70:185–202. https://doi.org/10.1007/s10472-013-9371-9.

Alina Beygelzimer, Sanjoy Dasgupta, and John Langford. 2009. Importance weighted active learning. In *Proceedings of ICML*. https://doi.org/10.1145/1553374.1553381.

Steffen Bickel, Michael Brückner, and Tobias Scheffer. 2009. Discriminative Learning Under Covariate Shift. *Journal of Machine Learning Research* 10.

Jochen Bröcker. 2009. Reliability, sufficiency, and the decomposition of proper scores. *Quarterly Journal of the Royal Meteorological Society* 135(643):1512–1519.

Dallas Card, Amber E. Boydstun, Justin H. Gross, Philip Resnik, and Noah A. Smith. 2015. The media frames corpus: Annotations of frames across issues. In *Proceedings of ACL*. https://doi.org/10.3115/v1/P15-2072.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of ICWSM*.

Morris H. DeGroot and Stephen E. Fienberg. 1983. The comparison and evaluation of forecasters. *The Statistician: Journal of the Institute of Statisticians* 32:12–22.

Andrea Esuli and Fabrizio Sebastiani. 2015. Optimizing text quantifiers for multivariate loss functions. *ACM Trans. Knowl. Discov. Data* 9(4). https://doi.org/10.1145/2700406.

Christian Fong and Justin Grimmer. 2016. Discovery of treatments from text corpora. In *Proceedings of ACL*. https://doi.org/10.18653/v1/P16-1151.

George Forman. 2005. Counting positives accurately despite inaccurate classification. In *Proceedings of the European Conference on Machine Learning*.

George Forman. 2008. Quantifying counts and costs via classification. *Data Mining and Knowledge Discovery* 17(2):164–206. https://doi.org/10.1007/s10618-008-0097-y.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. Technical report.

Justin Grimmer, Solomon Messing, and Sean J. Westwood. 2012. How words and money cultivate a personal vote: The effect of legislator credit claiming on constituent credit allocation. *American Political Science Review* 106(4):703–719. https://doi.org/10.1017/S0003055412000457.

Justin Grimmer and Brandon M. Stewart. 2013. Text as data: The promise and pitfalls of automatic content analysis methods for political texts. *Political Analysis* 21(3):267–297. https://doi.org/10.1093/pan/mps028.

Daniel Hopkins and Gary King. 2010. A method of automated nonparametric content analysis for social science. *American Journal of Political Science* 54(1):220–247. https://doi.org/10.1111/j.1540-5907.2009.00428.x.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard H. Hovy. 2013. Learning whom to trust with MACE. In *Proceedings of NAACL*.

Jiayuan Huang, Alexander J. Smola, Arthur Gretton, Karsten M. Borgwardt, and Bernhard Schölkopf. 2006. Correcting sample selection bias by unlabeled data. In *Proceedings of NIPS*.

Klaus Krippendorff. 2012. *Content analysis: an introduction to its methodology*. SAGE.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of SIGIR*. https://doi.org/10.1145/2766462.2767755.

Frederick Mosteller and David L. Wallace. 1964. *Inference and Disputed Authorship*. Addison-Wesley publishing company, Inc. https://doi.org/10.1080/01621459.1963.10500849.

Khanh Nguyen and Brendan O'Connor. 2015. Posterior calibration and exploratory analysis for natural language processing models. In *Proceedings of EMNLP*.

Alexandru Niculescu-Mizil and Rich Caruana. 2005. Predicting good probabilities with supervised learning. In *Proceedings of ICML*. https://doi.org/10.1145/1102351.1102430.

Brendan O'Connor, David Bamman, and Noah A. Smith. 2011. Computational text analysis for social science: Model assumptions and complexity. In *NIPS Workshop on Comptuational Social Science and the Wisdom of Crowds*.

S.J. Pan and Q. Yang. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering* 22(10):1345–1359. https://doi.org/10.1109/TKDE.2009.191.

Jonas Peters, Joris M. Mooij, Dominik Janzing, and Bernhard Schölkopf. 2014. Causal discovery with continuous additive noise models. *Journal of Machine Learning Research* 15:2009–2053.

John C. Platt. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74.

Piyush Rai, Avishek Saha, III Hal Daumé, and Suresh Venkatasubramanian. 2010. Domain adaptation meets active learning. In *Proceedings of NAACL*.

Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Anna K. Jerebko, Charles Florin, Gerardo Hermosillo, Luca Bogoni, and Linda Moy. 2009. Supervised learning from multiple experts: whom to trust when everyone lies a bit. In *Proceedings of ICML*. https://doi.org/10.1145/1553374.1553488.

Burr Settles. 2012. *Active Learning*. Morgan & Claypool. https://doi.org/10.2200/S00429ED1V01Y201207AIM018.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast—but is it good?: Evaluating non-expert annotations for natural language tasks. In *Proceedings of EMNLP*.

Amos J. Storkey. 2009. When training and test sets are different: Characterising learning transfer. In Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D. Candela Sugiyama Schwaighofer Lawrence Lawrence, editors, *Dataset Shift in Machine Learning*, MIT Press, chapter 1, pages 3–28.

Masashi Sugiyama, Makoto Yamada, Paul von Bünau, Taiji Suzuki, Takafumi Kanamori, and Motoaki Kawanabe. 2011. Direct density-ratio estimation with dimensionality reduction via least-squares hetero-distributional subspace search. *Neural Networks* 24(2). https://doi.org/10.1016/j.neunet.2010.10.005.

Karl Weiss, Taghi M. Khoshgoftaar, and DingDing Wang. 2016. A survey of transfer learning. *Journal of Big Data* 3(1). https://doi.org/10.1186/s40537-016-0043-6.

Yan Yan, Rómer Rosales, Glenn Fung, Subramanian Ramanathan, and Jennifer G. Dy. 2013. Learning from multiple annotators with varying expertise. *Machine Learning* 95:291–327. https://doi.org/10.1007/s10994-013-5412-1.

Bianca Zadrozny and Charles Elkan. 2002. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of KDD*. https://doi.org/10.1145/775047.775151.

Jing Zhang, Xindong Wu, and Victor S. Sheng. 2016. Learning from crowdsourced labeled data: a survey. *Artif. Intell. Rev.* 46:543–576. https://doi.org/10.1007/s10462-016-9491-9.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2017. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. In *Proceedings of the EMNLP*.

A Dataset of Peer Reviews (PeerRead): Collection, Insights and NLP Applications

Dongyeop Kang[1] **Waleed Ammar**[2] **Bhavana Dalvi Mishra**[2] **Madeleine van Zuylen**[2]
Sebastian Kohlmeier[2] **Eduard Hovy**[1] **Roy Schwartz**[2,3]
[1]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA
[2]Allen Institute for Artificial Intelligence, Seattle, WA, USA
[3]Paul G. Allen Computer Science & Engineering, University of Washington, Seattle, WA, USA
{dongyeok,hovy}@cs.cmu.edu
{waleeda,bhavanad,madeleinev,sebastiank,roys}@allenai.org

Abstract

Peer reviewing is a central component in the scientific publishing process. We present the first public dataset of scientific peer reviews available for research purposes (PeerRead v1),[1] providing an opportunity to study this important artifact. The dataset consists of 14.7K paper drafts and the corresponding accept/reject decisions in top-tier venues including ACL, NIPS and ICLR. The dataset also includes 10.7K textual peer reviews written by experts for a subset of the papers. We describe the data collection process and report interesting observed phenomena in the peer reviews. We also propose two novel NLP tasks based on this dataset and provide simple baseline models. In the first task, we show that simple models can predict whether a paper is accepted with up to 21% error reduction compared to the majority baseline. In the second task, we predict the numerical scores of review aspects and show that simple models can outperform the mean baseline for aspects with high variance such as 'originality' and 'impact'.

1 Introduction

Prestigious scientific venues use peer reviewing to decide which papers to include in their journals or proceedings. While this process seems essential to scientific publication, it is often a subject of debate. Recognizing the important consequences of peer reviewing, several researchers studied various aspects of the process, including consistency, bias, author response and general review quality (e.g., Greaves et al., 2006; Ragone et al., 2011; De Silva and Vance, 2017). For example, the organizers of the NIPS 2014 conference assigned 10% of conference submissions to two different sets of reviewers to measure the consistency of the peer reviewing process, and observed that the two committees disagreed on the accept/reject decision for more than a quarter of the papers (Langford and Guzdial, 2015).

Despite these efforts, quantitative studies of peer reviews had been limited, for the most part, to the few individuals who had access to peer reviews of a given venue (e.g., journal editors and program chairs). The goal of this paper is to lower the barrier to studying peer reviews for the scientific community by introducing the first public dataset of peer reviews for research purposes: PeerRead.

We use three strategies to construct the dataset: (i) We collaborate with conference chairs and conference management systems to allow authors and reviewers to opt-in their paper drafts and peer reviews, respectively. (ii) We crawl publicly available peer reviews and annotate textual reviews with numerical scores for aspects such as 'clarity' and 'impact'. (iii) We crawl arXiv submissions which coincide with important conference submission dates and check whether a similar paper appears in proceedings of these conferences at a later date. In total, the dataset consists of 14.7K paper drafts and the corresponding accept/reject decisions, including a subset of 3K papers for which we have 10.7K textual reviews written by experts. We plan to make periodic releases of PeerRead, adding more sections for new venues every year. We provide more details on data collection in §2.

The PeerRead dataset can be used in a variety of ways. A quantitative analysis of the peer reviews can provide insights to help better understand (and potentially improve) various nuances of the review process. For example, in §3, we analyze correlations between the overall recommendation score and individual aspect scores (e.g., clarity, impact and originality) and quantify how reviews recom-

[1]https://github.com/allenai/PeerRead

Proceedings of NAACL-HLT 2018, pages 1647–1661
New Orleans, Louisiana, June 1 - 6, 2018.

Section	#Papers	#Reviews	Asp.	Acc / Rej
NIPS 2013–2017	2,420	9,152	×	2,420 / 0
ICLR 2017	427	1,304	✓	172 / 255
ACL 2017	137	275	✓	88 / 49
CoNLL 2016	22	39	✓	11 / 11
arXiv 2007–2017	11,778	—	—	2,891 / 8,887
total	14,784	10,770		

Table 1: The PeerRead dataset. **Asp.** indicates whether the reviews have aspect specific scores (e.g., clarity). Note that ICLR contains the aspect scores assigned by our annotators (see Section 2.4). **Acc/Rej** is the distribution of accepted/rejected papers. Note that NIPS provide reviews only for accepted papers.

mending an oral presentation differ from those recommending a poster. Other examples might include aligning review scores with authors to reveal gender or nationality biases. From a pedagogical perspective, the PeerRead dataset also provides inexperienced authors and first-time reviewers with diverse examples of peer reviews.

As an NLP resource, peer reviews raise interesting challenges, both from the realm of sentiment analysis—predicting various properties of the reviewed paper, e.g., clarity and novelty, as well as that of text generation—given a paper, automatically generate its review. Such NLP tasks, when solved with sufficiently high quality, might help reviewers, area chairs and program chairs in the reviewing process, e.g., by lowering the number of reviewers needed for some paper submission.

In §4, we introduce two new NLP tasks based on this dataset: (i) predicting whether a given paper would be accepted to some venue, and (ii) predicting the numerical score of certain aspects of a paper. Our results show that we can predict the accept/reject decisions with 6–21% error reduction compared to the majority reject-all baseline, in four different sections of PeerRead. Since the baseline models we use are fairly simple, there is plenty of room to develop stronger models to make better predictions.

2 Peer-Review Dataset (PeerRead)

Here we describe the collection and compilation of PeerRead, our scientific peer-review dataset. For an overview of the dataset, see Table 1.

2.1 Review Collection

Reviews in PeerRead belong to one of the two categories:

Opted-in reviews. We coordinated with the Softconf conference management system and the conference chairs for CoNLL 2016[2] and ACL 2017[3] conferences to allow authors and reviewers to opt-in their drafts and reviews, respectively, to be included in this dataset. A submission is included only if (i) the corresponding author opts-in the paper draft, and (ii) at least one of the reviewers opts-in their anonymous reviews. This resulted in 39 reviews for 22 CoNLL 2016 submissions, and 275 reviews for 137 ACL 2017 submissions. Reviews include both text and aspect scores (e.g., calrity) on a scale of 1–5.

Peer reviews on the web. In 2013, the NIPS conference[4] began attaching all accepted papers with their anonymous textual review comments, as well as a confidence level on a scale of 1–3. We collected all accepted papers and their reviews for NIPS 2013–2017, a total of 9,152 reviews for 2,420 papers.

Another source of reviews is the OpenReview platform:[5] a conference management system which promotes open access and open peer reviewing. Reviews include text, as well as numerical recommendations between 1–10 and confidence level between 1–5. We collected all submissions to the ICLR 2017 conference,[6] a total of 1,304 official, anonymous reviews for 427 papers (177 accepted and 255 rejected).[7]

2.2 arXiv Submissions

arXiv[8] is a popular platform for pre-publishing research in various scientific fields including physics, computer science and biology. While arXiv does not contain reviews, we automatically label a subset of arXiv submissions in the years 2007–2017 (inclusive)[9] as accepted or probably-rejected, with

[2]The 20th SIGNLL Conference on Computational Natural Language Learning; `http://www.conll.org/2016`

[3]The 55th Annual Meeting of the Association for Computational Linguistics; `http://acl2017.org/`

[4]The Conference on Neural Information Processing Systems; `https://nips.cc/`

[5]`http://openreview.net`

[6]The 5th International Conference on Learning Representations; `https://iclr.cc/archive/www/2017.html`

[7]The platform also allows any person to review the paper by adding a comment, but we only use the official reviews of reviewers assigned to review that paper.

[8]`https://arxiv.org/`

[9]For consistency, we only include the first arXiv version of each paper (accepted or rejected) in the dataset.

respect to a group of top-tier NLP, ML and AI venues: ACL, EMNLP, NAACL, EACL, TACL, NIPS, ICML, ICLR and AAAI.

Accepted papers. In order to assign 'accepted' labels, we use the dataset provided by Sutton and Gong (2017) who matched arXiv submissions to their bibliographic entries in the DBLP directory[10] by comparing titles and author names using Jaccard's distance. To improve our coverage, we also add an arXiv submission if its title matches an accepted paper in one of our target venues with a relative Levenshtein distance (Levenshtein, 1966) of < 0.1. This results in a total of 2,891 accepted papers.

Probably-rejected papers. We use the following criteria to assign a 'probably-rejected' label for an arXiv submission:

- The paper wasn't accepted to any of the target venues.[11]
- The paper was submitted to one of the arXiv categories `cs.cl`, `cs.lg` or `cs.ai`.[12]
- The paper wasn't cross-listed in any non-`cs` categories.
- The submission date[13] was within one month of the submission deadlines of our target venues (before or after).
- The submission date coincides with at least one of the arXiv papers accepted for one of the target venues.

This process results in 8,887 'probably-rejected' papers.

Data quality. We did a simple sanity check in order to estimate the number of papers that we labeled as 'probably-rejected', but were in fact accepted to one of the target venues. Some authors add comments to their arXiv submissions to indicate the publication venue. We identified arXiv papers with a comment which matches the term "accept" along with any of our target venues (e.g., "nips"), but not the term "workshop". We found 364 papers which matched these criteria, 352 out of which were labeled as 'accepted'. Manual inspection of the remaining 12 papers showed that one of the papers was indeed a false negative (i.e., labeled as 'probably-rejected' but accepted to one of the target venues) due to a significant change in the paper title. The remaining 11 papers were not accepted to any of the target venues (e.g., "accepted at WMT@ACL 2014").

2.3 Organization and Preprocessing

We organize v1.0 of the PeerRead dataset in five sections: CoNLL 2016, ACL 2017, ICLR 2017, NIPS 2013–2017 and arXiv 2007–2017.[14] Since the data collection varies across sections, different sections may have different license agreements. The papers in each section are further split into standard training, development and test sets with 0.9:0.05:0.05 ratios. In addition to the PDF file of each paper, we also extract its textual content using the `Science Parse` library.[15] We represent each of the splits as a json-encoded text file with a list of paper objects, each of which consists of paper details, accept/reject/probably-reject decision, and a list of reviews.

2.4 Aspect Score Annotations

In many publication venues, reviewers assign numeric aspect scores (e.g., clarity, originality, substance) as part of the peer review. Aspect scores could be viewed as a structured summary of the strengths and weaknesses of a paper. While aspect scores assigned by reviewers are included in the opted-in sections in PeerRead, they are missing from the remaining reviews. In order to increase the utility of the dataset, we annotated 1.3K reviews with aspect scores, based on the corresponding review text. Annotations were done by two of the authors. In this subsection, we describe the annotation process in detail.

Feasibility study. As a first step, we verified the feasibility of the annotation task by annotating nine reviews for which aspect scores are available. The annotators were able to infer about half of the aspect scores from the corresponding review text (the other half was not discussed in the review text). This is expected since reviewer comments often focus on the key strengths or weaknesses of the paper and are not meant to be a comprehensive assessment of each aspect. On average, the absolute difference between our annotated scores and the gold scores originally provided by reviewers is 0.51 (on a 1–5 scale, considering only those cases where the aspect was discussed in the review text).

[10] `http://dblp.uni-trier.de/`

[11] Note that some of the 'probably-rejected' papers may be published at workshops or other venues.

[12] See `https://arxiv.org/archive/cs` for a description of the computer science categories in arXiv.

[13] If a paper has multiple versions, we consider the submission date of the first version.

[14] We plan to periodicly release new versions of PeerRead.

[15] `https://github.com/allenai/science-parse`

Data preprocessing. We used the official reviews in the ICLR 2017 section of the dataset for this annotation task. We excluded unofficial comments contributed by arbitrary members of the community, comments made by the authors in response to other comments, as well as "meta-reviews" which state the final decision on a paper submission. The remaining 1,304 official reviews are all written by anonymous reviewers assigned by the program committee to review a particular submission. We randomly reordered the reviews before annotation so that the annotator judgments based on one review are less affected by other reviews of the same paper.

Annotation guidelines. We annotated seven aspects for each review: appropriateness, clarity, originality, soundness/correctness, meaningful comparison, substance, and impact. For each aspect, we provided our annotators with the instructions given to ACL 2016 reviewers for this aspect.[16] Our annotators' task was to read the detailed review text (346 words on average) and select a score between 1–5 (inclusive, integers only) for each aspect.[17] When review comments do not address a specific aspect, we do not select any score for that aspect, and instead use a special "not discussed" value.

Data quality. In order to assess annotation consistency, the same annotators re-annotated a random sample consisting of 30 reviews. On average, 77% of the annotations were consistent (i.e., the re-annotation was exactly the same as the original annotation, or was off by 1 point) and 2% were inconsistent (i.e., the re-annotation was off by 2 points or more). In the remaining 21%, the aspect was marked as "not discussed" in one annotation but not in the other. We note that different aspects are discussed in the textual reviews at different rates. For example, about 49% of the reviews discussed the 'originality' aspect, while only 5% discussed 'appropriateness'.

3 Data-Driven Analysis of Peer Reviews

In this section, we showcase the potential of using PeerRead for data-driven analysis of peer reviews.

Overall recommendation vs. aspect scores. A critical part of each review is the overall recommendation score, a numeric value which best characterizes a reviewer's judgment of whether the draft should be accepted for publication in this venue. While aspect scores (e.g., clarity, novelty, impact) help explain a reviewer's assessment of the submission, it is not necessarily clear which aspects reviewers appreciate the most about a submission when considering their overall recommendation.

To address this question, we measure pair-wise correlations between the overall recommendation and various aspect scores in the ACL 2017 section of PeerRead and report the results in Table 2.

Aspect	ρ
Substance	0.59
Clarity	0.42
Appropriateness	0.30
Impact	0.16
Meaningful comparison	0.15
Originality	0.08
Soundness/Correctness	0.01

Table 2: Pearson's correlation coefficient ρ between the overall recommendation and various aspect scores in the ACL 2017 section of PeerRead.

The aspects which correlate most strongly with the final recommendation are substance (which concerns the amount of work rather than its quality) and clarity. In contrast, soundness/correctness and originality are least correlated with the final recommendation. These observations raise interesting questions about what we collectively care about the most as a research community when evaluating paper submissions.

Oral vs. poster. In most NLP conferences, accepted submissions may be selected for an oral presentation or a poster presentation. The presentation format decision of accepted papers is based on recommendation by the reviewers. In the official blog of ACL 2017,[18] the program chairs recommend that reviewers and area chairs make this decision based on the expected size of interested audience and whether the ideas can be grasped without back-and-forth discussion. However, it remains unclear what criteria are used by reviewers to make this decision.

To address this question, we compute the mean aspect score in reviews which recommend an oral vs. poster presentation in the ACL 2017 section of

[16] Instructions are provided in Appendix B.

[17] Importantly, our annotators only considered the review text, and did not have access to the papers.

[18] https://acl2017.wordpress.com/2017/03/23/conversing-or-presenting-poster-or-oral/

PeerRead, and report the results in Table 3. Notably, the average 'overall recommendation' score in reviews recommending an oral presentation is 0.9 higher than in reviews recommending a poster presentation, suggesting that reviewers tend to recommend oral presentation for submissions which are holistically stronger.

Presentation format	Oral	Poster	Δ	stdev
Recommendation	3.83	2.92	0.90	0.89
Substance	3.91	3.29	0.62	0.84
Clarity	4.19	3.72	0.47	0.90
Meaningful comparison	3.60	3.36	0.24	0.82
Impact	3.27	3.09	0.18	0.54
Originality	3.91	3.88	0.02	0.87
Soundness/Correctness	3.93	4.18	-0.25	0.91

Table 3: Mean review scores for each presentation format (oral vs. poster). Raw scores range between 1–5. For reference, the last column shows the sample standard deviation based on all reviews.

ACL 2017 vs. ICLR 2017. Table 4 reports the sample mean and standard deviation of various measurements based on reviews in the ACL 2017 and the ICLR 2017 sections of PeerRead. Most of the mean scores are similar in both sections, with a few notable exceptions. The comments in ACL 2017 reviews tend to be about 50% longer than those in the ICLR 2017 reviews. Since review length is often thought of as a measure of its quality, this raises interesting questions about the quality of reviews in ICLR vs. ACL conferences. We note, however, that ACL 2017 reviews were explicitly opted-in while the ICLR 2017 reviews include all official reviews, which is likely to result in a positive bias in review quality of the ACL reviews included in this study.

Another interesting observation is that the mean appropriateness score is lower in ICLR 2017 compared to ACL 2017. While this might indicate that ICLR 2017 attracted more irrelevant submissions, this is probably an artifact of our annotation process: reviewers probably only address appropriateness explicitly in their review if the paper is inappropriate, which leads to a strong negative bias against this category in our ICLR dataset.

4 NLP Tasks

Aside from quantitatively analyzing peer reviews, PeerRead can also be used to define interesting

Measurement	ACL'17	ICLR'17
Review length (words)	531±323	346±213
Appropriateness	4.9±0.4	2.6±1.3
Meaningful comparison	3.5±0.8	2.9±1.1
Substance	3.6±0.8	3.0±0.9
Originality	3.9±0.9	3.3±1.1
Clarity	3.9±0.9	4.2±1.0
Impact	3.2±0.5	3.4±1.0
Overall recommendation	3.3±0.9	3.3±1.4

Table 4: Mean ± standard deviation of various measurements on reviews in the ACL 2017 and ICLR 2017 sections of PeerRead. Note that ACL aspects were written by the reviewers themselves, while ICLR aspects were predicted by our annotators based on the review.

NLP tasks. In this section, we introduce two novel tasks based on the PeerRead dataset. In the first task, given a paper draft, we predict whether the paper will be accepted to a set of target conferences. In the second task, given a textual review, we predict the aspect scores for the paper such as novelty, substance and meaningful comparison.[19]

Both these tasks are not only challenging from an NLP perspective, but also have potential applications. For example, models for predicting the accept/reject decisions of a paper draft might be used in recommendation systems for arXiv submissions. Also, a model trained to predict the aspect scores given review comments using thousands of training examples might result in better-calibrated scores.

4.1 Paper Acceptance Classification

Paper acceptance classification is a binary classification task: given a paper draft, predict whether the paper will be accepted or rejected for a predefined set of venues.

Models. We train a binary classifier to estimate the probability of accept vs. reject given a paper, i.e., $P(\texttt{accept=True} \mid \texttt{paper})$. We experiment with different types of classifiers: logistic regression, SVM with linear or RBF kernels, Random Forest, Nearest Neighbors, Decision Tree, Multilayer Perceptron, AdaBoost, and Naive Bayes. We use hand-engineered features, instead of neural models, because they are easier to interpret.

[19] We also experiment with conditioning on the paper itself to make this prediction.

	ICLR	cs.cl	cs.lg	cs.ai
Majority	57.6	68.9	67.9	92.1
Ours	65.3	75.7	70.7	92.6
(Δ)	+7.7	+6.8	+2.8	+0.5

Table 5: Test accuracies (%) for acceptance classification. Our best model outperforms the majority classifiers in all cases.

We use 22 coarse features, e.g., length of the title and whether jargon terms such as 'deep' and 'neural' appear in the abstract, as well as sparse and dense lexical features. The full feature set is detailed in Appendix A.

Experimental setup. We experiment with the ICLR 2017 and the arXiv sections of the PeerRead dataset. We train separate models for each of the arXiv category: `cs.cl`, `cs.lg`, and `cs.ai`. We use python's sklearn's implementation of all models (Pedregosa et al., 2011).[20] We consider various regularization parameters for SVM and logistic regression (see Appendix A.1 for a detailed description of all hyperparameters). We use the standard test split and tune our hyperparameters using 5-fold cross validation on the training set.

Results. Table 5 shows our test accuracies for the paper acceptance task. Our best model outperforms the majority classifier in all cases, with up to 22% error reduction. Since our models lack the sophistication to assess the quality of the work discussed in the given paper, this might indicate that some of the features we define are correlated with strong papers, or bias reviewers' judgments.

We run an ablation study for this task for the ICLR and arXiv sections. We train only one model for all three categories in arXiv to simplify our analysis. Table 6 shows the absolute degradation in test accuracy of the best performing model when we remove one of the features. The table shows that some features have a large contribution on the classification decision: adding an appendix, a large number of theorems or equations, the average length of the text preceding a citation, the number of papers cited by this paper that were published in the five years before the submission of this paper, whether the abstract contains a phrase "state of the art" for ICLR or "neural" for arXiv, and length of title.[21]

[20] `http://scikit-learn.org/stable/`

[21] Coefficient values of each feature are provided in Appendix A.

ICLR	%
Best model	65.3
– appendix	–5.4
– num_theorems	–3.8
– num_equations	–3.8
– avg_len_ref	–3.8
– $\text{abstract}_{\text{state-of-the-art}}$	–3.5
– #recent_refs	–2.5

arXiv	%
Best model	79.1
– avg_len_ref	–1.4
– num_uniq_words	–1.1
– num_theorems	–1.0
– $\text{abstract}_{\text{neural}}$	–1.0
– num_refmentions	–1.0
– title_length	–1.0

Table 6: The absolute % difference in accuracy on the paper acceptance prediction task when we remove only one feature from the full model. Features with larger negative differences are more salient, and we only show the six most salient features for each section. The features are num_X: number of X (e.g., theorems or equations), avg_len_ref: average length of context before a reference, appendix: does paper have an appendix, abstract_X: does the abstract contain the phrase X, num_uniq_words: number of unique words, num_refmentions: number of reference mentions, and #recent_refs: number of cited papers published in the last five years.

4.2 Review Aspect Score Prediction

The second task is a multi-class regression task to predict scores for seven review aspects: 'impact', 'substance', 'appropriateness', 'comparison', 'soundness', 'originality' and 'clarity'. For this task, we use the two sections of PeerRead which include aspect scores: ACL 2017 and ICLR 2017.[22]

Models. We use a regression model which predicts a floating-point score for each aspect of interest given a sequence of tokens. We train three variants of the model to condition on (i) the paper text only, (ii) the review text only, or (iii) both paper and review text.

We use three neural architectures: convolutional neural networks (CNN, Zhang et al., 2015), recurrent neural networks (LSTM, Hochreiter and Schmidhuber, 1997), and deep averaging networks (DAN, Iyyer et al., 2015). In all three architectures, we use a linear output layer to make the final prediction. The loss function is the mean squared error between predicted and gold scores. We compare against a baseline which always predicts the mean score of an aspect, computed on the training set.[23]

Experimental setup. We train all models on the standard training set for 100 iterations, and

[22] The CoNLL 2016 section also includes aspect scores but is too small for training.

[23] This baseline is guaranteed to obtain mean square errors less than or equal to the majority baseline.

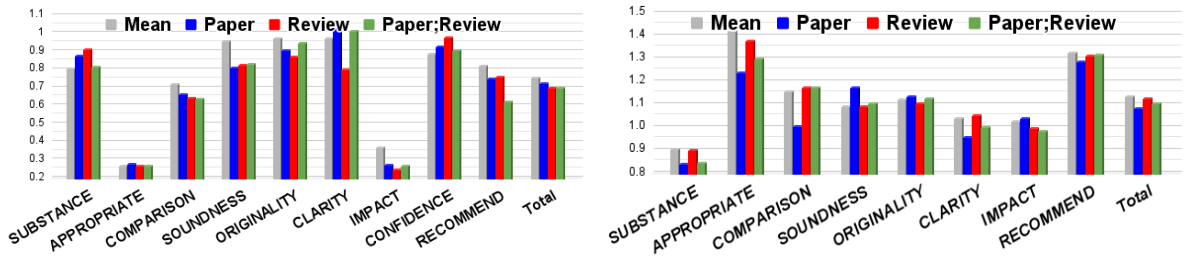


Figure 1: Root mean squared error (RMSE, lower is better) on the test set for the aspect prediction task on the ACL 2017 (left) and the ICLR 2017 (right) sections of PeerRead.

select the best performing model on the standard development set. We use a single 100 dimension layer LSTM and CNN, and a single output layer of 100 dimensions for all models. We use GloVe 840B embeddings (Pennington et al., 2014) as input word representations, without tuning, and keep the 35K most frequent words and replace the rest with an UNK vector. The CNN model uses 128 filters and 5 kernels. We use an RMSProp optimizer (Tieleman and Hinton, 2012) with 0.001 learning rate, 0.9 decay rate, 5.0 gradient clipping, and a batch size of 32. Since scientific papers tend to be long, we only take the first 1000 and 200 tokens of each paper and review, respectively, and concatenate the two prefixes when the model conditions on both the paper and review text.[24]

Results. Figure 1 shows the test set root mean square error (RMSE) on the aspect prediction task (lower is better). For each section (ACL 2017 and ICLR 2017), and for each aspect, we report the results of four systems: 'Mean' (baseline), 'Paper', 'Review' and 'Paper;Review' (i.e., which information the model conditions on). For each variant, the model which performs best on the development set is selected.

We note that aspects with higher RMSE scores for the 'Mean' baseline indicate higher variance among the review scores for this aspect, so we focus our discussion on these aspects. In the ACL 2017 section, the two aspects with the highest variance are 'originality' and 'clarity'. In the ICLR 2017 section, the two aspects with the highest variance are 'appropriateness' and 'meaningful comparison'. Surprisingly, the 'Paper;Review' model outperforms the 'Mean' baseline in all four aspects, and the 'Review' model outperforms the 'Mean' baseline in three out of four. On average, all models slightly improve over the 'Mean' baseline.

5 Related Work

Several efforts have recently been made to collect peer reviews. Publons[25] consolidates peer reviews data to build public reviewer profiles for participating reviewers. Crossref maintains the database of DOIs for its 4000+ publisher members. They recently launched a service to add peer reviews as part of metadata for the scientific articles.[26] Surprisingly, however, most of the reviews are not made publicly available. In contrast, we collected and organized PeerRead such that it is easy for other researchers to use it for research purposes, replicate experiments and make a fair comparison to previous results.

There have been several efforts to analyze the peer review process (e.g., Bonaccorsi et al., 2018; Rennie, 2016). Editors of the British Journal of Psychiatry found differences in courtesy between signed and unsigned reviews (Walsh et al., 2000). Ragone et al. (2011) and Birukou et al. (2011) analyzed ten CS conferences and found low correlation between review scores and the impact of papers in terms of future number of citations. Fang et al. (2016) presented similar observations for NIH grant application reviews and their productivity. Langford and Guzdial (2015) pointed to inconsistencies in the peer review process.

Several recent venues had single vs. double blind review experiments, which pointed to single-blind reviews leading to increased biases towards male authors (Roberts and Verhoef, 2016) and famous institutions (Tomkins et al., 2017). Further, Le Goues et al. (2017) showed that reviewers are unable to

[24] We note that the goal of this paper is to demonstrate potential uses of PeerRead, rather than develop the best model to address this task, which explains the simplicity of the models we use.

[25] publons.com/dashboard/records/review/

[26] https://www.crossref.org/blog/peer-reviews-are-open-for-registering-at-crossref/

successfully guess the identity of the author in a double-blind review. Recently, there have been several initiatives by program chairs in major NLP conferences to study various aspects of the review process, mostly author response and general review quality.[27] In this work, we provide a large scale dataset that would enable the wider scientific community to further study the properties of peer review, and potentially come up with enhancements to current peer review model.

Finally, the peer review process is meant to judge the quality of research work being disseminated to the larger research community. With the ever-growing rates of articles being submitted to top-tier conferences in Computer Science and pre-print repositories (Sutton and Gong, 2017), there is a need to expedite the peer review process. Balachandran (2013) proposed a method for automatic analysis of conference submissions to recommend relevant reviewers. Also related to our acceptance predicting task are (Tsur and Rappoport, 2009) and Ashok et al. (2013), both of which focuses on predicting book reviews. Various automatic tools like Grammerly[28] can assist reviewers in discovering grammar and spelling errors. Tools like Citeomatic[29] (Bhagavatula et al., 2018) are especially useful in finding relevant articles not cited in the manuscript. We believe that the NLP tasks presented in this paper, predicting the acceptance of a paper and the aspect scores of a review, can potentially serve as useful tools for writing a paper, reviewing it, and deciding about its acceptance.

6 Conclusion

We introduced PeerRead, the first publicly available peer review dataset for research purposes, containing 14.7K papers and 10.7K reviews. We analyzed the dataset, showing interesting trends such as a high correlation between overall recommendation and recommending an oral presentation. We defined two novel tasks based on PeerRead: (i) predicting the acceptance of a paper based on textual features and (ii) predicting the score of each aspect in a review based on the paper and review contents. Our experiments show that certain properties of a paper, such as having an appendix, are correlated with higher acceptance rate. Our primary goal is to motivate other researchers to explore these tasks and develop better models that outperform the ones used in this work. More importantly, we hope that other researchers will identify novel opportunities which we have not explored to analyze the peer reviews in this dataset. As a concrete example, it would be interesting to study if the accept/reject decisions reflect author demographic biases (e.g., nationality).

[27] See https://nlpers.blogspot.com/2015/06/some-naacl-2013-statistics-on-author.html and https://acl2017.wordpress.com/2017/03/27/author-response-does-it-help/

[28] https://www.grammarly.com/

[29] http://allenai.org/semantic-scholar/citeomatic/

Acknowledgements

This work would not have been possible without the efforts of Rich Gerber and Paolo Gai (developers of the softconf.com conference management system), Stefan Riezler, Yoav Goldberg (chairs of CoNLL 2016), Min-Yen Kan, Regina Barzilay (chairs of ACL 2017) for allowing authors and reviewers to opt-in for this dataset during the official review process. We thank the openreview.net, arxiv.org and semanticscholar.org teams for their commitment to promoting transparency and openness in scientific communication. We also thank Peter Clark, Chris Dyer, Oren Etzioni, Matt Gardner, Nicholas FitzGerald, Dan Jurafsky, Hao Peng, Minjoon Seo, Noah A. Smith, Swabha Swayamdipta, Sam Thomson, Trang Tran, Vicki Zayats and Luke Zettlemoyer for their helpful comments.

References

Vikas Ganjigunte Ashok, Song Feng, and Yejin Choi. 2013. Success with style: Using writing style to predict the success of novels. In *Proc. of EMNLP*. pages 1753–1764.

Vipin Balachandran. 2013. Reducing human effort and improving quality in peer code reviews using automatic static analysis and reviewer recommendation. In *Proc. of ICSE*.

Chandra Bhagavatula, Sergey Feldman, Russell Power, and Waleed Ammar. 2018. Content-based citation recommendation. In *Proc. of NAACL*.

Aliaksandr Birukou, Joseph R. Wakeling, Claudio Bartolini, Fabio Casati, Maurizio Marchese, Katsiaryna Mirylenka, Nardine Osman, Azzurra Ragone, Carles Sierra, and Aalam Wassef. 2011. Alternatives to peer review: Novel approaches for research evaluation. In *Front. Comput. Neurosci.*.

Andrea Bonaccorsi, Antonio Ferrara, and Marco Malgarini. 2018. Journal ratings as predictors of article

quality in arts, humanities, and social sciences: An analysis based on the italian research evaluation exercise. In *The Evaluation of Research in Social Sciences and Humanities*, Springer, pages 253–267.

Pali UK De Silva and Candace K Vance. 2017. Preserving the quality of scientific research: Peer review of research articles. In *Scientific Scholarly Communication*, Springer, pages 73–99.

Ferric C Fang, Anthony Bowen, and Arturo Casadevall. 2016. NIH peer review percentile scores are poorly predictive of grant productivity. *eLife* .

Sarah Greaves, Joanna Scott, Maxine Clarke, Linda Miller, Timo Hannay, Annette Thomas, and Philip Campbell. 2006. Nature's trial of open peer review. *Nature* 444(971):10–1038.

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

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proc. of ACL-IJCNLP*. volume 1, pages 1681–1691.

John Langford and Mark Guzdial. 2015. The arbitrariness of reviews, and advice for school administrators. *Communications of the ACM Blog* 58(4):12–13.

Claire Le Goues, Yuriy Brun, Sven Apel, Emery Berger, Sarfraz Khurshid, and Yannis Smaragdakis. 2017. Effectiveness of anonymization in double-blind review. ArXiv:1709.01609.

Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics doklady* 10(8):707–710.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *JMLR* 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Azzurra Ragone, Katsiaryna Mirylenka, Fabio Casati, and Maurizio Marchese. 2011. A quantitative analysis of peer review. In *Proc. of ISSI*.

Drummond Rennie. 2016. Make peer review scientific: thirty years on from the first congress on peer review, drummond rennie reflects on the improvements brought about by research into the process–and calls for more. *Nature* 535(7610):31–34.

Seán G Roberts and Tessa Verhoef. 2016. Double-blind reviewing at evolang 11 reveals gender bias. *Journal of Language Evolution* 1(2):163–167.

Charles Sutton and Linan Gong. 2017. Popularity of arxiv.org within computer science. ArXiv:1710.05225.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* 4(2):26–31.

Andrew Tomkins, Min Zhang, and William D Heavlin. 2017. Single versus double blind reviewing at wsdm 2017. ArXiv:1702.00502.

Oren Tsur and Ari Rappoport. 2009. Revrank: A fully unsupervised algorithm for selecting the most helpful book reviews. In *Proc. of ICWSM*.

E. Walsh, Michael W Rooney, Louis Appleby, and Greg Wilkinson. 2000. Open peer review: a randomised controlled trial. *The British journal of psychiatry : the journal of mental science* 176:47–51.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proc. of NIPS*.

Appendices

A Acceptance Classification Features

Table 7 shows the features used by our acceptance classification model. Figure 2 shows the coefficients of all our features as learned by our best classifier on both datasets.

A.1 Hyperparameters

This section describes the hyperparameters used in our acceptance classification experiment. Unless stated otherwise, we used the sklearn default hyperparameters. For decision tree and random forest, we used maximum depth=5. For the latter, we also used max_features=1. For MLP, we used $\alpha = 1$. For k-nearest neighbors, we used $k = 3$. For logistic regression, we considered both $l1$ and $l2$ penalty.

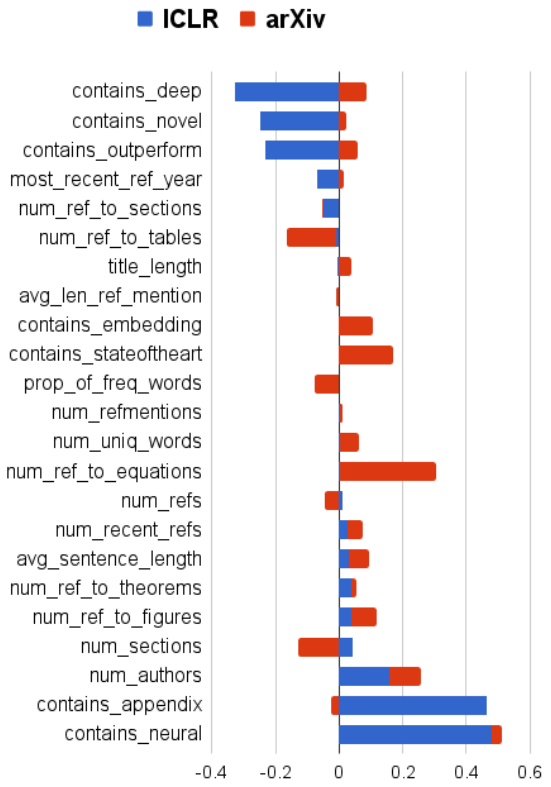


Figure 2: Coefficient values for coarse features in the paper acceptance classification, for ICLR and arXiv.

B Reviewer Instructions

Below is the list of instructions to ACL 2016 reviewers on how to assign aspect scores to reviewed papers.

	Features	Description	Labels
coarse	abstract_contains_X	Whether abstract contains keywords X ⊂ deep, neural, embedding, outperform, outperform, novel, state_of_the_art	boolean
	title_length	Length of title	integer
	num_authors	Number of authors	integer
	most_recent_refs_year	Most recent reference year	2001-2017
	num_refs	Number of references (*sp*)	integer
	num_refmentions	Number of reference mentioned (*sp*)	integer
	avg_length_refs_mention	Average length of references mentioned (*sp*)	float
	num_recent_refs	Number of recent references since the paper submitted (*sp*)	integer
	num_ref_to_X	Number of X ⊂ figures, tables, sections, equations, theorems (*sp*)	integer
	num_uniq_words	Number of unique words (*sp*)	integer
	num_sections	Number of sections (*sp*)	integer
	avg_sentence_length	Average sentence length (*sp*)	float
	contains_appendix	Whether contains an appendix or not (*sp*)	boolean
	prop_of_freq_words	Proportion of frequent words (*sp*)	float
Lexical	BOW	Bag-of-words in abstract	integer
	BOW+TFIDF	TFIDF weighted BOW in abstract	float
	GloVe	Average of GloVe word embeddings in abstract	float
	GloVe+TFIDF	TFIDF weighted average of word embeddings in abstract	float

Table 7: List of *coarse* and *lexical* features used for acceptance classification task. *sp* refers features extracted from science-parse.

APPROPRIATENESS (1-5)

Does the paper fit in ACL 2016? (Please answer this question in light of the desire to broaden the scope of the research areas represented at ACL.)

5: Certainly.
4: Probably. W
3: Unsure.
2: Probably not.
1: Certainly not.

CLARITY (1-5)
For the reasonably well-prepared reader, is it clear what was done and why? Is the paper well-written and well-structured?

5 = Very clear.
4 = Understandable by most readers.
3 = Mostly understandable to me with some effort.
2 = Important questions were hard to resolve even with effort.
1 = Much of the paper is confusing.

ORIGINALITY (1-5)
How original is the approach? Does this paper break new ground in topic, methodology, or content? How exciting and innovative is the research it describes?
Note that a paper could score high for originality even if the results do not show a convincing benefit.

5 = Surprising: Significant new problem, technique, methodology, or insight -- no prior research has attempted something similar.
4 = Creative: An intriguing problem, technique, or approach that is substantially different from previous research.
3 = Respectable: A nice research contribution that represents a notable extension of prior approaches or methodologies.
2 = Pedestrian: Obvious, or a minor improvement on familiar techniques.
1 = Significant portions have actually been done before or done better.

EMPIRICAL SOUNDNESS / CORRECTNESS (1-5)
First, is the technical approach sound and well-chosen? Second, can one trust the empirical claims of the paper -- are they supported by proper experiments and are the results of the experiments correctly interpreted?

5 = The approach is very apt, and the claims are convincingly supported.

4 = Generally solid work, although there are some aspects of the approach or evaluation I am not sure about.
3 = Fairly reasonable work. The approach is not bad, and at least the main claims are probably correct, but I am not entirely ready to accept them (based on the material in the paper).
2 = Troublesome. There are some ideas worth salvaging here, but the work should really have been done or evaluated differently.
1 = Fatally flawed.

THEORETICAL SOUNDNESS / CORRECTNESS (1-5)
First, is the mathematical approach sound and well-chosen? Second, are the arguments in the paper cogent and well-supported?

5 = The mathematical approach is very apt, and the claims are convincingly supported.
4 = Generally solid work, although there are some aspects of the approach I am not sure about or the argument could be stronger.
3 = Fairly reasonable work. The approach is not bad, and at least the main claims are probably correct, but I am not entirely ready to accept them (based on the material in the paper).
2 = Troublesome. There are some ideas worth salvaging here, but the work should really have been done or argued differently.
1 = Fatally flawed.

MEANINGFUL COMPARISON (1-5)
Do the authors make clear where the problems and methods sit with respect to existing literature? Are the references adequate? For empirical papers, are the experimental results meaningfully compared with the best prior approaches?

5 = Precise and complete comparison with related work. Good job given the space constraints.
4 = Mostly solid bibliography and comparison, but there are some references missing.
3 = Bibliography and comparison are somewhat helpful, but it could be hard for a reader to determine exactly how this work relates to previous work.
2 = Only partial awareness and understanding of related work, or a flawed empirical comparison.
1 = Little awareness of related work, or lacks necessary empirical comparison.

SUBSTANCE (1-5)
Does this paper have enough substance, or would it benefit from more ideas or results? Note that this question mainly concerns the amount of work; its quality is evaluated in other categories.

5 = Contains more ideas or results than most publications in this conference; goes the extra mile.
4 = Represents an appropriate amount of work for a publication in this conference. (most submissions)

3 = Leaves open one or two natural questions that should have been pursued within the paper.
2 = Work in progress. There are enough good ideas, but perhaps not enough in terms of outcome.
1 = Seems thin. Not enough ideas here for a full-length paper.

IMPACT OF IDEAS OR RESULTS (1-5)
How significant is the work described? If the ideas are novel, will they also be useful or inspirational? Does the paper bring any new insights into the nature of the problem?

5 = Will affect the field by altering other people's choice of research topics or basic approach.
4 = Some of the ideas or results will substantially help other people's ongoing research.
3 = Interesting but not too influential. The work will be cited, but mainly for comparison or as a source of minor contributions.
2 = Marginally interesting. May or may not be cited.
1 = Will have no impact on the field.

IMPACT OF ACCOMPANYING SOFTWARE (1-5)
If software was submitted or released along with the paper, what is the expected impact of the software package? Will this software be valuable to others? Does it fill an unmet need? Is it at least sufficient to replicate or better understand the research in the paper?

5 = Enabling: The newly released software should affect other people's choice of research or development projects to undertake.
4 = Useful: I would recommend the new software to other researchers or developers for their ongoing work.
3 = Potentially useful: Someone might find the new software useful for their work.
2 = Documentary: The new software useful to study or replicate the reported research, although for other purposes they may have limited interest or limited usability. (Still a positive rating)
1 = No usable software released.

IMPACT OF ACCOMPANYING DATASET (1-5)
If a dataset was submitted or released along with the paper, what is the expected impact of the dataset? Will this dataset be valuable to others in the form in which it is released? Does it fill an unmet need?

5 = Enabling: The newly released datasets should affect other people's choice of research or development projects to undertake.
4 = Useful: I would recommend the new datasets to other researchers or developers for their ongoing work.
3 = Potentially useful: Someone might find the new datasets useful for their work.

2 = Documentary: The new datasets are useful to study or replicate the reported research, although for other purposes they may have limited interest or limited usability. (Still a positive rating)
1 = No usable datasets submitted.

RECOMMENDATION (1-5)
There are many good submissions competing for slots at ACL 2016; how important is it to feature this one? Will people learn a lot by reading this paper or seeing it presented?

In deciding on your ultimate recommendation, please think over all your scores above. But remember that no paper is perfect, and remember that we want a conference full of interesting, diverse, and timely work. If a paper has some weaknesses, but you really got a lot out of it, feel free to fight for it. If a paper is solid but you could live without it, let us know that you're ambivalent. Remember also that the authors have a few weeks to address reviewer comments before the camera-ready deadline.

Should the paper be accepted or rejected?

5 = This paper changed my thinking on this topic and I'd fight to get it accepted;
4 = I learned a lot from this paper and would like to see it accepted.
3 = Borderline: I'm ambivalent about this one.
2 = Leaning against: I'd rather not see it in the conference.
1 = Poor: I'd fight to have it rejected.

REVIEWER CONFIDENCE (1-5)
5 = Positive that my evaluation is correct. I read the paper very carefully and am familiar with related work.
4 = Quite sure. I tried to check the important points carefully. It's unlikely, though conceivable, that I missed something that should affect my ratings.
3 = Pretty sure, but there's a chance I missed something. Although I have a good feel for this area in general, I did not carefully check the paper's details, e.g., the math, experimental design, or novelty.
2 = Willing to defend my evaluation, but it is fairly likely that I missed some details, didn't understand some central points, or can't be sure about the novelty of the work.
1 = Not my area, or paper is very hard to understand. My evaluation is just an educated guess.

Deep Communicating Agents for Abstractive Summarization

Asli Celikyilmaz[1], **Antoine Bosselut**[2], **Xiaodong He**[3] and **Yejin Choi**[2,4]
[1]Microsoft Research
[2]Paul G. Allen School of Computer Science & Engineering, University of Washington
[3]JD AI Research
[4]Allen Institute for Artificial Intelligence
{aslicel}@microsoft.com {xiaodong.he}@jd.com
{antoineb, yejin}@cs.washington.edu

Abstract

We present *deep communicating agents* in an encoder-decoder architecture to address the challenges of representing a long document for abstractive summarization. With deep communicating agents, the task of encoding a long text is divided across multiple collaborating agents, each in charge of a subsection of the input text. These encoders are connected to a single decoder, trained end-to-end using reinforcement learning to generate a focused and coherent summary. Empirical results demonstrate that multiple communicating encoders lead to a higher quality summary compared to several strong baselines, including those based on a single encoder or multiple non-communicating encoders.

1 Introduction

We focus on the task of *abstractive* summarization of a *long* document. In contrast to *extractive* summarization, where a summary is composed of a subset of sentences or words lifted from the input text as is, *abstractive* summarization requires the generative ability to rephrase and restructure sentences to compose a coherent and concise summary. As recurrent neural networks (RNNs) are capable of generating fluent language, variants of encoder-decoder RNNs (Sutskever et al., 2014; Bahdanau et al., 2015) have shown promising results on the abstractive summarization task (Rush et al., 2015; Nallapati et al., 2017).

The fundamental challenge, however, is that the strong performance of neural models at encoding short text does not generalize well to long text. The motivation behind our approach is to be able to dynamically attend to different parts of the input to capture salient facts. While recent work in summarization addresses these issues using improved attention models (Chopra et al., 2016), pointer networks with coverage mechanisms (See et al., 2017), and coherence-focused training objectives (Paulus et al., 2018; Jaques et al., 2017), an effective mechanism for representing a long document remains a challenge.

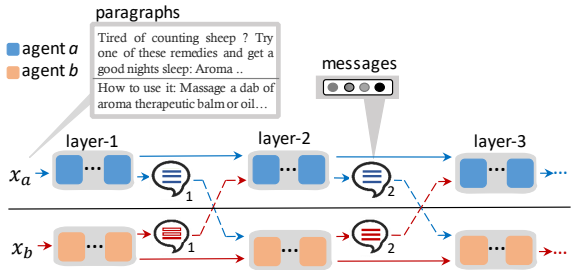


Figure 1: Illustration of deep communicating agents presented in this paper. Each agent a and b encodes one paragraph in multiple layers. By passing new messages through multiple layers the agents are able to coordinate and focus on the important aspects of the input text.

Simultaneous work has investigated the use of deep communicating agents (Sukhbaatar et al., 2016) for collaborative tasks such as logic puzzles (Foerster et al., 2016), visual dialog (Das et al., 2017), and reference games (Lazaridou et al., 2016). Our work builds on these approaches to propose the first study on using communicating agents to encode long text for summarization.

The key idea of our model is to divide the hard task of encoding a long text across multiple collaborating encoder agents, each in charge of a different subsection of the text (Figure 1). Each of these agents encodes their assigned text independently, and broadcasts their encoding to others, allowing agents to share global context information with one another about different sections of the document. All agents then adapt the encoding of their assigned text in light of the global context and re-

Proceedings of NAACL-HLT 2018, pages 1662–1675
New Orleans, Louisiana, June 1 - 6, 2018.

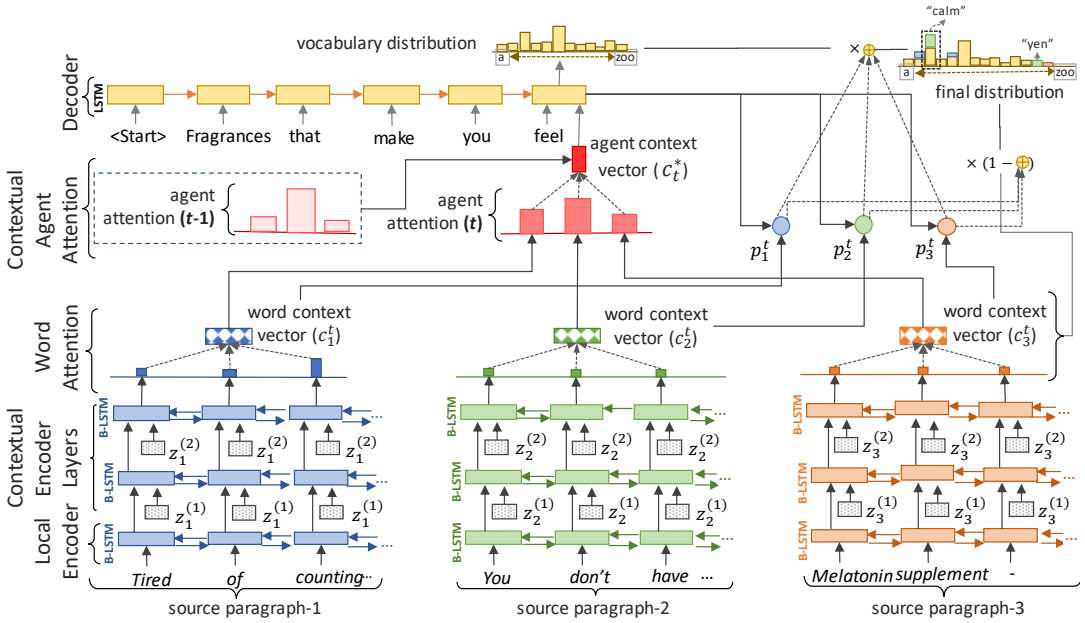


Figure 2: Multi-agent-encoder-decoder overview. Each agent a encodes a paragraph using a local encoder followed by multiple contextual layers with agent communication through concentrated messages $z_a^{(k)}$ at each layer k. **Communication is illustrated in Figure 3**. The word context vectors c_a^t are condensed into agent context c_t^*. Agent specific generation probabilities, p_a^t, enable voting for the suitable out-of-vocabulary words (e.g., '*yen*') in the final distribution.

peat the process across multiple layers, generating new messages at each layer. Once each agent completes encoding, they deliver their information to the decoder with a novel *contextual agent attention* (Figure 2). Contextual agent attention enables the decoder to integrate information from multiple agents smoothly at each decoding step. The network is trained end-to-end using self-critical reinforcement learning (Rennie et al., 2016) to generate focused and coherent summaries.

Empirical results on the CNN/DailyMail and New York Times datasets demonstrate that multiple communicating encoders lead to higher quality summaries compared to strong baselines, including those based on a single encoder or multiple non-communicating encoders. Human evaluations indicate that our model is able to produce more focused summaries. The agents gather salient information from multiple areas of the document, and communicate their information with one another, thus reducing common mistakes such as missing key facts, repeating the same content, or including unnecessary details. Further analysis reveals that our model attains better performance when the decoder interacts with multiple agents in a more balanced way, confirming the benefit of representing a long document with multiple encoding agents.

2 Model

We extend the CommNet model of Sukhbaatar et al. (2016) for sequence generation.

Notation Each document d is a sequence of paragraphs x_a, which are split across multiple encoding agents a=1,..,M (e.g., agent-1 encodes the first paragraph x_1, agent-2 the second paragraph x_2, so on). Each paragraph x_a=$\{w_{a,i}\}^I$, is a sequence of I words. We construct a V-sized vocabulary from the training documents from the most frequently appearing words. Each word $w_{a,i}$ is embedded into a n-dimensional vector $\mathbf{e}_{a,i}$. All W variables are linear projection matrices.

2.1 Multi-Agent Encoder

Each agent encodes the word sequences with the following two stacked encoders.

Local Encoder The first layer is a local encoder of each agent a, where the tokens of the corresponding paragraph x_a are fed into a single layer bi-directional LSTM (bLSTM), producing the local encoder hidden states, $h_i^{(1)} \in \mathcal{R}^H$:

$$\overrightarrow{h}_i^{(1)}, \overleftarrow{h}_i^{(1)} = \texttt{bLSTM}(e_i, \overrightarrow{h}_{i-1}^{(1)}, \overleftarrow{h}_{i+1}^{(1)}) \quad (1)$$

$$h_i^{(1)} = W_1[\overrightarrow{h}_i^{(1)}, \overleftarrow{h}_i^{(1)}] \quad (2)$$

where H is the hidden state dimensionality. The

output of the local encoder layer is fed into the contextual encoder.

Contextual Encoder Our framework enables agent communication cycles across multiple encoding layers. The output of each contextual encoder is an adapted representation of the agent's encoded information conditioned on the information received from the other agents. At each layer k=1,..,K, each agent a jointly encodes the information received from the previous layer (see Figure 3). Each cell of the (k+1)th contextual layer is a bLSTM that takes three inputs: the hidden states from the adjacent LSTM cells, $\overrightarrow{h}_{i-1}^{(k+1)} \in R^H$ or $\overleftarrow{h}_{i+1}^{(k+1)} \in R^H$, the hidden state from the previous layer $h_i^{(k)}$, and the message vector from other agents $z^{(k)} \in R^H$ and outputs $h_i^{(k+1)} \in R^H$:

$$\overrightarrow{h}_i^{(k+1)}, \overleftarrow{h}_i^{(k+1)} = \texttt{bLSTM}(f(h_i^{(k)}, z^{(k)}), \overrightarrow{h}_{i-1}^{(k+1)}, \overleftarrow{h}_{i+1}^{(k+1)}) \quad (3)(4)$$

$$h_i^{(k+1)} = W_2[\overrightarrow{h}_i^{(k+1)}, \overleftarrow{h}_i^{(k+1)}] \quad (5)$$

where i=1..I indicates the index of each token in the sequence.

The message $z^{(k)}$ received by any agent a in layer k is the average of the outputs of the other agents from layer k:

$$z^{(k)} = \frac{1}{M-1} \sum_{m \neq a} h_{m,I}^{(k)} \quad (6)$$

where $h_{m,I}^{(k)}$ is the last hidden state output from the kth contextual layer of each agent where $m \neq a$. Here, we take the average of the messages received from other encoder agents, but a parametric function such as a feed forward model or an attention over messages could also be used.

The message $z^{(k)}$ is projected with the agent's previous encoding of its document:

$$f(h_i^{(k)}, z^{(k)}) = v_1^{\mathrm{T}} \tanh(W_3 h_i^{(k)} + W_4 z^{(k)}) \quad (7)$$

where v_1, W_3, and W_4 are learned parameters shared by every agent. Equation (7) combines the information sent by other agents with the context of the current token from this paragraph. This yields different features about the current context in relation to other topics in the source document. At each layer, the agent modifies its representation of its own context relative to the information received from other agents, and updates the information it sends to other agents accordingly.

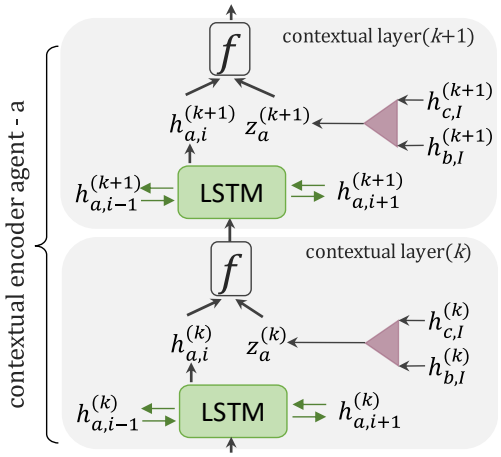


Figure 3: Multi-agent encoder message passing. Agents b and c transmit the last hidden state output (I) of the current layer k as a message, which are passed through an average pool (Eq. (6)). The receiving agent a uses the new message $z_a^{(k)}$ as additional input to its next layer.

2.2 Decoder with Agent Attention

The output from the last contextual encoder layer of each agent $\{h_{a,i}^{(K)}\}^I$, which is a sequence of hidden state vectors of each token i, is sent to the decoder to calculate word-attention distributions. We use a single-layer LSTM for the decoder and feed the last hidden state from the first agent $s_0 = h_{1,I}^{(K)}$ as the initial state. At each time step t, the decoder predicts a new word in the summary w_t and computes a new state s_t by attending to relevant input context provided by the agents.

The decoder uses a new hierarchical attention mechanism over the agents. First, a ***word attention*** distribution l_a^t (Bahdanau et al. (2015)) is computed over every token $\{h_{a,i}^{(K)}\}^I$ for each agent a:

$$l_a^t = \text{softmax}(v_2^{\mathrm{T}} \tanh(W_5 h_a^{(K)} + W_6 s_t + b_1)) \quad (8)$$

where $l_a^t \in [0,1]^I$ is the attention over all tokens in a paragraph x_a and v_2, W_5, W_6, b_1 are learned parameters. For each decoding step t, a new decoder context is calculated for each agent:

$$c_a^t = \sum_i l_{a,i}^t h_{a,i}^{(K)} \quad (9)$$

which is the weighted sum of the encoder hidden states of agent a. Each *word context vector* represents the information extracted by the agent from the paragraph it has read. Here the decoder has to decide which information is more relevant to the

current decoding step t. This is done by weighting each context vector by an agent attention yielding the document global ***agent attention*** distribution g^t (see Figure 2):

$$g^t = \text{softmax}(v_3^{\text{T}}\tanh(W_7 c^t + W_8 s_t + b_2)) \quad (10)$$

where v_3, W_7, W_8, and b_2 are learned, and $g^t \in [0,1]^M$ is a soft selection over M agents. Then, we compute the agent context vector c_t^*:

$$c_t^* = \sum_a g_a^t c_a^t \quad (11)$$

The agent context $c_t^* \in R^H$ is a fixed length vector encoding salient information from the entire document provided by the agents. It is then concatenated with the decoder state s_t and fed through a multi-layer perception to produce a vocabulary distribution (over all vocabulary words) at time t:

$$P^{voc}(w_t|s_t, w_{t-1}) = \text{softmax}(\text{MLP}([s_t, c_t^*])) \quad (12)$$

To keep the topics of generated sentences intact, it is reasonable that the decoder utilize the same agents over the course of short sequences (e.g., within a sentence). Because the decoder is designed to select which agent to attend to at each time step, we introduce contextual agent attention (*caa*) to prevent it from frequently switching between agents. The previous step's agent attention c_{t-1}^* is used as additional information to the decoding step to generate a distribution over words:

$$P^{voc}(w_t|\cdot) = \text{softmax}(\text{MLP}([s_t, c_t^*, c_{t-1}^*])) \quad (13)$$

2.3 Multi-Agent Pointer Network

Similar to See et al. (2017), we allow for copying candidate words from different paragraphs of the document by computing a *generation* probability value for each agent $p_a^t \in [0,1]$ at each timestep t using the context vector c_a^t, decoder state s_t and decoder input y_t:

$$p_a^t = \sigma(v_5^{\text{T}} c_a^t + v_6^{\text{T}} s_t + v_7^{\text{T}} y_t + b) \quad (14)$$

where b is a learned scalar, y_t is the ground-truth/predicted output (depending on the training/testing time). The generation probability determines whether to generate a word from the vocabulary by sampling from $P^{voc}(w|\cdot)$, or copying a word from the corresponding agent's input paragraph x_a by sampling from its attention distribution l_a^t. This produces an extended vocabulary that includes words in the document that are considered out-of-vocabulary (OOV). A probability distribution over the extended vocabulary is computed for each agent:

$$P^a(w_t|\cdot) = p_a^t P^{voc}(w_t|\cdot) + (1 - p_a^t) u_{a,w}^t \quad (15)$$

where $u_{a,w}^t$ is the sum of the attention for all instances where w appears in the source document. The final distribution over the extended vocabulary, from which we sample, is obtained by weighting each agent by their corresponding agent attention values g_a^t:

$$P(w_t|s_t, w_{t-1}) = \sum_a g_a^t P^a(w_t|\cdot) \quad (16)$$

In contrast to a single-agent baseline (See et al., 2017), our model allows each agent to vote for different OOV words at time t (Equation (16)). In such a case, only the word that is relevant to the generated summary up to time t is collaboratively voted as a result of agent attention probability g_a^t.

3 Mixed Objective Learning

To train the deep communicating agents, we use a mixed training objective that jointly optimizes multiple losses, which we describe below.

MLE Our baseline multi-agent model uses maximum likelihood training for sequence generation. Given $y^* = \{y_1^*, y_2^*, ..., y_T^*\}$ as the ground-truth output sequence (human summary word sequences) for a given input document d, we minimize the negative log-likelihood of the target word sequence:

$$L_{\text{MLE}} = -\sum_{t=1}^{N} \log p(y_t^*|y_1^* \ldots y_{t-1}^*, d) \quad (17)$$

Semantic Cohesion To encourage sentences in the summary to be informative without repetition, we include a *semantic cohesion* loss to integrate sentence-level semantics into the learning objective. As the decoder generates the output word sequence $\{y_1, y_2 \ldots y_T\}$, it keeps track of the end of sentence delimiter token ('.') indices. The hidden state vectors at the end of each sentence s'_q, q=1...Q, where $s'_q \in \{s_t : y_t = \text{'.'}, 1 \le t \le T\}$, are used to compute the cosine similarity between two consecutively generated sentences. To minimize the similarity between end-of-sentence hidden states we define a *semantic cohesion* loss:

$$L_{\text{SEM}} = \sum_{q=2}^{Q} \cos(s'_q, s'_{q-1}) \quad (18)$$

The final training objective is then:

$$L_{\text{MLE-SEM}} = L_{\text{MLE}} + \lambda L_{\text{SEM}} \quad (19)$$

where λ is a tunable hyperparameter.

Reinforcement Learning (RL) Loss Policy gradient methods can directly optimize discrete target evaluation metrics such as ROUGE that are non-differentiable (Paulus et al., 2018; Jaques et al., 2017; Pasunuru and Bansal, 2017; Wu et al., 2016). At each time step, the word generated by the model can be viewed as an action taken by an RL agent. Once the full sequence $\hat{y}$ is generated, it is compared against the ground truth sequence y^* to compute the reward $r(\hat{y})$.

Our model learns using a *self-critical training* approach (Rennie et al., 2016), which learns by exploring new sequences and comparing them to the best greedily decoded sequence. For each training example d, two output sequences are generated: $\hat{y}$, which is sampled from the probability distribution at each time step, $p(\hat{y}_t|\hat{y}_1 \dots \hat{y}_{t-1}, d)$, and $\tilde{y}$, the baseline output, which is greedily generated by argmax decoding from $p(\tilde{y}_t|\tilde{y}_1 \dots \tilde{y}_{t-1}, d)$. The training objective is then to minimize:

$$L_{\text{RL}} = (r(\tilde{y}) - r(\hat{y})) \sum_{t=1}^{N} \log p(\hat{y}_t|\hat{y}_1 \dots \hat{y}_{t-1}, d) \quad (20)$$

This loss ensures that, with better exploration, the model learns to generate sequences $\hat{y}$ that receive higher rewards compared to the baseline $\tilde{y}$, increasing overall reward expectation of the model.

Mixed Loss While training with only MLE loss will learn a better language model, this may not guarantee better results on global performance measures. Similarly, optimizing with only RL loss may increase the reward gathered at the expense of diminished readability and fluency of the generated summary (Paulus et al., 2018). A combination of the two objectives can yield improved task-specific scores while maintaining fluency:

$$L_{\text{MIXED}} = \gamma L_{\text{RL}} + (1 - \gamma) L_{\text{MLE}} \quad (21)$$

where γ is a tunable hyperparameter used to balance the two objective functions. We pre-train our models with MLE loss, and then switch to the mixed loss. We can also add the semantic cohesion loss term: $L_{\text{MIXED-SEM}} = \gamma L_{\text{RL}} + (1-\gamma) L_{\text{MLE-SEM}}$ to analyze its impact in RL training.

Intermediate Rewards We introduce sentence-based rewards as opposed to end of summary rewards, using differential ROUGE metrics, to promote generating diverse sentences. Rather than rewarding sentences based on the scores obtained at the end of the generated summary, we compute incremental rouge scores of a generated sentence $\hat{o}_q$:

$$r(\hat{o}_q) = r([\hat{o}_1, \dots \hat{o}_q]) - r([\hat{o}_1, \dots \hat{o}_{q-1}]) \quad (22)$$

Sentences are rewarded for the increase in ROUGE they contribute to the full sequence, ensuring that the current sentence contributed novel information to the overall summary.

4 Experimental Setup

Datasets We conducted experiments on two summarization datasets: CNN/DailyMail (Nallapati et al., 2017; Hermann et al., 2015) and New York Times (NYT) (Sandhaus, 2008). We replicate the preprocessing steps of Paulus et al. (2018) to obtain the same data splits, except that we do not anonymize named entities. For our DCA models, we initialize the number of agents before training, and partition the document among the agents (i.e., three agent $\rightarrow$ three paragraphs). Additional details can be found in Appendix A.1.

Training Details During training and testing we truncate the article to 800 tokens and limit the length of the summary to 100 tokens for training and 110 tokens at test time. We distribute the truncated articles among agents for multi-agent models, preserving the paragraph and sentences as possible. For both datasets, we limit the input and output vocabulary size to the 50,000 most frequent tokens in the training set. We train with up to two contextual layers in all the DCA models as more layers did not provide additional performance gains. We fix $\gamma = 0.97$ for the RL term in Equation (21) and $\lambda = 0.1$ for the SEM term in MLE and MIXED training. Additional details are provided in Appendix A.2.

Evaluation We evaluate our system using ROUGE-1 (unigram recall), ROUGE-2 (bigram recall) and ROUGE-L (longest common sequence).[1] We select the MLE models with the lowest negative log-likelihood and the MLE+RL models with the highest ROUGE-L scores on a sample of validation data to evaluate on the test

[1] We use `pyrouge` (pypi.python.org/pypi/pyrouge/0.1.3).

Model	ROUGE-1	ROUGE-2	ROUGE-L
SummaRuNNer (Nallapati et al., 2017)	39.60	16.20	35.30
graph-based attention (Tan et al., 2017)	38.01	13.90	34.00
pointer generator (See et al., 2017)	36.44	15.66	33.42
pointer generator + coverage (See et al., 2017)	39.53	17.28	36.38
controlled summarization with fixed values (Fan et al., 2017)	39.75	17.29	36.54
RL, with intra-attention (Paulus et al., 2018)	41.16	15.75	**39.08**
ML+RL, with intra-attention(Paulus et al., 2018)	39.87	15.82	36.90
(**m1**) MLE, pgen, no-comm (1-agent) (our baseline-1)	36.12	14.38	33.83
(**m2**) MLE+SEM, pgen, no-comm (1-agent) (our baseline-2)	36.90	15.02	33.00
(**m3**) MLE+RL, pgen, no-comm (1-agent) (our baseline-3)	38.01	16.43	35.49
(**m4**) DCA MLE+SEM, pgen, no-comm (3-agents)	37.45	15.90	34.56
(**m5**) DCA MLE+SEM, *mpgen*, with-comm (3-agents)	39.52	17.12	36.90
(**m6**) DCA MLE+SEM, *mpgen*, with-comm, with *caa* (3-agents)	41.11	**18.21**	36.03
(**m7**) DCA MLE+SEM+RL, *mpgen*, with-comm, with *caa* (3-agents)	**41.69**	**19.47**	37.92

Table 1: Comparison results on the **CNN/Daily Mail** test set using the **F1** variants of **Rouge**. Best model models are bolded.

Model	Rouge-1	Rouge-2	Rouge-L
ML, no intra-attention (Paulus et al., 2018)	44.26	27.43	40.41
RL, no intra-attention (Paulus et al., 2018)	47.22	30.51	**43.27**
ML+RL, no intra-attention(Paulus et al., 2018)	47.03	30.72	43.10
(**m1**) MLE, pgen, no-comm (1-agent) (our baseline-1)	44.28	26.01	37.87
(**m2**) MLE+SEM, pgen, no-comm (1-agent) (our baseline-2)	44.50	28.04	38.80
(**m3**) MLE+RL, pgen, no-comm (1-agent) (our baseline-3)	46.15	29.50	39.38
(**m4**) DCA MLE+SEM, pgen, no-comm (3-agents)	45.84	28.23	39.32
(**m5**) DCA MLE+SEM, *mpgen*, with-comm (3-agents)	46.20	30.01	40.65
(**m6**) DCA MLE+SEM, *mpgen*, with-comm, with *caa* (3-agents)	**47.30**	30.50	41.06
(**m7**) DCA MLE+SEM+RL, *mpgen* with-comm, with *caa* (3-agents)	**48.08**	**31.19**	42.33

Table 2: Comparison results on the **New York Times** test set using the **F1** variants of **Rouge**. Best model models are bolded.

set. At test time, we use beam search of width 5 on all our models to generate final predictions.

Baselines We compare our DCA models against previously published models: SummaRuNNer (Nallapati et al., 2017), a graph-based attentional neural model (Tan et al., 2017) an RNN-based extractive summarizer that combines abstractive features during training; Pointer-networks with and without coverage (See et al., 2017), RL-based training for summarization with intra-decoder attention (Paulus et al., 2018)), and Controllable Abstractive Summarization (Fan et al., 2017) which allows users to define attributes of generated summaries and also uses a copy mechanism for source entities and decoder attention to reduce repetition.

Ablations We investigate each new component of our model with a different ablation, producing seven different models. Our first three ablations are: a single-agent model with the same local encoder, context encoder, and pointer network architectures as the DCA encoders trained with MLE loss (**m1**); the same model trained with additional semantic cohesion SEM loss (**m2**), and the same model as the (**m1**) but trained with a mixed loss and end-of-summary rewards (**m3**).

The rest of our models use 3 agents and incrementally add one component. First, we add the semantic cohesion loss (**m4**). Then, we add multi-agent pointer networks (*mpgen*) and agent communication (**m5**). Finally, we add contextual agent attention (caa) (**m6**), and train with the mixed MLE+RL+SEM loss (**m7**). All DCA models use pointer networks.

5 Results

5.1 Quantitative Analysis

We show our results on the CNN/DailyMail and NYT datasets in Table 1 and 2 respectively. Overall, our (**m6**) and (**m7**) models with multi-agent encoders, pointer generation, and communication are the strongest models on ROUGE-1 and ROUGE-2. While weaker on ROUGE-L than the RL model from Paulus et al. (2018), the human evaluations in that work showed that their model received lower *readability* and *relevance* scores than a model trained with MLE, indicating the additional boost in ROUGE-L was not correlated with summary quality. This result can also account for our best models being more abstractive. Our models use mixed loss not just to op-

timize for sentence level structure similarity with the reference summary (to get higher ROUGE as reward), but also to learn parameters to improve semantic coherence, promoting higher abstraction (see Table 4 and Appendix B for generated summary examples).

Model	ROUGE-1	ROUGE-2	ROUGE-L
2-agent	40.94	19.16	37.54
3-agent	**41.69**	**19.47**	37.92
5-agent	40.99	19.02	**38.21**

Table 3: Comparison of multi-agent models varying the number of agents using ROUGE results of model (m7) from Table 1 on CNN/Daily Maily Dataset.

Single vs. Multi-Agents All multi-agent models show improvements over the single agent baselines. On the CNN/DailyMail dataset, compared to MLE published baselines, we improve across all ROUGE scores. We found that the 3-agent models generally outperformed both 2- and 5-agent models (see Table 3). This is in part because we truncate documents before training and the larger number of agents might be more efficient for multi-document summarization.

Independent vs. Communicating Agents When trained on multiple agents with no communication (**m4**), the performance of our DCA models is similar to the single agent baselines (**m1**) and (**m3**). With communication, the biggest jump in ROUGE is seen on the CNN/DailyMail data, indicating that the encoders can better identify the key facts in the input, thereby avoiding unnecessary details.

Contextual Agent Attention (*caa*) Compared to the model with no *contextualized agent attention* (**m5**), the (**m6**) model yields better ROUGE scores. The stability provided by the *caa* helps the decoder avoid frequent switches between agents that would dilute the topical signal captured by each encoder.

Repetition Penalty As neurally generated summaries can be redundant, we introduced the semantic cohesion penalty and incremental rewards for RL to generate semantically diverse summaries. Our baseline model optimized together with SEM loss **(m2)** improves on all ROUGE scores over the baseline **(m1)**. Similarly, our model trained with reinforcement learning uses sentence based intermediate rewards, which also improves ROUGE scores across both datasets.

5.2 Human Evaluations

We perform human evaluations to establish that our model's ROUGE improvements are correlated with human judgments. We measure the communicative multi-agent network with contextual agent attention in comparison to a single-agent network with no communication. We use the following as evaluation criteria for generated summaries: (1) *non-redundancy*, fewer of the same ideas are repeated, (2) *coherence*, ideas are expressed clearly; (3) *focus*, the main ideas of the document are shared while avoiding superfluous details, and (4) *overall*, the summary effectively communicates the article's content. The focus and non-redundancy dimensions help quantify the impact of multi-agent communication in our model, while coherence helps to evaluate the impact of the reward based learning and repetition penalty of the proposed models.

Evaluation Procedure We randomly selected 100 samples from the CNN/DailyMail test set and use workers from Amazon Mechanical Turk as judges to evaluate them on the four criteria defined above. Judges are shown the original document, the ground truth summary, and two model summaries and are asked to evaluate each summary on the four criteria using a Likert scale from 1 (worst) to 5 (best). The ground truth and model summaries are presented to the judges in random order. Each summary is rated by 5 judges and the results are averaged across all examples and judges.

We also performed a head-to-head evaluation (more common in DUC style evaluations) and randomly show two model generated summaries. We ask the human annotators to rate each summary on the same metrics as before without seeing the source document or ground truth summaries.

Results Human evaluators significantly prefer summaries generated by the communicating encoders. In the rating task, evaluators preferred the multi-agent summaries to the single-agent cases for all metrics. In the head-to-head evaluation, humans consistently preferred the DCA summaries to those generated by a single agent. In both the head-to-head and the rating evaluation, the largest improvement for the DCA model was on the *focus* question, indicating that the model learns to generate summaries with more pertinent details by capturing salient information from later portions of the document.

Human	Mr Turnbull was interviewed about his childhood and his political stance. He also admitted he **planned to run for prime minister if Tony Abbott had been successfully toppled in February's leadership spill.** The words 'primed minister' were controversially also printed on the cover.
Single	Malcolm Turnbull is set to feature on the front cover of the GQ Australia in a bold move that will no doubt set senators' tongues wagging. Posing in a suave blue suit with a pinstriped shirt and a contrasting red tie , Mr Turnbull's confident demeanour is complimented by the bold, confronting words printed across the page: 'primed minister'.
Multi	Malcolm Turnbull was set to **run for prime minister if Tony Abbott had been successfully toppled in February's leadership spill**. He is set to feature on the front cover of the liberal party's newsletter.
Human	Daphne Selfe has been modelling since the fifties. She has recently landed a new campaign with vans and & other stories. **The 86-year-old commands 1,000 a day for her work.**
Single	Daphne Selfe, 86, shows off the collaboration between the footwearsuper-brandand theetherealhigh street store with uncompromisinggrace. Daphne said of the collection , in which she appears with 22-year-old flo dron: 'the & other stories collection that is featured in this story is truly relaxed and timeless with a modern twist'. The shoes are then worn with pieces from the brands ss2015 collection.
Multi	Daphne Selfe, 86, has starred in the campaign for vans and & other stories. The model appears with 22-year-old flo dron & other hair collection . **She was still commanding 1,000 a day for her work.**

Table 4: Comparison of a human summary to best single- and multi-agent model summaries, (m3) and (m7) from CNN/DailyMail dataset. Although single-agent model generates a coherent summary, it is less focused and contains more unnecessary details (highlighed red) and misses **keys facts** that the multi-agent model successfully captures (**bolded**).

	Head-to-Head			Score Based	
Criteria	SA	MA	=	SA	MA
non-redundancy	68	**159**	73	4.384	**4.428**
coherence	89	**173**	38	3.686	**3.754**
focus	83	**181**	36	3.694	**3.884***
overall	102	**158**	40	3.558	**3.682***

Table 5: Head-to-Head and score-based comparison of human evaluations on random subset of CNN/DM dataset. SA=single, MA=multi-agent. * indicates statistical significance at $p < 0.001$ for focus and $p < 0.03$ for the overall.

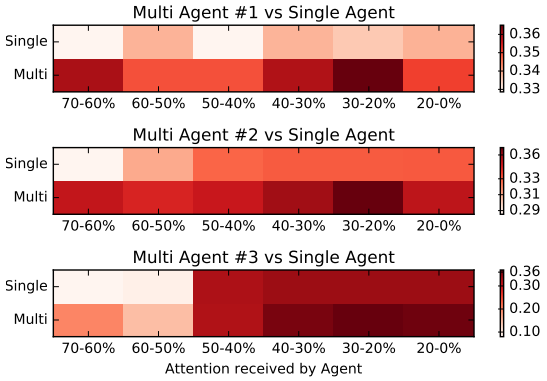


Figure 4: The average ROUGE-L scores for summaries that are binned by each agent's average attention when generating the summary (see Section 5.2). When the agents contribute equally to the summary, the ROUGE-L score increases.

5.3 Communication improves focus

To investigate how much the multi-agent models discover salient concepts in comparison to single agent models, we analyze ROUGE-L scores based on the average attention received by each agent. We compute the average attention received by each agent per decoding time step for every generated summary in the CNN/Daily Mail test corpus, bin the document-summary pairs by the attention received by each agent, and average the ROUGE-L scores for the summaries in each bin.

Figure 4 outlines two interesting results. First, summaries generated with a more distributed attention over the agents yield higher ROUGE-L scores, indicating that attending to multiple areas of the document allows the discovery of salient concepts in the later sections of the text. Second, if we use the same bins and generate summaries for the documents in each bin using the single-agent model, the average ROUGE-L scores for the single-agent summaries are lower than for the corresponding multi-agent summaries, indicating that even in cases where one agent dominates the attention, communication between agents allows the model to generate more focused summaries.

Qualitatively, we see this effect in Table 4, where we compare the human generated summaries against our best single agent model (**m3**) and our best multi-agent model (**m7**). Model (**m3**) generates good summaries but does not capture all the facts in the human summary, while (**m7**) is able to include all the facts with few extra details, generating more relevant and diverse summaries.

6 Related Work

Several recent works investigate attention mechanisms for encoder-decoder models to sharpen the

context that the decoder should focus on within the input encoding (Luong et al., 2015; Vinyals et al., 2015b; Bahdanau et al., 2015). For example, Luong et al. (2015) proposes global and local attention networks for machine translation, while others investigate hierarchical attention networks for document classification (Yang et al., 2016), sentiment classification (Chen et al., 2016), and dialog response selection (Zhou et al., 2016).

Attention mechanisms have shown to be crucial for summarization as well (Rush et al., 2015; Zeng et al., 2016; Nallapati et al., 2017), and pointer networks (Vinyals et al., 2015a), in particular, help address redundancy and saliency in generated summaries (Cheng and Lapata, 2016; See et al., 2017; Paulus et al., 2018; Fan et al., 2017). While we share the same motivation as these works, our work uniquely presents an approach based on CommNet, the deep communicating agent framework (Sukhbaatar et al., 2016). Compared to prior multi-agent works on logic puzzles (Foerster et al., 2017), language learning (Lazaridou et al., 2016; Mordatch and Abbeel, 2017) and starcraft games (Vinyals et al., 2017), we present the first study in using this framework for long text generation.

Finally, our model is related to prior works that address repetitions in generating long text. See et al. (2017) introduce a post-trained coverage network to penalize repeated attentions over the same regions in the input, while Paulus et al. (2018) use intra-decoder attention to punish generating the same words. In contrast, we propose a new semantic coherence loss and intermediate sentence-based rewards for reinforcement learning to discourage semantically similar generations (§3).

7 Conclusions

We investigated the problem of encoding long text to generate abstractive summaries and demonstrated that the use of deep communicating agents can improve summarization by both automatic and manual evaluation. Analysis demonstrates that this improvement is due to the improved ability of covering all and only salient concepts and maintaining semantic coherence in summaries.

References

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

Huimin Chen, Maosong Sun, Cunchao Tu, Yankai Lin, and Zhiyuan Liu. 2016. Neural sentiment classification with user and product attention. In *EMNLP*.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *ACL*.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *NAACL*.

Abhishek Das, Satwik Kottur, Jos M. F. Moura, and Dhruv Batra Stefan Lee. 2017. Learning cooperative visual dialog agents with deep reinforcement learning. In *CVPR*.

Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. 2016. Learning-based single-document summarization with compression and anaphoricity constraints. In *ACL*.

Angela Fan, David Grangier, and Michael Auli. 2017. Controllable abstractive summarization. In *arXiv:1711.05217v1*.

J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson. 2017. Counterfactual multi-agent policy gradients. In *arXiv:1705.08926*.

Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. 2016. Learning to communicate with deep multi-agent reinforcement learning. In *NIPS*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*.

Kai Hong, Michel Marcus, and Ani Nenkova. 2015. System combination for multi-document summarization. In *EMNLP*.

Kai Hong and Ani Nenkova. 2014. Improving the estimation of word importance for news multi-document summarization. In *Extended technical report*.

Natasha Jaques, Shixiang Gu, Dzmitry Bahdanau, Jose Miguel Hernandez-Lobato, Richard E. Turner, and Douglas Eck. 2017. Sequence tutor: Conservative fine-tuning of sequence generation models with kl-control. In *ICML*.

Angeliki Lazaridou, Nghia The Pham, and Marco Baroni. 2016. Towards multi-agent communication-based language learning. In *arXiv:1605.07133*.

Junyi Jessy Li, Kapil Thandani, and Amanda Stent. 2016. The role of discourse units in near-extractive summarization. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialog*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

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

I. Mordatch and P. Abbeel. 2017. Emergence of grounded compositional language in multi-agent populations. In *arXiv:1703.04908*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Association for the Advancement of Artificial Intelligence*.

Ramakanth Pasunuru and Mohit Bansal. 2017. Reinforced video captioning with entailment rewards. In *EMNLP*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. *ICLR* .

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing*.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2016. Self-critical sequence training for image captioning. *arXiv preprint arXiv:1612.00563* .

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *EMNLP*.

Evan Sandhaus. 2008. New york times annotated corpus. In *Linguistic Data Consortium, Philadelphia*.

Abigale See, Peter J. Liu, and Christopher Manning. 2017. Gettothepoint: Summarization with pointer-generatornetworks. In *ACL*.

Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. 2016. Learning multiagent communication with back-propagation. In *NIPS*.

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

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *ACL*.

O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Kuttler, J. Agapiou, and J. et al. Schrittwieser. 2017. Starcraft ii: A new challenge for reinforcement learning. In *arXiv preprint arXiv:1708.04782*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015a. Pointer networks. In *NIPS*.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015b. Grammar as a foreign language. In *NIPS*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* .

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *NAACL*.

Wenyuan Zeng, Wenjie Luo, Sanja Fidler, and Raquel Urtasun. 2016. Efcient summarization with read-again and copy mechanism. In *arXiv preprint arXiv:1611.03382*.

Xiangyang Zhou, Daxiang Dong, Hua Wu, Shiqi Zhao, D Yu R Yan, Xuan Liu, and H Tian. 2016. Multiview response selection for human-computer conversation. In *EMNLP*.

A Supplementary Material

Stats	CNN/DM	NYT
Avg. # tokens document	781	549
Avg. # tokens summary	56	40
Total # train doc-summ. pair	287,229	589,284
Total # validation doc-summ. pair	13,368	32,736
Total # test doc-summ. pair	11,490	32,739
Input token length	400/800	800
Output token length	100	100
(2-agent) Input token length / agent	375	400
(3-agent) Input token length / agent	250	200
(5-agent) Input token length / agent	150	160

Table 6: Summary statistics of CNN/DailyMail (DM) and New York Times (NYT) Datasets.

A.1 Datasets

CNN/DailyMail: CNN/DailyMail dataset (Nallapati et al., 2017; Hermann et al., 2015) is a collection of online news articles along with multi-sentence summaries. We use the same data splits as in Nallapati et al. (2017). While earlier work anonymized entities by replacing each named entity with a unique identifier (e.g., *Dominican Republic*→`entity15`), we opted for non-anonymized version.

New York Times (NYT): Although this dataset has mainly been used to train extractive summarization systems (Hong and Nenkova, 2014; Hong et al., 2015; Li et al., 2016; Durrett et al., 2016), it has recently been used for the abstractive summarization task (Paulus et al., 2018). NYT dataset (Sandhaus, 2008) is a collection of articles published between 1996 and 2007. We use the scripts provided in Li et al. (2016) to extract and preprocess the NYT dataset with some modifications in order to replicate the pre-processing steps presented in Paulus et al. (2018). Similar to (Paulus et al., 2018), we sorted the documents by their publication date in chronological order and used the first 90% for training, the next 5% for validation and last 5% for testing. They also use pointer supervision by replacing all named entities in the abstract if the type is "PERSON", "LOCATION", "ORGANIZATION" or "MISC" using the Stanford named entity recognizer (Manning et al., 2014). By contrast, we did not anonymize the NYT dataset to reduce pre-processing.

A.2 Training Details

We train our models on an NVIDIA P100 GPU machine. We set the hidden state size of the encoders and decoders to 128. For both datasets, we limit the input and output vocabulary size to the 50,000 most frequent tokens in the training set. We initialize word embeddings with 200-d GloVe vectors (Pennington et al., 2014) and fine-tune them during training. We train using Adam with a learning rate of 0.001 for the MLE models and 10^{-5} for the MLE+RL models. We tune the $gamma$ hyper-parameter in the mixed loss by iterating γ={0.95, 0.97, 0.99}. In almost all DCA models, the 0.97 value yielded the best gains. We train our models for 200,000 iterations. which took 4-5 days for 2-3 agents and 5-6 days for 5 agents since it has more encoder parameters to tune.

To avoid repetition, we prevent the decoder from generating the same trigram more than once during test, following Paulus et.al. (2018). In addition, for every predicted out-of-vocabulary token (UNK), we replace it with its most likely origin by choosing the source word w with the largest cascaded attention $w := \arg\max_{a,i} l^t_{a,i} * g^t_a$ (Eq. (8), (10)).

B Generated Summary Examples

This appendix provides example documents from the test set, with side-by-side comparisons of the human generated (golden) summaries and the summaries produced by our models. **Baseline** is a **single-agent** model trained with MLE+RL loss, **(m3)** model in Table 1, while our best **multi-agent** model is optimized by mixed MLE+SEM+RL loss, the **(m7)** model in Table 1.

- red highlights : indicate details that should not appear in the summary but the models generated them.
- red : indicates factual errors in the summary.
- green highlights : indicate key facts in the human (gold) summary that only one of the models manage to capture.

Document	model abbey clancy is helping to target breast cancer , by striking a sultry pose in a new charity campaign . the winner of 2013 's strictly come dancing joins singer foxes , 25 , victoria 's secret angel lily donaldson , 28 , and model alice dellal , 27 , in the new series of pictures by photographer simon emmett for fashion targets breast cancer . clancy , 29 , looks chic as she shows off her famous legs , wearing just a plain white shirt . abbey clancy leads the glamour as she joins forces with her famous friends to target breast cancer , by striking a sultry pose in a new charity campaign the model , who is mother to four - year - old daughter sophia with footballer husband peter crouch , said: ' as a mum , it makes me proud to be part of a campaign that funds vital work towards ensuring the next generation of young women do not have be afraid of a diagnosis of breast cancer . ' i'm wearing my support , and i want everyone across the uk to do the same and get behind this campaign . ' holding onto heaven singer foxes looks foxy in cropped stripy top and jeans . abbey says she is proud to be part of a campaign that funds vital work towards ensuring the next generation of young women do not have be afraid of a diagnosis of breast cancer victoria 's secret angel lily donaldson , who has been in the industry for years , also adds some glamour to the charity campaign holding onto heaven singer foxes dons a stripy top and jeans for the campaign she says she 's ' honoured ' to be a part of she said: ' i'm so honoured to be taking part in this year 's fashion targets breast cancer , and becoming part of the campaign 's awesome heritage . ' fashion is a huge part of my life , and if by taking part i can inspire women to wear their support , join the fight and take on breast cancer head on , then that will be something to be really proud of . ' now in its 19th year , the campaign has so far raised 13 . 5m for breakthrough breast cancer 's research funding . this year the range of clothes and accessories have been produced in conjunction with high street partners m&s , river island , warehouse , topshop , laura ashley , debenhams , superga , baukjen and the cambridge satchel company . they can be viewed online at www . fashiontargetsbreastcancer . org . uk/lookbook the campaign , which also stars alice dellal , has so far raised 13 . 5m for breakthrough breast cancer 's research funding
Human (Gold)	models abbey and lily are joined by alice dellal and singer foxes . the women are pictured ' wearing ' their support . abbey , 29 , says she is proud to be part of a campaign that funds vital work . campaign has raised 13 . 5m for breakthrough breast cancer 's research .
Single Agent Baseline	strictly come dancing joins singer foxes , 25 , victoria 's secret angel lily donaldson , 28 , and model alice dellal , 27 , in the new series of pictures by photographer simon emmett for fashion targets breast cancer . clancy , 29 , looks chic as she shows off her famous legs , wearing just a plain white shirt .
Multi Agent	abbey says she is proud to be part of a campaign that funds vital work towards ensuring the next generation of young women do not have been afraid of a diagnosis of breast cancer . the campaign has raised 13 . 5m for breakthrough breast cancer 's research .

Table 7: In this example both single- and multi-agent models demonstrate extractive behaviors. However, each select sentences from different sections of the document. While the single model extracts the second and the third sentences, the multi-agent model successfully selects salient sentences from sentences that are further down in the document, specifically sentence 8 and 10. This can be attributed to the fact that agents can successfully encode salient aspects distributed in distant sections of the document. An interesting result is that even though the multi-agent model shows extractive behaviour in this example, it successfully selects the most salient sentences while the single agent model includes superfluous details.

Document	michelle pfeiffer is the latest hollywood star preparing to hit the small screen . the oscar nominated star known for her roles in iconic films such as scarface , dangerous liaisons andthe age of innocence , has teamed up with katie couric to pitch a new television comedy about a morning news program . also involved in the project , in which pfeiffer is attached to star , is diane english , the creator of murphy brown . scroll down for video michelle pfeiffer (left) is set to star in a new television comedy about a morning news program produced by katie couric (right) the series was created by diane english (above with candice bergen) , who was behind the show murphy brown , about a female news anchor according to variety , pfeiffer 's role will be that of a morning news anchor , making it very similar to the real life role couric played as co - host of today for 15 years . **couric will serve as an executive producer and help ' ensure the series strikes realistic notes . ' the creator behind the project , english , was previously the brains behind brown , the show starring candice bergen that centered around a female news anchor and ran for ten seasons , winning 18 emmys . english would also serve as a writer , producer and showrunner on the program**. the ladies are currently in talks with hbo , showtime , amc , netflix and amazon to pick up the program . couric will serve as an executive producer , drawing on her experience as an anchor on today for 15 years pfeiffer would be the one of the biggest stars yet to move to television , joining a group that now includes house of cards stars robin wright and kevin spacey , true detective leads matthew mcconaughey and woody harrelson , and even lady gaga , who recently announced she would be appearing on the next season of american horror story . the actress has kept a low profile for the past 20 years since becoming a mother , only doing a handful of films in that time . she most recently appeared alongside robert de niro in the mob comedy ' the family . '
Human (Gold)	michelle pfeiffer is set to star in a new television comedy about a morning news program . katie couric will serve as an executive producer , drawing on her experience as an anchor on today for 15 years . the series was created by diane english , who was behind the show murphy brown , about a female news anchor . the ladies are currently in talks with hbo , showtime , amc , netflix and amazon to pick up the program .
Single Agent Baseline	the oscar nominated star known for her roles in iconic filmssuch as scarface , dangerous liaisons and the age of innocence , has teamed up with katie couric to pitch a new television comedy about a morning news program . also involved in the project , in which pfeiffer is attached to star , is diane english , the creator of murphy brown .
Multi Agent	michelle pfeiffer is set to star in a new tv comedy about a morning news program . couric will serve as an executive producer and showrunner on the project . the series was created by diane english , the creator of murphy brown . pfeiffer is the one of the biggest stars .

Table 8: The baseline model generates non-coherent summary that references the main character "Michelle Pfeiffer" in an ambiguous way towards the end of the generated summary. In contrast, the multi-agent model successfully captures the main character including the **key facts**. One interesting feature that the multi-agent model showcases is its simplification property, which accounts for its strength in **abstraction**. Specifically, it simplified the **bold** long sentence in the document starting with "*couric will...* and only generated the salient words.

Document	everton manager roberto martinez was forced to defend another penalty fiasco at the club after ross barkley missed from the spot in their 1 - 0 win against burnley at goodison park . the untried barkley inexplicably took the 10th minute kick awarded for a foul by david jones on aaron lennon rather than leighton baines , who has scored 15 penalties from 16 attempts in the premier league . although there was no dispute between the team - mates this time , it brought back memories of everton 's match against west brom in january when kevin mirallas grabbed the ball from baines to take a penalty - and missed . ross barkley steps up to take a 10th minute penalty despite the presence of leighton baines on the pitch barkley 's effort is saved byburnley goalkeeper tom heaton at goodison park martinez insisted barkley was within his rights to request penalty - taking duties on saturday . ' if romelu lukaku had been on the pitch , he would have taken it . otherwise , i am happy to have three or four players who can take penalties and let it depend on how they feel at that moment , ' argued the everton manager . baines (left)has scored 15 penalties from 16 attempts in the premier league ' ross showed incredible responsibility to take it . i love seeing players take control of the big moments and leighton was happy to given him that responsibility . ' barkley 's penalty was well - struck but wasn't put in the corner and burnley goalkeeper tom heaton dived to his right to save . fortunately for the young england player , it didn't prove costly as mirallas went on to score the only goal of the game after 29 minutes . everton boss roberto martinez issues instructions to his players during a break in play against burnley
Human (Gold)	everton defeated burnley 1 - 0 at goodison park on saturday . kevin mirallas scored the only goal of the game in the 29th minute . ross barkley had earlier missed a 10th - minute penalty . leighton baines has scored 15 penalties from 16 attempts this season .
Single Agent Baseline	everton manager roberto martinez was forced to defend another penalty fiasco at the club after ross barkley missed from the spot in their 1 - 0 win against burnley at goodison park . the untried barkley inexplicably took the 10th minute kick awarded for a foul by david jones on aaron lennon rather than leighton baines , who has scored 15 penalties from 16 attempts in the premier league .
Multi Agent	everton beat burnley 1 - 0 at goodison park in the premier league . ross barkley steps up to take a 10th minute penalty but missed it . barkley has scored 15 penalties from 16 attempts in the pitch .

Table 9: The single agent model generates summary with superfluous details and the facts are not clearly expressed. Although it was able to capture the statistics of the player correctly (e.g., *15 penalties*, *16 attempts*), it still missed the player who scored the only goal in the game (i.e., *kevin mirallas*). On the other hand multi-agent model was able to generate a concise summary with several key facts. However, similar to single agent model, it missed to capture the player who scored the only goal in the game. Interestingly, the document contains the word "*defeated*' but the multi-agent model chose to use *beat* instead, which does not exist in the original document.

Encoding Conversation Context for Neural Keyphrase Extraction from Microblog Posts

Yingyi Zhang[†*] **Jing Li**[‡] **Yan Song**[‡] **Chengzhi Zhang**[†]
[†]Nanjing University of Science and Technology
{yingyizhang, zhangcz}@njust.edu.cn
[‡]Tencent AI Lab
{ameliajli,clksong}@tencent.com

Abstract

Existing keyphrase extraction methods suffer from data sparsity problem when they are conducted on short and informal texts, especially microblog messages. Enriching context is one way to alleviate this problem. Considering that conversations are formed by reposting and replying messages, they provide useful clues for recognizing essential content in target posts and are therefore helpful for keyphrase identification. In this paper, we present a neural keyphrase extraction framework for microblog posts that takes their conversation context into account, where four types of neural encoders, namely, averaged embedding, RNN, attention, and memory networks, are proposed to represent the conversation context. Experimental results on Twitter and Weibo datasets[1] show that our framework with such encoders outperforms state-of-the-art approaches.

1 Introduction

The increasing popularity of microblogs results in a huge volume of daily-produced user-generated data. As a result, such explosive growth of data far outpaces human beings' reading and understanding capacity. Techniques that can automatically identify critical excerpts from microblog posts are therefore in growing demand. Keyphrase extraction is one of the techniques that can meet this demand, because it is defined to identify salient phrases, generally formed by one or multiple words, for representing key focus and main topics for a given collection (Turney, 2000; Zhao et al., 2011). Particularly for microblogs, keyphrase extraction has been proven useful to downstream applications such as information retrieval (Choi et al., 2012), text summarization (Zhao et al., 2011), event tracking (Ribeiro et al., 2017), etc.

Target post for keyphrase extraction:
"I will curse you in that forum" is the lowest of low. You are an embarrassment **president Duterte**. Childish!
Messages forming a conversation:
[R1]: any *head of state* will be irked if asked to report to *another head of state*
[R2]: Really? Did *Obama* really asked *Duterte* to report to him? LOL

Table 1: An example conversation about "president Duterte" on Twitter. [Ri]: The i-th message in conversation ordered by their positing time. **president Duterte**: keyphrase to be detected; *Italic words*: words that are related to the main topic in conversations and can indicate the keyphrase.

To date, most efforts on keyphrase extraction on microblogs treat messages as independent documents or sentences, and then apply ranking-based models (Zhao et al., 2011; Bellaachia and Al-Dhelaan, 2012; Marujo et al., 2015) or sequence tagging models (Zhang et al., 2016) on them. It is arguable that these methods are suboptimal for recognizing salient content from short and informal messages due to the severe data sparsity problem. Considering that microblogs allow users to form conversations on issues of interests by *reposting with comments*[2] and *replying to messages* for voicing opinions on previous discussed points, these conversations can enrich context for short messages (Chang et al., 2013; Li et al., 2015), and have been proven useful for identifying topic-related content (Li et al., 2016). For example, Table 1 displays a target post with keyphrase "*president Duterte*" and its reposting and replying messages forming a conversation.

Easily identified, critical words are mentioned multiple times in conversations. Such as in [R2], keyword "*Duterte*" re-occurs in the conversation.

[*]Work was done during the internship at Tencent AI Lab.

[1]Our datasets are released at: http://ai.tencent.com/ailab/Encoding_Conversation_Context_for_Neural_Keyphrase_Extraction_from_Microblog_Posts.html

[2]On Twitter, reposting behavior is named as retweet.

Proceedings of NAACL-HLT 2018, pages 1676–1686
New Orleans, Louisiana, June 1 - 6, 2018.

Also, topic-relevant content, e.g., "*head of state*", "*another head of state*", "*Obama*", helps to indicate keyphrase "*president Duterte*". Such contextual information embedded in a conversation is nonetheless ignored for keyphrase extraction in existing approaches.

In this paper, we present a neural keyphrase extraction framework that exploits conversation context, which is represented by neural encoders for capturing salient content to help in indicating keyphrases in target posts. Conversation context has been proven useful in many NLP tasks on social media, such as sentiment analysis (Ren et al., 2016), summarization (Chang et al., 2013; Li et al., 2015), and sarcasm detection (Ghosh et al., 2017). We use four context encoders in our model, namely, averaged embedding, RNN (Pearlmutter, 1989), attention (Bahdanau et al., 2014), and memory networks (Weston et al., 2015), which are proven useful in text representation (Cho et al., 2014; Weston et al., 2015; Huang et al., 2016; Nie et al., 2017). Particularly in this task, to the best of our knowledge, we are the first to encode conversations for detecting keyphrases in microblog posts. Experimental results on Twitter and Sina Weibo datasets demonstrate that, by effectively encoding context in conversations, our proposed approach outperforms existing approaches by a large margin. Quantitative and qualitative analysis suggest that our framework performs robustly on keyphrases with various length. Some encoders such as memory networks can detect salient and topic-related content, whose occurrences are highly indicative of keyphrases. In addition, we test ranking-based models with and without considering conversations. The results also confirm that conversation context can boost keyphrase extraction of ranking-based models.

2 Keyphrase Extraction with Conversation Context Encoding

Our keyphrase extraction framework consists of two parts, i.e., a keyphrase tagger and a conversation context encoder. The keyphrase tagger aims to identify keyphrases from a target post, and the context encoders captures the salient content in conversations, which would indicate keyphrases in the target post. The entire framework is learned synchronously with the given target posts and their corresponding conversation context. In prediction, the keyphrase tagger identifies keyphrases in a post with the help of representations generated by the encoder. Figure 1 shows the overall structure of our keyphrase extraction framework. In the rest of this section, Section 2.1 describes the keyphrase taggers used in our framework; Section 2.2 gives the details of different context encoders.

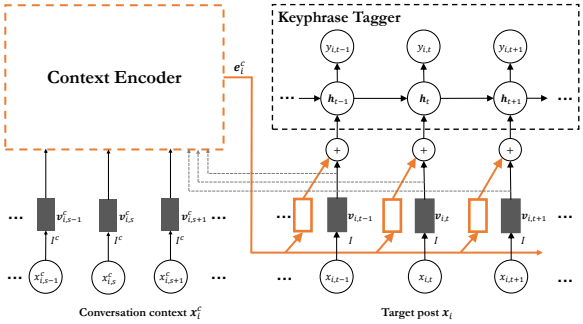


Figure 1: The overall structure of our keyphrase extraction framework with context encoder. Grey dotted array refer to the inputs of target posts that are also used in context encoding.

SINGLE	$x_{i,t}$ is a one-word keyphrase (keyword).
BEGIN	$x_{i,t}$ is the first word of a keyphrase.
MIDDLE	$x_{i,t}$ is part of a keyphrase but it is neither the first nor the last word of the keyphrase.
END	$x_{i,t}$ is the last word of a keyphrase
NOT	$x_{i,t}$ is not a keyword or part of a keyphrase.

Table 2: Definitions of different $y_{i,t}$.

2.1 Keyphrase Taggers

We follow Zhang et al. (2016) to cast keyphrase extraction into the sequence tagging task. Formally, given a target microblog post $\mathbf{x}_i$ formulated as word sequence $< x_{i,1}, x_{i,2}, ..., x_{i,|\mathbf{x}_i|} >$, where $|\mathbf{x}_i|$ denotes the length of $\mathbf{x}_i$, we aim to produce a tag sequence $< y_{i,1}, y_{i,2}, ..., y_{i,|\mathbf{x}_i|} >$, where $y_{i,t}$ indicates whether $x_{i,t}$ is part of a keyphrase. In detail, $y_{i,t}$ has five possible values:

$$y_{i,t} \in \{\text{SINGLE, BEGIN, MIDDLE, END, NOT}\}$$

Table 2 lists the definition of each value. Zhang et al. (2016) has shown that keyphrase extraction methods with this 5-value tagset perform better than those with binary outputs, i.e., only marked with yes or no for a word to be part of a keyphrase.

To predict keyphrase tags, we use four state-of-the-art neural sequence taggers, namely, recurrent neural networks (RNN) (Pearlmutter, 1989), RNN with gated recurrent units (GRU) (Chung et al., 2014), long short-term memory (LSTM) networks (Hochreiter and Schmidhuber, 1997), and bidirectional LSTM (BiLSTM) (Graves and Schmidhuber, 2005).

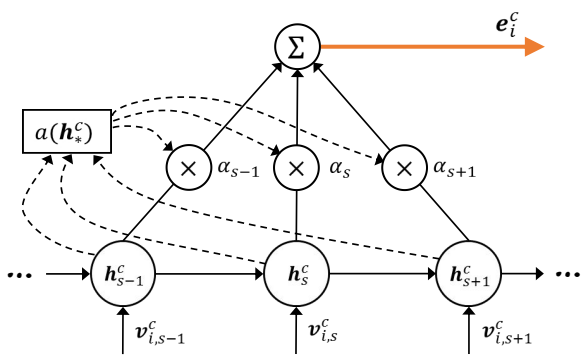


Figure 2: The structure of attention-based conversation context encoder.

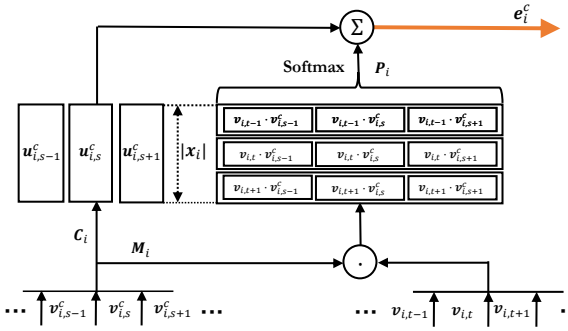


Figure 3: The structure of the conversation context encoder based on memory networks.

In addition to one-type output, we also use joint-layer RNN proposed by Zhang et al. (2016), which is demonstrated to be the state-of-the-art keyphrase tagger in previous work without modeling conversation context. As a multi-task learner (Collobert and Weston, 2008), joint-layer RNN tackles two tasks with two types of outputs, $y_{i,t}^1$ and $y_{i,t}^2$. $y_{i,t}^1$ has a binary tagset, which indicates whether word $x_{i,t}$ is part of a keyphrase or not. $y_{i,t}^2$ employs the 5-value tagset defined in Table 2. Besides the standard RNN version, in implementation, we also build the joint-layer RNN with its GRU, LSTM, and BiLSTM counterparts. To be consistent, taggers with one-type output with the 5-value tagset are named as single-layer taggers.

As shown in Figure 1, our keyphrase tagger is built upon input feature map $I(\cdot)$, which embeds each word $x_{i,t}$ in target post into a dense vector format, i.e., $I(x_{i,t}) = \boldsymbol{\nu}_{i,t}$. We initialize input feature map by pre-trained embeddings, and update embeddings during training.

2.2 Context Encoders

We aggregate all reposting and replying messages in conversations to form a pseudo-document as *context* by their posting time, and input context in forms of word sequences into context encoder. Let $\mathbf{x}_i^c$ denote the context word sequence of the target post $\mathbf{x}_i$, we propose four methods to encode $\mathbf{x}_i^c$, namely, *averaged embedding*, *RNN*, *attention*, and *memory networks*. Similar to keyphrase taggers (see Section 2.1), each word $x_{i,s}^c$ in context $\mathbf{x}_i^c$ takes the form of a vector $\boldsymbol{\nu}_{i,s}^c$ mapped by an input layer $I^c(\cdot)$, which is also initialized by pre-trained embeddings, and updated in the training process.

2.2.1 Averaged Embedding

As a straightforward sentence representation technique, averaged embedding simply takes the average embeddings of words in a context, i.e., $\boldsymbol{\nu}_{i,s}^c$, as the encoding of context representation, i.e.,

$$\mathbf{e}_i^c = \frac{1}{|\mathbf{x}_i^c|} \sum_{s=1}^{|\mathbf{x}_i^c|} \boldsymbol{\nu}_{i,s}^c \tag{1}$$

where $|\mathbf{x}_i^c|$ is the length of $\mathbf{x}_i^c$ in the context.

2.2.2 RNN

RNN encoders employ the recurrent neural network model for the embedded context sequence $< \boldsymbol{\nu}_{i,1}^c, \boldsymbol{\nu}_{i,2}^c, ..., \boldsymbol{\nu}_{i,|\mathbf{x}_i^c|}^c >$, through the recurrent functions over all the states:

$$\mathbf{h}_{i,s}^c = \delta_h(\mathbf{W}_h^1 \mathbf{h}_{i,s-1}^c + \mathbf{W}_h^2 \boldsymbol{\nu}_{i,s}^c) \tag{2}$$

where $\mathbf{W}_h^1$ and $\mathbf{W}_h^2$ are learnable weight matrices, and δ_h is the component-wise sigmoid function. The encoder representation is thus given by the hidden units at the last state:

$$\mathbf{e}_i^c = \mathbf{h}_{|\mathbf{x}_i^c|}^c \tag{3}$$

In this paper, RNN-based encoders have four variants, namely, RNN, GRU, LSTM, and BiLSTM. Particularly, as BiLSTM has two opposite directions, its context representation takes the concatenation of the last states from both directions, which come from two ends of a given context.

2.2.3 Attention

Attention-based encoders put attention mechanism (Bahdanau et al., 2014) upon RNN model for "soft-addressing" important words in the conversation context. In this paper, we use the feed-forward attention (Raffel and Ellis, 2015; SØnderby et al., 2015), as shown in Figure 2. The encoder is thus represented as

$$\mathbf{e}_i^c = \sum_{s=1}^{|x_i^c|} \alpha_{i,s}^c \mathbf{h}_{i,s}^c \tag{4}$$

where $\alpha^c_{i,s}$ is the attention coefficient obtained for word $\mathbf{x}^c_s$, which implicitly reflects its importance for helping keyphrase identification. $\alpha^c_{i,s}$ is computed via a softmax over the hidden states by

$$\alpha^c_{i,s} = softmax(a(\mathbf{h}^c_{i,s})) \quad (5)$$

where $a(\cdot)$ is a learnable function formulated as:

$$a(\mathbf{h}^c_{i,s}) = tanh(\mathbf{W}_a \mathbf{h}^c_{i,s}) \quad (6)$$

which takes input only from on $\mathbf{h}^c_{i,s}$. $\mathbf{W}_a$ are parameters of the function $a(\cdot)$ to be learned.

2.2.4 Memory Networks

The encoder based on memory networks (MemNN) (Weston et al., 2015) stores and updates the representations of conversation contexts in a memory module. The updated representations are used to guide the keyphrase tagger. Figure 3 illustrates its structure.

Formally, each embedded context sequence $\mathbf{V}^c_i =< \boldsymbol{\nu}^c_{i,1}, \boldsymbol{\nu}^c_{i,2}, ..., \boldsymbol{\nu}^c_{i,|\mathbf{x}^c_i|} >$ is stored into memory $\mathbf{M}_i$. We then yield the match between embedded target post $\mathbf{V}_i =< \boldsymbol{\nu}_{i,1}, \boldsymbol{\nu}_{i,2}, ..., \boldsymbol{\nu}_{i,|\mathbf{x}_i|} >$ and context memory $\mathbf{M}_i$ by their inner product activated by softmax:

$$\mathbf{P}_i = softmax(\mathbf{V}_i \cdot \mathbf{M}_i) \quad (7)$$

where $\mathbf{P}_{i,j,j'}$ captures the similarity between the j-th word in conversation context $\mathbf{x}^c_i$ and the j'-th word in target post $\mathbf{x}_i$.

To transform context input $\mathbf{x}^c_i$ into an aligned form so that it is able to be added with $\mathbf{P}_i$, we include another embedding matrix $\mathbf{C}_i =< \boldsymbol{\mu}_{i,1}, ..., \boldsymbol{\mu}_{i,|\mathbf{x}^c_i|} >$. Similar to attention encoder, the MemNN encoder aims to generate a representation, which addresses the important part in the conversation context that helps tagging keyphrases in target post $\mathbf{x}_i$. The sum of $\mathbf{C}_i$ and matching matrix $\mathbf{P}_i$ serves as the encoded representation for conversation context:

$$\mathbf{e}^c_i = \mathbf{P}_i + \mathbf{C}_i \quad (8)$$

In particular, both attention and MemNN explores salient words in conversations that describe main focus of the conversation, which helps indicate keyphrases of a target post. In comparison, MemNN explicitly exploits the affinity of target posts and conversations in matching each other, while attention implicitly highlights certain context without taking target posts into account.

Dataset	# of annot. msgs	# of msgs in context	Context length	Vocab
Twitter				
Train	3,976	3.38	49.74	34,412
Dev	497	3.19	46.44	7,186
Test	497	3.30	48.09	8,779
Weibo				
Train	13,816	1.97	55.77	25,259
Dev	1,727	2.01	45.00	9,106
Test	1,727	1.82	51.95	9,305

Table 3: Statistics of two datasets. Train, Dev, and Test denotes training, development, and test set, respectively. # of annot. msgs: number of messages with keyphrase annotation, each containing conversation context. # of msgs in context: average count of message in conversation context. Context length: average count of words in conversation context. Vocab: vocabulary size.

3 Experiment Setup

3.1 Datasets

Our experiments are conducted on two datasets collected from Twitter and Weibo[3], respectively. The **Twitter** dataset is constructed based on TREC2011 microblog track[4]. To recover conversations, we used Tweet Search API[5] to retrieve full information of a tweet with its "in reply to status id" included. Recursively, we searched the "in reply to" tweet till the entire conversation is recovered. Note that we do not consider retweet relations, i.e., reposting behaviors on Twitter, because retweets provide limited extra textual information for the reason that Twitter did not allow users to add comments in retweets until 2015. To build the **Weibo** dataset, we tracked real-time trending hashtags[6] on Weibo and used the hashtag-search API[7] to crawl the posts matching the given hashtag queries. In the end, a large-scale Weibo corpus is built containing Weibo messages posted during January 2nd to July 31st, 2014.

For keyphrase annotation, we follow Zhang et al. (2016) to use microblog hashtags as gold-

[3]Weibo is short for Sina Weibo, the biggest microblog platform in China and shares the similar market penetration as Twitter (Rapoza, 2011). Similar to Twitter, it has a length limitation of 140 Chinese characters.

[4]http://trec.nist.gov/data/tweets/

[5]http://developer.twitter.com/en/docs/tweets/search/api-reference/get-saved_searches-show-id

[6]http://open.weibo.com/wiki/Trends/hourly

[7]http://www.open.weibo.com/wiki/2/search/topics

	Single-layer Taggers				Joint-layer Taggers			
	RNN	GRU	LSTM	BiLSTM	RNN	GRU	LSTM	BiLSTM
No Encoder	44.9±1.4	53.9±4.7	54.9±3.8	60.8±3.6	51.0±3.3	56.1±3.4	55.1±2.6	62.5±0.9
Context Encoder								
Avg Emb	50.4±0.9	58.8±2.9	56.0±0.7	62.2±3.0	51.5±1.7	59.0±3.5	58.7±3.7	64.5±0.4
RNN	46.4±1.6	56.4±1.9	55.6±2.5	59.0±2.4	52.2±2.8	54.4±2.8	58.3±1.8	63.7±1.3
GRU	50.3±0.8	53.7±1.0	58.0±0.9	56.8±2.3	50.8±4.8	52.3±3.8	57.0±2.1	63.0±1.3
LSTM	51.6±2.0	56.4±1.4	57.9±2.3	**64.0**±3.1	50.8±3.1	57.9±2.3	58.3±4.0	64.2±0.6
BiLSTM	49.2±1.7	58.3±1.1	56.0±2.0	62.6±3.2	52.7±3.4	56.8±1.0	56.5±3.6	63.7±2.3
Att (LSTM)	48.7±1.7	58.1±1.7	58.1±3.1	**64.0**±1.8	51.7±4.8	57.4±2.3	58.0±2.4	63.8±1.5
Att (BiLSTM)	51.7±1.4	58.3±1.5	57.0±3.6	62.8±2.5	52.3±4.3	58.0±1.8	59.0±3.9	64.2±3.4
MemNN	**53.6**±0.3	**59.4**±3.1	**59.5**±4.1	62.4±4.8	**53.7**±3.5	**59.4**±2.1	**62.3**±3.3	**65.5**±1.6

Table 4: Comparisons of the average F1 scores (%) and their standard deviations measured on Twitter over the results of models with 5 sets of parameters for random initialization. The left half reports results of single-layer taggers; The right half reports results of joint-layer taggers. Each column: results of the same tagger with different encoders. Each row: results of different taggers with the same encoder. No Encoder: taggers without encoding context. Abbreviations for context encoders: Avg Emb – averaged embedding; Att (LSTM) – attention on LSTM; Att (BiLSTM) – attention on BiLSTM; MemNN – memory networks.

standard keyphrases[8] and filtered all microblog posts by two rules: first, there is only one hashtag per post; second, the hashtag is inside a post, i.e., containing neither the first nor the last word of a post. Then, we removed all the "#" symbols in hashtags before keyphrase extraction. For both Twitter and Weibo dataset, we randomly sample 80% for training, 10% for development, and the rest 10% for test. Table 3 reports the statistics of the two datasets. The dataset released by Zhang et al. (2016) is not used because it does not contain conversation information.

We preprocessed Twitter dataset with Twitter NLP tool[9] (Gimpel et al., 2011; Owoputi et al., 2013) for tokenization. For Weibo dataset, we used NLPIR tool[10] (Zhang et al., 2003) for Chinese word segmentation. In particular, Weibo conversations have an relatively wide range (from 3 to 8,846 words), e.g., one conversation could contain up to 447 messages. If use the maximum length of all conversations as the input length for encoders, padding the inputs will lead to a sparse matrix. Therefore, for long conversations (with more than 10 messages), we use KLSum (Haghighi and Vanderwende, 2009) to produce summaries with a length of 10 messages and then encode the produced summaries. In contrast, we do not summarize Twitter conversations because their length range is much narrower (from 4 to 1,035 words).

[8] Zhang et al. (2016) proves that 90% of the hashtag-annotated keyphrases match human annotations.

[9] http://www.cs.cmu.edu/~ark/TweetNLP/

[10] https://github.com/NLPIR-team/NLPIR

3.2 Model Settings

For keyphrase taggers based on RNN, GRU, and LSTM, we follow Zhang et al. (2016) and set their state size to 300. For the BiLSTM tagger, which has two directions, we set the state size for each direction to 150. The joint-layer taggers employ the same hyper-parameters according to Zhang et al. (2016). The state size of context encoders shares the same settings with keyphrase taggers. In training, the entire keyphrase extraction framework uses cross-entropy loss and RMSprop optimizer (Graves, 2013) for parameter updating.

We initialize input feature map I for target post and I_c for conversation context by embeddings pre-trained on large-scale external microblog collections from Twitter and Weibo. Twitter embeddings are trained on 99M tweets with 27B tokens and 4.6M words in the vocabulary. Weibo embeddings are trained on 467M Weibo messages with 1.7B words and 2.5M words in the vocabulary.

In comparison, we employ neural taggers without encoding conversation context, which are based on RNN, GRU, LSTM, and BiLSTM. We also compare our models with the state-of-the-art joint-layer RNN (Zhang et al., 2016) and its GRU, LSTM, and BiLSTM variations.

To further illustrate the effectiveness of leveraging conversation context for keyphrase extraction, we also evaluate some ranking-based models, namely, TF-IDF (Salton and Buckley, 1988), TextRank (Mihalcea and Tarau, 2004), and KEA implemented by KEA-3.0[11] (Witten et al., 1999).

[11] www.nzdl.org/Kea/Download/KEA-3.0.zip

	Single-layer Taggers				Joint-layer Taggers			
	RNN	GRU	LSTM	BiLSTM	RNN	GRU	LSTM	BiLSTM
No encoder	58.8±1.4	66.2±0.8	67.3±1.6	74.8±0.7	55.5±0.5	64.1±0.7	64.9±0.6	76.8±0.5
Context Encoder								
Avg Emb	**63.3**±0.9	68.2±0.7	69.4±0.4	76.6±0.9	61.1±1.2	69.7±1.3	69.3±0.7	79.8±0.6
RNN	58.2±1.6	64.9±0.6	65.3±0.8	73.1±0.1	60.9±0.5	67.1±0.6	66.7±0.5	71.2±0.7
GRU	56.5±0.8	67.0±0.6	67.4±1.1	73.8±0.7	58.4±1.1	65.5±0.8	67.1±0.4	76.2±0.7
LSTM	59.4±2.0	67.6±0.8	68.1±0.5	75.5±0.2	61.1±1.9	68.4±1.1	69.5±0.7	78.1±1.0
BiLSTM	60.8±1.7	68.6±1.0	68.4±0.7	75.9±0.7	61.6±1.8	69.3±1.0	69.6±0.3	78.2±0.8
Att (LSTM)	62.4±1.8	67.6±1.1	69.0±0.7	75.8±1.2	**63.1**±1.3	70.2±0.8	70.8±1.3	79.3±0.5
Att (BiLSTM)	59.6±1.4	68.6±0.6	**70.4**±1.0	76.5±0.8	61.5±2.2	**70.5**±0.6	**71.0**±0.5	**80.5**±1.7
MemNN	61.1±0.4	**69.3**±0.5	69.9±0.7	**79.1**±1.1	61.8±1.4	68.7±0.9	69.3±0.4	79.6±1.4

Table 5: Comparisons of F1 scores on Weibo. The abbreviations are defined the same as those in Table 4.

We design two experiment settings when running these models: 1) each target post is treated as a document; 2) each conversation (containing the target post) is treated as a document. We select the top N words for each target post by their ranked-orders and the threshold N is tuned on the development set. As a result, N ranges from 2 to 7 for various methods. Particularly, since TF-IDF and TextRank extract keywords instead of keyphrases, we aggregate the selected keywords according to Bellaachia and Al-Dhelaan (2012).

4 Experimental Results

Section 4.1 to 4.5 present quantitative and qualitative analysis of our neural keyprhase extraction models. Section 4.6 reports the performance of ranking-based models where we test the general applicability of incorporating conversation context to non-neural keyphase extraction methods.

4.1 Overall Comparisons

Table 4 and Table 5 report F1 scores on Twitter and Weibo, respectively.[12] We have the following observations.

Conversation context is useful for keyphrase extraction. By combining the encoded context in conversations, the F1 scores of all taggers are better than their basic versions without context encoders. It confirms that content in conversations helps in indicating keyphrases in target posts.

Selecting the correct context encoder is important. Encoding context simply by RNN or GRU yields poor results. The reason for RNN is that it suffers from gradient vanishing problem when encoding long conversations (conversions in our two datasets have over 45 words on average). The reason for GRU is that its forget gates may be not well trained to process important content when the training set is small.

The results of AvgEmb are the worst on Twitter while competitive to other encoders on Weibo. The performance of AvgEmb is competitive to other complex context encoders on Weibo. The reason may be that incorrect word orders generally do not affect the understanding in Chinese, where word order misuse is prevalent in Chinese Weibo messages. As a result, encoding word orders, as is done by the encoders except AvgEmb, might bring noise to keyphrase extraction on Weibo dataset. In contrast, AvgEmb is the worst encoder on Twitter dataset, as word order is crucial in English.

Identifying salient content in context is important. Four types of context encoders have different behaviors. Avg Emb considers all words in conversation context are equally important. RNN-variant context encoders, i.e., RNN, GRU, LSTM, and BiLSTM, additionally explore the relations between succeeded words without distinguishing salient and non-salient words. Attention (Att (LSTM) and Att (BiLSTM)) and MemNN can recognize critical content in conversations, which would indicate keyphrases in target posts. Therefore, our keyphrase extraction framework with attention or MemNN encoder has generally better F1 scores than those with other encoders.

MemNN can effectively capture salient content in context. On Twitter dataset, MemNN achieves the best F1 scores when combining with various keyphrase taggers except for single-layer GRU and BiLSTM. On Weibo dataset, although MemNN does not always outperform other encoders, its performance is close to the best ones.

[12] We also tried BiRNN and BiGRU as keyphrase taggers and as context encoders. They are outperformed by BiLSTM. We don't report these results due to the space limitation.

	SL BiLSTM		JL BiLSTM	
	Twitter	Weibo	Twitter	Weibo
No encoder	60.8	74.8	62.5	76.8
Avg Emb	61.0	75.7	63.3	79.2
RNN	58.8	72.7	63.1	71.1
GRU	56.4	73.1	62.7	76.0
LSTM	61.3	75.2	63.9	77.8
BiLSTM	62.0	75.4	62.3	78.0
Att (LSTM)	**62.6**	75.6	63.7	79.2
Att (BiLSTM)	62.0	76.5	63.9	**79.9**
MemNN	61.6	**77.4**	**65.1**	79.2

Table 6: The F1 scores of BiLSTM taggers measured on test instances without conversation context (%). SL BiLSTM and JL BiLSTM denote keyphrase tagger as single-layer and joint-layer BiLSTM, respectively. The other abbreviations are defined the same as those in Table 4.

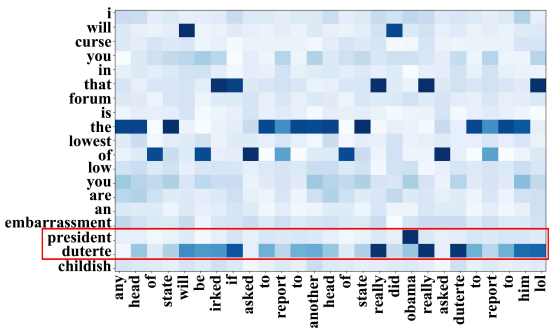


Figure 4: The heatmap of the context representation generated by MemNN (see Eq. 8). The horizontal axis refers to words in the conversation context, while the vertical axis refers to words in the target post. Darker colors indicate higher weights. The red box indicates the keyphrase to be detected.

4.2 Test without Conversation Context

Although we have shown in the previous section that conversation context is useful for training effective models for keyphrase extraction on microblog posts, it is necessary to consider that conversation context might be unavailable to some microblog posts, which do not sparking any repost or reply message. Under this circumstance, the models trained on messages with conversation context might be affected in extracting the keyphrases for messages without conversation context. To study whether conversation context is critical in testing process, we assume that the conversations are only available for training data, while all the target posts in the test set have no context to be leveraged. To this end, we apply the models trained for the experiment in Section 4.1 on the test posts without using their conversation context. In prediction, context encoders of the trained models take the target posts instead of conversation as input. Results are reported in Table 6, where models with context encoders yield better F1 scores than their counterparts without such encoders no matter providing conversation to test data or not. This observation indicates that encoding conversations in training data helps in learning effective keyphrase extraction models, which is beneficial to detect keyphrases in a microblog post with or without its conversation context. In addition, by comparing Table 6 with Table 4 and 5, we find that, for each model with context encoder, higher F1 scores are observed when conversation context is used in testing process. This observation confirms that, conversation context of target posts helps in indicating keyphrases in prediction.

4.3 Qualitative Analysis

To qualitatively analyze why MemNN encoder generally performs better in comparison, we conduct a case study on the sample instance in Table 1. Recall that the keyphrase should be "*president Duterte*". We compare the keyphrases produced by the joint-layer BiLSTM tagger with various context encoders, given in Table 7. Of all models, only the one with MemNN encoder tags correctly. Interestingly, Avg Emb does not extract any keyphrase. The reason might be that it considers each word in conversations independent and equally important. Therefore, when using this encoder, non-topic words like "*if*" and "*LOL*" may distract the keyphrase tagger in identifying the key information. Models with BiLSTM, Att (BiLSTM), and the basic model without encoder mistakenly extract the sentiment word "*childish*" since sentiment words are prominent on Twitter.

We also visualize context representation generated by MemNN for conversation context in a heatmap shown in Figure 4. It is observed that MemNN highlights different types of words for keyphrases and non-keyphrases. For keyphrases, MemNN highlights topical words such as "*Obama*". For non-keyphrases, MemNN highlights non-topic words, e.g., "*be*", "*to*". Therefore, features learned for keyphrases and non-keyphrases are different, which can thus benefit keyphrase tagger to correctly distinguish keyphrases from non-keyphrases.

4.4 Keyphrases with Various Lengths

To further evaluate our methods, we investigate them on keyphrases with various lengths. Figure 5

	Extracted keyphrase
Gold-standard	*president duterte*
No encoder	*duterte childish*
Context Encoder	
Avg Emb	*NULL*
BiLSTM	*duterte childish*
Att (BiLSTM)	*president duterte childish*
MemNN	*president duterte*

Table 7: Outputs of joint-layer BiLSTM combined with various context encoders given the example illustrated in Table1. "*NULL*": Avg Emb did not produce any keyphrase.

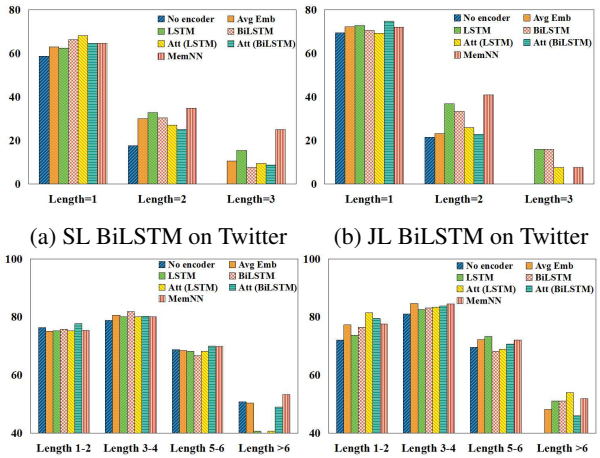
shows the histograms of F1 scores yielded by a single-layer and a joint-layer tagger on Twitter and Weibo when keyphrase lengths are different. Note that we only report the results of BiLSTM taggers because their overall F1 scores are the best according to Table 4 and Table 5.

In general, the F1 scores of all models decrease when keyphrases becomes longer, which implies that detecting longer keyphrases is harder than short ones. In comparison of different context encoders, we observe that MemNN obtained the best F1 score in detection of long keyphrases. This is because MemNN highlights salient content in conversation context by jointly considering its similarities with keyphrases in target posts. When the keyphrases become longer, there are more words in context highlighted, which hence helps keyphrase tagger. For short keyphrases, MemNN is still competitive with other context encoders. The observation suggests that MemNN is robust in detecting various length of keyphrases.

4.5 Error Analysis

In this section, we briefly discuss the errors found in our experiments. It is observed that one major incorrect prediction is additionally extracted neighboring words surrounding a gold-standard keyphrase. For example, in the tweet "*Hillary Clinton accepted gifts from UAE, Saudi Arabia, Oman and others while SOS. CROOKED Podesta Emails 29 ...*", in addition to the gold-standard "*Podesta Emails 29*", our models also extract out "*CROOKED*". In general, these additionally extracted words are mostly modifiers of keyphrases. External features for identifying modifiers can be used to filter these auxiliary parts of a keyphrase.

Another main error comes from the words that are not keyphrases in target posts but reflect the topics in conversations. For example, joint-layer



(a) SL BiLSTM on Twitter (b) JL BiLSTM on Twitter

(c) SL BiLSTM on Weibo (d) JL BiLSTM on Weibo

Figure 5: Histograms of F1 scores on extracting keyphrases with various lengths. SL BiLSTM: tagger based on single-layer BiLSTM. JL BiLSTM: tagger based on joint-layer BiLSTM. Length: count of words in keyphrases. For each length range, histograms from left to right show the results of No encoder, Avg Emb, LSTM, BiLSTM, Att (LSTM), ATT (BiLSTM), and MemNN.

BiLSTM tagger with MemNN encoder mistakenly extracts "*Hillary*" as a keyphrase for "*DOUBLE STANDARD: Obama DOJ Prosecuted Others For Leaking FAR LESS Than Hillary Espionage URL*" whose keyphrase should be "*Espionage*". Because the corresponding conversation of this post is centered around "*Hillary*" instead of "*Espionage*", such information is captured by the context encoder, which leads to incorrect keyphrase prediction. However, this type of error points out the potential of extending our framework to extracting keyphrases from conversations instead of a post, which would be beneficial to generating summary-worthy content for conversations (Fernández et al., 2008; Loza et al., 2014).

4.6 Ranking-based Models

Table 8 reports the results of ranking models on Twitter and Weibo. We have the following observations. First, tagging-based models perform much better than ranking-based ones in keyphrase extraction. Comparing the results in Table 8 with that in Table 4 and Table 5, all neural taggers outperform non-neural ranking-based models by a large margin. This fact, again, confirms that keyphrase extraction is a challenging task on short microblog messages. Compared to ranking-based models, neural tagging models have the ability

	Twitter			Weibo		
	Pre	Rec	F1	Pre	Rec	F1
w/o context						
TF-IDF	6.3	**48.8**	11.1	1.9	7.3	3.0
TextRank	6.6	18.8	9.7	1.0	8.6	1.7
KEA	3.5	0.8	1.3	0.1	0.2	0.1
w/ context						
TF-IDF	7.9	45.6	13.4	2.1	8.3	3.4
TextRank	4.8	20.8	7.8	1.0	9.5	1.8
KEA	**15.4**	12.9	**14.0**	**2.2**	**12.3**	**3.7**

Table 8: Precision, recall, and F1 scores of ranking-based baselines (%). w/o context: each target post is treated as a document; w/ context: each conversation and its corresponding target post is treated as a document.

to capture indicative features. Second, conversation context improves ranking-based models by a large margin. Simply by aggregating conversations to a pseudo-document, the F1 scores of TF-IDF, TextRank, and KEA are generally better than their counterparts that are only performed on target posts. For TF-IDF and TextRank, which are unsupervised, context remarkably improves recall by enriching more topic-related words. While for supervised method KEA, context improves both precision and recall, because supervision helps in identifying good features from conversations.

5 Related Work

Previous work on extracting keyphrases mainly focuses on formal texts like news reports (Wan and Xiao, 2008) and scientific articles (Nguyen and Kan, 2007). Existing keyphrase extraction models can be categorized as ranking-based models and tagging-based models. Ranking-based methods include models based on graph ranking (Mihalcea and Tarau, 2004; Wan and Xiao, 2008), text clustering (Liu et al., 2009), TF-IDF (Jones, 2004; Zhang et al., 2007; Lee and Kim, 2008; Kireyev, 2009; Wu and Giles, 2013), etc. The empirical study provided by Hasan and Ng (2010) shows that TF-IDF has robust performance and can serve as a strong baseline. Tagging models focus on using manually-crafted features for binary classifiers to predict keyphrases (Frank et al., 1999; Tang et al., 2004; Medelyan and Witten, 2006). Our models are in the line of tagging approaches, and provide an alternative choice that incorporates additionally knowledge from conversations.

Recently, keyphrase extraction methods have been extended to social media texts (Zhao et al., 2011; Bellaachia and Al-Dhelaan, 2012; Marujo et al., 2015; Zhang et al., 2016). These work suffers from the data sparsity issue because social media texts are normally short. Also, they only use internal information in the input text and ignore external knowledge in conversation context. Thus our work provides an improved approach that compensates their limitations.

6 Conclusion

This work presents a keyphrase extraction framework for microblog posts with considering conversation context to alleviate the data sparsity in short and colloquial messages. The posts to be tagged are enriched by conversation context through four types of encoders based on averaged embedding, RNN, attention, and memory networks, which are effective in capturing salient content in conversations that is indicative for keyphrase identification. Experimental results on Twitter and Weibo dataset have shown that by effectively encoding conversation context, our proposed models outperform existing approaches by a large margin. Qualitative analysis confirm that our context encoders capture critical content in conversations.

Acknowledgments

We thank Shuming Shi, Haisong Zhang, Jialong Han, and three anonymous reviewers for their valuable suggestions on different aspects of this work. Chengzhi Zhang was supported by National Social Science Fund of China 17ZDA291.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv pre-print* arXiv/1409.0473.

Abdelghani Bellaachia and Mohammed Al-Dhelaan. 2012. NE-Rank: A novel graph-based keyphrase extraction in Twitter. In *Proceedings of the IEEE/WIC/ACM International Conferences on Web Intelligence, WI*. pages 372–379.

Yi Chang, Xuanhui Wang, Qiaozhu Mei, and Yan Liu. 2013. Towards Twitter context summarization with user influence models. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining, WSDM*. pages 527–536.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv pre-print* arXiv/1406.1078.

Jaeho Choi, W. Bruce Croft, and Jinyoung Kim. 2012. Quality models for microblog retrieval. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM*. pages 1834–1838.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv pre-print* arXiv/1412.3555.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of the Twenty-Fifth International Conference Machine Learning, ICML*. pages 160–167.

Raquel Fernández, Matthew Frampton, John Dowding, Anish Adukuzhiyil, Patrick Ehlen, and Stanley Peters. 2008. Identifying relevant phrases to summarize decisions in spoken meetings. In *Proceedings of 9th Annual Conference of the International Speech Communication Association, INTERSPEECH*. pages 78–81.

Eibe Frank, Gordon W. Paynter, Ian H. Witten, Carl Gutwin, and Craig G. Nevill-manning. 1999. Domain-specific keyphrase extraction. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI*. pages 668–673.

Debanjan Ghosh, Alexander Richard Fabbri, and Smaranda Muresan. 2017. The role of conversation context for sarcasm detection in online interactions. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*. pages 186–196.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for Twitter: annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. pages 42–47.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv pre-print* arXiv/1308.0850.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks* 18(5-6):602–610.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring content models for multi-document summarization. In *Proceedings of the Conference of the North American Chapter of the Association of Computational Linguistics, NAACL*. pages 362–370.

Kazi Saidul Hasan and Vincent Ng. 2010. Conundrums in unsupervised keyphrase extraction: making sense of the state-of-the-Art. In *Proceedings of the 23rd International Conference on Computational Linguistics, COLING*. pages 365–373.

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

Haoran Huang, Qi Zhang, Yeyun Gong, and Xuanjing Huang. 2016. Hashtag recommendation using end-to-end memory networks with hierarchical attention. In *Proceedings of the 26th International Conference on Computational Linguistics, COLING*. pages 943–952.

Karen Spärck Jones. 2004. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation* 60(5):493–502.

Kirill Kireyev. 2009. Semantic-based estimation of term informativeness. In *Proceedings of the Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, NAACL*. pages 530–538.

Sungjick Lee and Han-Joon Kim. 2008. News keyword extraction for topic tracking. In *Proceedings of the Fourth International Conference on Networked Computing and Advanced Information Management, NCM*. pages 554–559.

Jing Li, Wei Gao, Zhongyu Wei, Baolin Peng, and Kam-Fai Wong. 2015. Using content-level structures for summarizing microblog repost trees. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP*. pages 2168–2178.

Jing Li, Ming Liao, Wei Gao, Yulan He, and Kam-Fai Wong. 2016. Topic extraction from microblog posts using conversation structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL*. pages 2114–2123.

Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. 2009. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, EMNLP, August, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 257–266.

Vanessa Loza, Shibamouli Lahiri, Rada Mihalcea, and Po-Hsiang Lai. 2014. Building a dataset for summarization and keyword extraction from emails. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC*. pages 2441–2446.

Luís Marujo, Wang Ling, Isabel Trancoso, Chris Dyer, Alan W. Black, Anatole Gershman, David Martins de Matos, João Paulo da Silva Neto, and Jaime G. Carbonell. 2015. Automatic keyword extraction on Twitter. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL*. pages 637–643.

Olena Medelyan and Ian H. Witten. 2006. Thesaurus based automatic keyphrase indexing. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries, JCDL*. pages 296–297.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing , EMNLP, A meeting of SIGDAT, a Special Interest Group of the ACL, held in conjunction with ACL*. pages 404–411.

Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In *Proceedings of the Asian Digital Libraries. Looking Back 10 Years and Forging New Frontiers, 10th International Conference on Asian Digital Libraries, ICADL*. pages 317–326.

Yuanping Nie, Yi Han, Jiuming Huang, Bo Jiao, and Aiping Li. 2017. Attention-based encoder-decoder model for answer selection in question answering. *Frontiers of IT & EE* 18(4):535–544.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of the Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, NAACL*. pages 380–390.

Barak A. Pearlmutter. 1989. Learning state space trajectories in recurrent neural networks. *Neural Computation* 1(2):263–269.

Colin Raffel and Daniel P. W. Ellis. 2015. Feed-forward networks with attention Can solve some long-term memory problems. *arXiv pre-print* arXiv/1512.08756.

Kenneth Rapoza. 2011. China's Weibos vs US's Twitter: And the winner is? *Forbes* .

Yafeng Ren, Yue Zhang, Meishan Zhang, and Donghong Ji. 2016. Context-sensitive Twitter sentiment classification using neural network. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. pages 215–221.

Ricardo Ribeiro, Anatole Gershman, David Martins de Matos, João Paulo Neto, and Jaime G. Carbonell. 2017. Event-based summarization using a centrality-as-relevance model. *Knowledge & Information Systems* 50(3):945–968.

Gerard Salton and Chris Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing & Management* 24(5):513–523.

SØren Kaae SØnderby, Casper Kaae SØnderby, Henrik Nielsen, and Ole Winther. 2015. Convolutional LSTM networks for subcellular localization of proteins. In *Proceedings of the Second International Conference on Algorithms for Computational Biology*. pages 68–80.

Jie Tang, Juan-Zi Li, Kehong Wang, and Yue-Ru Cai. 2004. Loss minimization based keyword distillation. In *Proceedings of the Advanced Web Technologies and Applications, 6th Asia-Pacific Web Conference, APWeb*. pages 572–577.

Peter D. Turney. 2000. Learning algorithms for keyphrase extraction. *Information Retrieval* 2(4):303–336.

Xiaojun Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI*. pages 855–860.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *Proceedings of the International Conference on Learning Representations, ICLR*.

Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin, and Craig G. Nevill-Manning. 1999. KEA: Practical automatic keyphrase extraction. In *Proceedings of the Fourth ACM conference on Digital Libraries*. pages 254–255.

Zhaohui Wu and C. Lee Giles. 2013. Measuring term informativeness in context. In *Proceedings of the Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, NAACL*. pages 259–269.

Hua-Ping Zhang, Hong-Kui Yu, De-Yi Xiong, and Qun Liu. 2003. HHMM-based chinese lexical analyzer ICTCLAS. In *Proceedings of the Second SIGHAN Workshop on Chinese Language Processing*. pages 184–187.

Qi Zhang, Yang Wang, Yeyun Gong, and Xuanjing Huang. 2016. Keyphrase extraction using deep recurrent neural networks on Twitter. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP*. pages 836–845.

Yongzheng Zhang, Evangelos E. Milios, and A. Nur Zincir-Heywood. 2007. A comparative study on key phrase extraction methods in automatic web site summarization. *Journal of Digital Information Management* 5(5):323–332.

Wayne Xin Zhao, Jing Jiang, Jing He, Yang Song, Palakorn Achananuparp, Ee-Peng Lim, and Xiaoming Li. 2011. Topical keyphrase extraction from Twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. pages 379–388.

Estimating Summary Quality with Pairwise Preferences

Markus Zopf
Research Training Group AIPHES / Knowledge Engineering Group
Department of Computer Science, Technische Universität Darmstadt
Hochschulstraße 10, 64289 Darmstadt, Germany
zopf@aiphes.tu-darmstadt.de

Abstract

Automatic evaluation systems in the field of automatic summarization have been relying on the availability of gold standard summaries for over ten years. Gold standard summaries are expensive to obtain and often require the availability of domain experts to achieve high quality. In this paper, we propose an alternative evaluation approach based on pairwise preferences of sentences. In comparison to gold standard summaries, they are simpler and cheaper to obtain. In our experiments, we show that humans are able to provide useful feedback in the form of pairwise preferences. The new framework performs better than the three most popular versions of ROUGE with less expensive human input. We also show that our framework can reuse already available evaluation data and achieve even better results.

1 Introduction

Due to the huge amount of information contained in texts, the task of automatic text summarization (Mani, 2001; Nenkova and McKeown, 2011) is a pressing challenge nowadays and will become even more important in the future. Building summarization systems is, however, not the only challenge in this field. Evaluation of automatically generated summaries is also an active field of research.

Ideally, we would like to ask humans for their opinion about the quality of automatically generated summaries in an extrinsic evaluation (Halteren and Teufel, 2003). Since summaries are generated for humans, they should also be evaluated directly by humans. Unfortunately, manual evaluation cannot be performed at a large scale because of the huge effort which is necessary for evaluation. (Lin, 2004) reported that 3,000 hours of human effort would be required for a simple evaluation of the summaries for the Document Understanding Conference (DUC) 2003, a popular summarization shared task series. This motivates research of automatic evaluation methods for automatic summarization.

ROUGE (Lin, 2004), the current method of choice for evaluating automated text summarization, relies on the availability of gold standard summaries. The gold standard summaries are used to define the optimal output of a summarization system. Writing high-quality summaries, however, requires the availability of expert writers and takes a lot of effort. (Dang, 2005) reported that creating the reference summaries for the DUC 2005 shared task was a difficult endeavor with an effort of five hours to produce each reference summary. Since ROUGE needs at least four reference summaries to become reasonably reliable, the effort sums up to at least 20 hours of annotation effort per topic. For this reason, gold standard summaries are only available for a few, rather small datasets. Also the more accurate (but also even more expensive) Pyramid method (Nenkova and Passonneau, 2004) requires expensive gold standard summaries.

Lack of larger and diverse evaluation corpora limits research in automatic summarization. Furthermore, currently available automatic evaluation methods are viewed with skepticism (Rankel et al., 2013). Proper evaluation is, however, an indispensable ingredient for good research. Computing the similarity between two summaries as in ROUGE is a very difficult task. This seems to be obvious since estimating the similarity between sentences and even words is still an active field of research.

In this work, we present an alternative evaluation framework which does not use gold standard summaries to estimate the quality of summaries. Instead of comparing automatically generated summaries with gold standard summaries,

Proceedings of NAACL-HLT 2018, pages 1687–1696
New Orleans, Louisiana, June 1 - 6, 2018.

our model is trained with simple and inexpensive pairwise preferences (Thurstone, 1927; Fürnkranz and Hüllermeier, 2010) of sentences. To this end, we provide pairs of sentences from the input document of a summarization task to human annotators and ask which of the two sentences contains more important information. We use here the idea of intrinsic information importance (Hong and Nenkova, 2014; Zopf et al., 2016) which describes that information can be intrinsically important. For example, the information *"Donald Trump won the U.S. presidential election"* is intrinsically important. It is likely that it should also be contained in the generated summary if this information is contained in an input document.

After collecting few preferences, our model uses the preferences to generate a ranking of all sentences according to information importance. Summaries which contain sentences similar the upper part of the ranking are then considered to be better than summaries which contain unimportant sentences from the lower part of the ranking.

Pairwise preferences are an appealing form of annotation, since they are much easier to generate than producing complex gold standard summaries. Not only collecting the annotations is easier, but also using the collected annotations is much simpler. The presented model does not have to solve the difficult task of estimating the similarity between generated and gold standard summaries. Instead, the model uses the ranking to estimate the summary quality.

Figure 1 provides an illustration of the traditional evaluation and our new model. On the left, the input documents are illustrated which should be summarized. In the upper part gold standard summaries are generated by humans and used to estimate the quality of an automatically generated summary. In the lower part, we collect pairwise preferences of sentences and use the preferences for evaluation.

An evaluation on topics from two standard datasets, looking at predicting the relative ratings of automatically generated summaries, shows that our new evaluation model is as good as or better than existing methods, at a much lower annotation cost.

2 Related Work

In this section, we will recapitulate previous work in automated text summarization evaluation, focusing on three important approaches, namely model-free evaluation, ROUGE, and Pyramid. The evaluation methods are ordered according to their annotation requirements from none (model-free evaluation) to high (Pyramid). In addition to the most prominent methods described below, several evaluation models were developed in the Automatically Evaluating Summaries Of Peers (AESOP) shared tasks. The systems in this shared task also considered reference summaries as additional information to evaluate a reference summary and are therefore as expensive as ROUGE in terms of required human annotation. Similarly, Giannakopoulos and Karkaletsis (2013) use machine learning to learn a linear combination of n-gram methods to evaluate summaries. Mackie et al. (2014), Giannakopoulos (2013), and Cohan and Goharian (2016) investigate evaluation for microblog, multilingual, and scientific summarization, respectively. Our evaluation, on contrary, uses newswire datasets since this is the most prominent application domain for automatic summarization. Furthermore, we focus on evaluating the information content of summaries and do not evaluate linguistic quality. This is, for example, captured by Pitler et al. (2004).

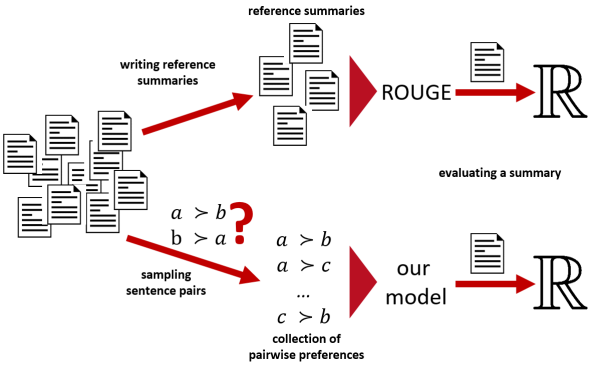


Figure 1: Illustration of traditional evaluation models based on reference summaries (top) and the new model (bottom) which is based on pairwise preferences.

2.1 Model-free Evaluation

Model-free evaluation methods Jensen-Shannon divergence (Louis and Nenkova, 2013) do not require human input such as gold standard summaries and can therefore be applied without additional cost. The quality of model-free evaluation methods is however limited, which is validated in our experiments (see Section 5).

2.2 ROUGE

ROUGE (Lin, 2004) was first used in the Document Understanding Conference (DUC) (Over et al., 2007) and is nowadays the method of choice for automatic evaluation in text summarization. Many popular summarization systems were evaluated with ROUGE (Erkan and Radev, 2004; Mihalcea and Tarau, 2004; Gillick et al., 2009; Lin and Bilmes, 2011). It is inspired from the BLEU evaluation method (Papineni et al., 2002) and is based on measuring lexical n-gram overlap of (stemmed) tokens between generated and gold standard summaries. Researchers usually report the n-gram recall of a summary to evaluate the quality of a summary. The quality of ROUGE is often criticized in the research community. Sjöbergh (2007), for example, shows nicely how the ROUGE recall scoring can be fooled easily. A simple greedy language model based on the source documents extracts frequent bi-grams which are likely to occur in the reference summaries. The generated texts are merely lists of bi-grams and not meaningful sentences which cannot be considered to be summaries. However, they achieve superhuman ROUGE recall scores. In the TAC 2008 shared task (Dang and Owczarzak, 2008), both ROUGE-2 and ROUGE-SU4 score automatic systems higher than human summaries, which would lead to the conclusion that these systems are able to produce better summaries than humans. Furthermore, studies show that the correlation between ROUGE scores and human judgments may not be significant in non-newswire genres and other summary types (Liu and Liu, 2008). ROUGE also has many parameters (Graham, 2015), which makes reproduction and comparison of results problematic. Last but not least, ROUGE computes text similarity only based on simple string matching. Expressing the same information with different words is not rewarded by ROUGE. In addition to Graham (2015), Owczarzak et al. (2012) and Rankel et al. (2013) analyze ROUGE in more detail.

Agreement with human judgments (Owczarzak et al., 2012) can be used instead of Pearson's correlation to validate an automatic evaluation model. Measuring agreement allows to obtain a better understanding of the performance of an evaluation model compared to the Pearson correlation. We will also use agreement similarly to Owczarzak et al. (2012) in our experiments.

2.3 Pyramid

The Pyramid method (Nenkova et al., 2007) (similar to (Teufel and Van Halteren, 2004)) was used in the Text Analysis Conference (TAC) (Dang and Owczarzak, 2008) and goes beyond lexical comparisons. It is based on *Summarization Content Units* (SCUs, later also called Summary Content Units). An SCU is a set of lexical expressions with same meaning (e.g. {"2 people passed away", "two persons died"}). After generating the gold standard summaries, SCUs are extracted from these summaries and are weighted by their occurrence frequency in an additional annotation step. Furthermore, every generated summary has to be annotated individually with SCUs before the Pyramid method can be applied. (Nenkova and Passonneau, 2004) have already reported that a large-scale application of the Pyramid method is infeasible. (Over et al., 2007) report a huge effort for the annotation process in the DUC challenges. This additional annotation effort is unattractive for researchers, who prefer automatic methods such as ROUGE. This is validated by the few applications of the Pyramid method until today. The need for more human annotation also introduces an additional source for annotation mistakes. Inspecting the annotations in the TAC 2008 dataset in detail reveals that this is not only a theoretical issue but has practical implications.[1] PEAK (Yang et al., 2016) is an attempt to automate the Pyramid evaluation (similar to (Passonneau et al., 2013)). PEAK also requires reference summaries and is therefore as expensive as ROUGE.

2.4 Qualitative Feedback in Other NLP Tasks

Simple and inexpensive qualitative human feedback has already been used in the field of machine translation (Callison-Burch, 2009; Callison-Burch et al., 2012; Guzmán et al., 2015). (Snow et al., 2008) showed that in a wide variate of NLP tasks, cheap non-expert labels can replace expensive expert annotations. In comparison to our work, we are not asking non-experts to perform the same task as expert annotators (namely writing references summaries) but replace the complex task with a simpler tasks (providing qualitative feedback in form of pairwise preferences).

[1] We found several issues such as not annotating parent SCUs, missing SCUs in sentences, and different annotations for equal sentences.

3 Problem Definition

First, we define T to be the set of all possible texts which can be considered to be summaries. For a given set of source documents D, we define a binary relation $>_D \subset T \times T$ with the intuition that $\mathbf{a} >_D \mathbf{b}$ holds for two texts $\mathbf{a}, \mathbf{b} \in T$ if and only if $\mathbf{a}$ is considered to be the better summary of document collection D than $\mathbf{b}$. Whenever the context is clear, we will omit D and write $\mathbf{a} > \mathbf{b}$ for short. The relation $>$ induces a strict total order (given that ties are not allowed) over T. A text which ranks high according to $>_D$ is a good summary of the document collection D.

How good summaries are is annotated in summarization corpora only for a very small subset of assessed texts $T^+ \subset T$ by human annotators. We use the relation $>^*$ to express in which order summaries are ranked by humans in summarization corpora for each document set D (also called topic or cluster). The relation $>^*$ therefore models the human judgments. For not assessed texts $T^- = T \setminus T^+$ the human judgment is unknown.

The quality of an evaluation method E can be assessed by measuring the agreement with the human judgments. Evaluation models define (implicitly) a ranking $>^E$ by assigning scores to summaries or predicting the ranking directly. Calculating the agreement of the ranking $>^E$ with the human ranking defined by $>^*$ provides a scores which can be used to assess the performance of evaluation models. Measuring the agreement between two relations (which are sets) can be easily done by computing the intersection of both sets:[2]

$$\text{Agreement}(>^*, >^E) = \frac{|>^* \cap >^E|}{|>^*|} \quad (1)$$

This evaluation of evaluation models is similar to the definition of *Agreement* and *Contradiction* in Owczarzak et al. (2012): *"Agreements occur when the two evaluation metrics make the same distinction between System A and System B (...). Contradictions occur when both metrics find a (...) difference between A and B, but in opposite directions."* A perfect evaluation model, which predicts the preference for all pairs of summaries correctly, will have an agreement of 1 whereas a random evaluation model, which always predicts the preference randomly, will have an expected value of 0.5 according to this measure.

[2]We require that an evaluation metric has to make a decision for two summaries if the two summaries are different according to the human judgment. Formally: $\mathbf{a} >^* \mathbf{b} \rightarrow \mathbf{a} >^E \mathbf{b}$ or $\mathbf{b} >^E \mathbf{a}$.

Donald Trump won the election and became president. $\succ$ *The U.S. Congress certified the results on January 6.*

Figure 2: Example of a pairwise preference annotation of two sentences. The first sentence is preferred over the second sentence because the first sentence contains more important information given that the information is not already known.

We prefer to use the agreement as defined in Equation 1 for evaluation since it can be much better interpreted (Owczarzak et al., 2012). Furthermore, Pearson's correlation is known to be sensitive to outliers, is only able to measure linear correlations, and requires normally distributed, interval scaled residuals (Anscombe, 1973). These properties cannot be assumed to be given when comparing human scores and automated evaluation measures. We therefore prefer to use the agreement according to human judgments as defined in Equation 1 instead of calculating Pearson's correlation.

4 Preference-based Evaluation of Summaries

In this section, we present a novel framework which does not infer a ranking of automatically generated summaries based on gold standard summaries but based on pairwise preferences. The fundamental idea is not to rely on expensive gold standard summaries as previous work does, but to ask annotators for their preferences about sentences. Annotates label pairs of sentences with a preference label which indicates which sentence contains more important information. Figure 2 illustrates such a pairwise preference annotation. A human would likely prefer the first sentence to be included in a summary instead of the second sentence because the first sentence contains, compared to the second sentence, relatively important information. Based on the preferences, our model generates a ranking which reflects the importance of information which is contained in the sentences. Sentences with important information will be ranked high whereas sentences containing only less important information will be ranked low.

4.1 Sampling Preference Annotations

The easiest strategy to select pairs of sentences for which preferences should be annotated is to sample pairs of sentences randomly and to ask annotators to provide a preference label for each sampled pair (i.e. annotating whether $a \succ b$ or $b \succ a$ for two randomly sampled $a, b \in S^*$). The sentences are sampled from all source document of a topic and are therefore independent from the automatically generated summaries. Our model will therefore not only be able to evaluate already generated summaries but also summaries which will be generated in the future.

All preferences are stored in a matrix M. An entry of n at position M_{i_j} indicates that sentence with index i was preferred n-times over sentence with index j. To reduce the number of annotations, we apply a *smooth propagation of knowledge*. The idea is that we do not only obtain information about the sampled sentence pair but also about pairs which are similar to the sampled pair.

To estimate how much information can be transfered from one to another pair, we calculate the similarities between all sentences. As similarity measure we use the average of the well-known and simple Cosine similarity of TF-IDF vectors and Jaccard similarities. The combination allows to both rely the similarity computation on lexical similarity (Jaccard) and on important content words (Cosine). We define the set of all sentences in the source documents of a topic as S^*. Let (a, b), $a, b \in S^*$ be one annotated sentence pair and $\dot{a}, \dot{b}$ the vector of similarities between a and b and all sentences (i.e. $\dot{a}_i$ denotes the similarity between a and the i-th sentence in S^*).

We define the similarity of the pair (a_1, b_1) and the pair (a_2, b_2) as $\text{sim}(a_1, a_2) * \text{sim}(b_1, b_2)$. If a_1 is the exact same sentence as a_2 and b_1 is similar with a degree of 0.7 to b_2, we will transfer 0.7 of the information from (a_1, b_1) to the pair (a_2, b_2). Transferring information means that we generate additional preferences based on human preferences. If a_1 was preferred over b_1 by a human annotator, we will additionally generate a weighted preferences of with a weight of 0.7 between a_2 and b_2. This can be modeled by the outer product $\dot{a}_1 \otimes \dot{b}_1$ of a_1 and b_1. For each annotated pair (a, b), in which a was preferred by a human over b, we update matrix M by $M \leftarrow M + \dot{a} \otimes \dot{b}$.

4.2 Sentence Score Prediction

The proposed usage of pairwise preferences between sentences is close to the idea of generating a ranking of sports teams by playing individual matches. Instead of competitions between teams, we observe competitions between sentences. The outcome of a match between teams equals to the annotation of a pair of sentences by a human annotator. Since different people can have different opinions about the importance of information (Gambhir and Gupta, 2016), we expect that one sentence will not always be preferred by humans similarly to the situation that the better sports team does not always win against a weaker opponent. This is expressed by the winning probability between teams (or sentences).

In sports, the term *power ranking* is used to describe a ranking which does not only rank the individual teams but also assigns a score to each team, the *skill*. A well-known method to generate a power ranking is the Bradly-Terry (BT) model (Bradley and Terry, 1952). It estimates the utilities $v(a), v(b)$ of two teams (or two sentences) a and b so that the winning probability of a against b equals the score of a divided by the sum of the scores of a and b:

$$p(a \text{ is prefered over } b) = \frac{v(a)}{v(a) + v(b)} \quad (2)$$

An algorithm to find a maximum-likelihood estimator (MLE) has already been proposed in (Zermelo, 1929). To find the MLE, we iteratively perform Equation 3 for all sentences s_i until the difference between two iterations is sufficiently small.[3] $\text{wins}(s_i)$ denotes the total number of wins of s_i and $\text{duels}(s_i, s_j)$ the number of duels played between sentences s_i and s_j. This information was collected in the previous step and is stored in matrix M.

$$v(s_i) \leftarrow \text{wins}(s_i) \sum_{i \neq j} \frac{\text{duels}(s_i, s_j)}{v(s_i) + v(s_j)} \quad (3)$$

We normalize the resulting skill vector after each iteration since every multiple of the solution is also a correct solution and therefore restrict the model to converge to one particular solution.

[3] We initialize all scores $v(s_i)$ equally with $v(s_i) \leftarrow \frac{1}{|\mathbf{s}|}$.

4.2.1 Summary Score Prediction

We estimate the score of summary $\mathbf{s}$ with function $u : T \rightarrow \mathbb{R}$ as follows:

$$u(\mathbf{s}) = \sum_{i=1}^{|\mathbf{s}|} w_{s_i} \cdot v(\arg\max_{s \in S^*} \text{sim}(s, s_i)) \quad (4)$$

The utility of a summary is therefore defined as the weighted sum of the sentence utilities v. Since we do not want to restrict our model to purely extractive summaries (which would mean that all sentences contained in the automatic summary have to be exactly contained in the source documents), we estimate the score of a sentence s_i in the summary by searching for the most similar sentence s in the source documents with similarity function $\text{sim} : S \times S \rightarrow [0, 1]$. As weight of s_i, we use $\frac{|s_i|}{|\mathbf{s}|}$ where $|.|$ denotes the length of the summary and sentence measured in number of characters. The intuition of the weight is that a sentence contributes more to the overall score of a summary if it is longer. The score of a summary will decrease if a large fraction of the summary is occupied with a poor sentence. By using a similarity function instead of a hard matching, our method is able to generalize to unseen sentences.

The definition of u in Equation 4 does not consider redundancy. Including a sentence s twice would result in adding the score of s twice to the summary score. This behavior of the evaluation measure is not desirable. We therefore include a redundancy penalization which does not reward redundant information. For a summary $\mathbf{s}$, we reduce the score of sentence s by

$$v_{\text{red}}(s) = v(s) \frac{1}{|s|} \sum_{g \in s} \frac{\text{num}(g, s)}{\text{num}(g, \mathbf{s})} \quad (5)$$

where $\text{num}(g, s)$ and $\text{num}(g, \mathbf{s})$ denote the number of occurrences of the bi-gram g in s and $\mathbf{s}$, respectively. $|s|$ denotes the number of bi-grams in s.

4.3 Reusing Available Annotation Information

For already existing summarization corpora, reference summaries and/or Pyramid annotations have already been created. Instead of generating new preference annotations by asking human annotators, we can also reuse the available data to simulate annotations. To this end, we define functions w_{r} and w_{p} which estimate the score of a single sentence based on reference summaries (r) and Pyramid scores (p), respectively. We will use the scores generated by w_{r} and w_{p} to simulate annotations of sentence pairs. For two sentence a, b we can simulate a human preference annotation of $a \succ b$ if $w_{\text{r}}(a) > w_{\text{r}}(b)$ and a win of b over a otherwise (equivalent for w_{p}).

For a set of gold standard summaries R, we define $w_{\text{r}} : S \rightarrow \mathbb{R}$ simply to be the maximum similarity to the sentences in the gold standard summaries:

$$w_r(s) = \max_{t \in \mathbf{r}, \mathbf{r} \in R} (\text{sim}(s, t)) \quad (6)$$

If a very similar sentence appears in a gold standard summary, s will receive a high score. If no similar sentences are in the gold standard summaries the sentence will receive a low score.

Given that Pyramid annotations are available (as in the TAC 2009 corpus, for example), we can define the score of a sentence as the sum of the weights of the matched unique SCUs (similar to the Pyramid method). Annotations are, unfortunately, only available for all sentences in the documents in T^+ and not for sentences in S^*. We therefore search for sentence s in S^* for the most similar sentence $\hat{s}$ in the documents in T^+

$$\hat{s} = \arg\max_{t \in \mathbf{t}, \mathbf{t} \in T^+} \text{sim}(s, t) \quad (7)$$

and set the score of s to

$$w_p(s) = \sum_{scu \in \hat{s}} \text{weight}(scu) \quad (8)$$

where $scu \in t$ are all unique SCUs contained in t and $\text{weight}(scu)$ denotes the weight of an SCU as defined in (Nenkova and Passonneau, 2004). As described above, we will observe wins and losses between pairs based on the estimated scores.

5 Experiments

We provide in this section a detailed analysis of our proposed evaluation method. For the experiment, we use eight topics from two popular multi-document summarization datasets, the DUC 2004 (DUC04) and TAC 2009 (TAC09) corpora, which are freely available upon request.[4] Each topic in the datasets contains ten source documents. Each topic contains automatically generated summaries which were generated in the DUC 2004 and TAC 2009 shared tasks. All automatically

[4] http://duc.nist.gov and https://tac.nist.gov

	JS	R1	R2	R3	R4	SU4	man
DUC04	0.480	0.651	0.639	0.649	0.606	0.558	**0.673**
TAC09	0.565	0.638	0.668	0.660	0.674	0.663	**0.688**

Table 1: Agreement of preference based evaluation as defined in Equation 1 of different versions of Jensen-Shannon, ROUGE and our novel model based on manually labeled pairwise preferences.

generated summaries were evaluated by humans. Each summary was labeled with a score from 1 to 5 (DUC04) or 1 to 10 (TAC09) indicating the information content of the summary. Evaluation of grammatically, writing style, etc. is not included in the scores. An evaluation model predicts the preference for two selected summaries correctly if the model predicts the same preference according to the annotated reference scores and incorrectly otherwise. We do not consider ties in the experiments. In the following, we report the agreement as described in Equation 1 for various experiments. We use the abbreviations **JS** (Jensen-Shannon), **R1** - R4 (ROUGE-1 - ROUGE-4), **SU4** (ROUGE-SU4), and **PY** (Pyramid (Nenkova and Passonneau, 2004)) to denote the reference systems.

5.1 Manual Annotations

In the first experiment, we investigate whether humans are able to provide useful feedback in the form of pairwise preferences.

To evaluate our model, we annotated 200 randomly sampled sentence pairs for the first four topics in the DUC04 and the first four topics in the TAC09 corpus with pairwise preferences. The preferences were used (including the previously described smoothed sampling) as input for the proposed model. The results are shown in Table 1. Column **man** denotes the performance of our now model and column **Time (min)** indicates how much time was required to generate the annotations. This information is in particular important for this paper since our main aim is to develop a cheap evaluation framework. In average, our model achieves an agreement of 0.673 in DUC04 and 0.688 in TAC09. This means, that 67.3/68.8 percent of all pairs of manually rated summaries were predicted correctly. This outperforms the best versions of ROUGE in the respective corpora (SU4 with 65.1 percent in DUC04 and R2 with 66.0 percent in TAC09).

With an average annotation time per topic of 53 and 54 minutes our model needs much less annotation effort than ROUGE.

	R1	R2	R4	PY	man +ref	man +py	man +ref +py
DUC04	0.651	0.639	0.606	n/a	**0.722**	n/a	n/a
TAC09	0.638	0.668	0.674	0.715	0.682	0.707	**0.717**

Table 2: Agreement of different versions of ROUGE and Pyramid (PY) and our novel models based on human and automatically generated pairwise preferences in addition to manually labeled preferences.

5.2 Weak Supervision with Additionally Simulated Annotations

In the next experiment, we are interested whether we can simulate additional annotations based on already available reference summaries and Pyramid annotations. The automatically annotated pairs can be considered to be an additional weak supervision for the model. We simulated 200 additional annotations based on reference summaries and/or Pyramid annotations in addition to the 200 manual annotations per topic. To this end, we randomly sampled 200 additional pairs and annotated the pairs with a preference label based on reference summaries and/or Pyramid annotations. Table 2, column **man+ref** contains the results for 200 manual + 200 simulated reference summary-based annotations; column **man+py** contains the results for 200 manual + 200 simulated Pyramid score-based annotations; and column **man+ref+py** contains results for 200 manual + 200 reference summary-based + 200 Pyramid score-based annotations. The results show that we can improve the agreement with additional simulated annotations based on reference summaries in DUC04 by 5 percent points. Additional annotations increased Agreement in TAC09 by 3 percent points. This leads to the conclusion that we can use already available reference summaries in order to substitute more human preference annotations, which makes the trade-off between performance and annotations effort of our model even better.

5.3 Solely Using Simulated Annotations

Now, we investigate if simulated preferences are already sufficient to produce reasonable good results. Table 3, columns **ref** and **py** contain the results of an experiment where we sampled 1,000 simulated pairwise annotations. Without any additional annotation effort, the new model is able to

	R1	R2	R4	PY	ref	py
DUC04	0.651	0.639	0.606	n/a	**0.716**	n/a
TAC09	0.638	0.668	0.674	**0.715**	0.644	0.709

Table 3: Agreement with human judgments for reference systems and our model fed with only automatically generated preferences labels.

perform much better than ROUGE at DUC 2004. In TAC 2009, our model achieves similar performance as the best performing evaluation based on Pyramid annotations. We conclude that automatically generating pairwise preferences based on already available reference summaries is already sufficient to outperform ROUGE. Pairwise preferences generated based on the more expensive Pyramid annotations do not improve the performance.

5.4 Convergence

In the next experiment, we investigate how agreement changes with an increasing amount of annotations. Figure 3 shows how agreement improves with more annotations. We sampled n annotations (horizontal axis) randomly from the human annotations and averaged the resulting agreement scores (vertical axis) of 100 runs to obtain reliable results. We observe a continuous improvement of agreement in all four topics in the TAC 2009 dataset which indicates that sampling more annotations can further improve the performance of our system.

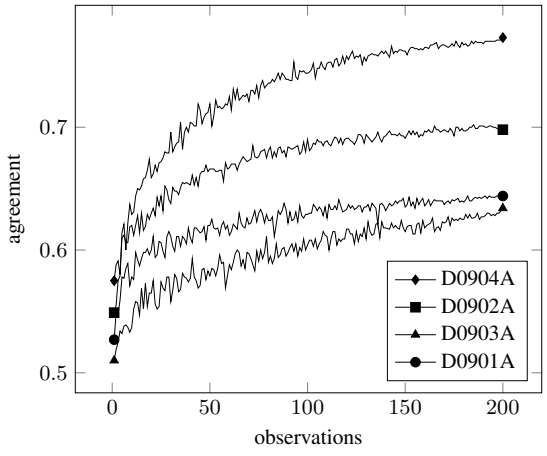


Figure 3: Agreement trajectories averaged over 100 runs per topic in the TAC 2009 corpus.

5.5 Ranking Evaluation

We now investigate the ranking generated by our model directly. Since individual sentences are annotated in the TAC 2009 corpus with SCUs, we can generate a ranking of the sentences and directly compare this ranking with the ranking generated by our model. Table 4 shows the percentage of correctly ordered sentence pairs (similar to Kendall's τ) for our model without and with smoothed sampling.

non-smoothed			smoothed		
man	pyr	ref	man	pyr	ref
0.683	0.987	0.661	0.727	0.941	0.698

Table 4: Percentage of correctly ordered sentence pairs in the TAC 2009 corpus for both a non-smoothed and a smoothed sampling.

Smoothed sampling improves the raking of the model if we use 200 manual or 200 reference summary-based preferences in the TAC 2009 corpus. Given that we can sample pairs based on Pyramid scores, the model is able to reconstruct the ranking almost perfectly if we do not use smoothed sampling. With smoothed sampling, the performance decreases in this case. The result confirms the previously observed performance at summary scoring where preferences based on Pyramid annotations performed best followed by manually generated preference annotations.

6 Conclusions & Outook

Evaluating automatically generated summaries is a challenging task and creating annotations which are required by applications such as ROUGE or Pyramid is laborious and expensive. We presented an alternative model which does not rely on reference summaries or Pyramid annotations but only on simple pairwise preferences of sentences.

We showed in our experiments that the proposed model is able to perform better than the current state-of-the-art ROUGE method with less expensive annotations and that humans are able to provide useful feedback in the form of pairwise preferences. In combination with already available references summaries and Pyramid annotations, we were able to simulate more annotations, which improved performance further.

We conclude that gold standard summaries are not the only usable human feedback which can be used for summary evaluation. Investigating other kinds of feedback such as pairwise preferences might be a promising future research direction.

In future work, we would like to investigate whether we can use crowd-sourcing platforms to

collect pairwise preferences on a large scale. Furthermore, we want to investigate whether we can reduce the number of required preferences with smarter sampling methods. Active learning methods can be used to replace the simple random sampling strategy. Additionally, the investigation of more sophisticated similarity functions can potentially improve the model's performance.

Acknowledgments

This work has been supported by the German Research Foundation (DFG) as part of the Research Training Group Adaptive Preparation of Information from Heterogeneous Sources (AIPHES) under grant No. GRK 1994/1.

References

F. J. Anscombe. 1973. Graphs in Statistical Analysis. *The American Statistician* 27(1):17–21.

Ralph Allan Bradley and Milton E. Terry. 1952. Rank Analysis of Incomplete Block Designs. *Biometrika* 39(3):324–345.

Chris Callison-Burch. 2009. Fast, cheap, and creative: evaluating translation quality using Amazon's Mechanical Turk. *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing* 1(August):286–295. https://doi.org/10.3115/1699510.1699548.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2012. Findings of the 2012 Workshop on Statistical Machine Translation. *Proceedings of the Seventh Workshop on Statistical Machine Translation* pages 10–51. https://doi.org/10.3115/1626431.1626433.

Arman Cohan and Nazli Goharian. 2016. Revisiting Summarization Evaluation for Scientific Articles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation*. pages 806–813.

Hoa Trang Dang. 2005. Overview of DUC 2005. In *Proceedings of the Document Understanding Conference*. https://doi.org/10.3115/1654679.1654689.

Hoa Trang Dang and Karolina Owczarzak. 2008. Overview of the TAC 2008 Update Summarization Task. In *Proceedings of the First Text Analysis Conference*.

Günes Erkan and Dragomir R Radev. 2004. LexRank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research* 22:457–479.

Johannes Fürnkranz and Eyke Hüllermeier, editors. 2010. *Preference Learning*. Springer.

Mahak Gambhir and Vishal Gupta. 2016. Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review* 47(1):1–66. https://doi.org/10.1007/s10462-016-9475-9.

G. Giannakopoulos and V. Karkaletsis. 2013. Summary Evaluation: Together We Stand NPowER-ed. *14th International Conference on Computational Linguistics and Intelligent Text Processing* pages 436–450.

George Giannakopoulos. 2013. Multi-document multilingual summarization and evaluation tracks in ACL 2013 MultiLing Workshop. *MultiLing 2013* page 20.

Dan Gillick, Benoit Favre, Dilek Hakkani-Tür, Berndt Bohnet, Yang Liu, and Shasha Xie. 2009. The ICSI/UTD Summarization System at TAC 2009. In *Proceedings of the Second Text Analysis Conference*.

Yvette Graham. 2015. Re-evaluating Automatic Summarization with BLEU and 192 Shades of ROUGE. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 128–137.

Francisco Guzmán, Shafiq Joty, Llu\'is Màrquez, and Preslav Nakov. 2015. Pairwise Neural Machine Translation Evaluation. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* pages 805–814. http://www.aclweb.org/anthology/P15-1078.

Hans Van Halteren and Simone Teufel. 2003. Examining the consensus between human summaries: initial experiments with factoid analysis. *Proceedings of the HLT-NAACL2003 Workshop on Text Summarization* pages 57–64. https://doi.org/10.3115/1119467.1119475.

Kai Hong and Ani Nenkova. 2014. Improving the Estimation of Word Importance for News Multi-Document Summarization. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. pages 712–721.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*. pages 25–26.

Hui Lin and Jeff Bilmes. 2011. A Class of Submodular Functions for Document Summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*. pages 510–520.

Feifan Liu and Yang Liu. 2008. Correlation between ROUGE and Human Evaluation of Extractive Meeting Summaries. In *Proceedings of the 46th Annual Meeting of the Association for Computational*

Linguistics: Human Language Technologies. pages 201–204.

Annie Louis and Ani Nenkova. 2013. Automatically Assessing Machine Summary Content Without a Gold Standard. *Computational Linguistics* 39(2):267–300.

Stuart Mackie, Richard Mccreadie, Craig Macdonald, and Iadh Ounis. 2014. On Choosing an Effective Automatic Evaluation Metric for Microblog Summarisation. *Proceedings of the 5th Information Interaction in Context Symposium* pages 115–124. https://doi.org/10.1145/2637002.2637017.

Inderjeet Mani. 2001. *Automatic Summarization*. John Benjamins Publishing Co.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into texts. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. volume 85, pages 404–411. https://doi.org/10.3115/1219044.1219064.

Ani Nenkova and Kathleen McKeown. 2011. Automatic Summarization. *Foundations and Trends in Information Retrieval* 5(3):103–233. https://doi.org/10.1561/1500000015.

Ani Nenkova and Rebecca Passonneau. 2004. Evaluating content selection in summarization: The pyramid method. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*. pages 145–152.

Ani Nenkova, Rebecca Passonneau, and Kathleen McKeown. 2007. The Pyramid Method. In *ACM Transactions on Speech and Language Processing*. volume 4, pages 1–23. https://doi.org/10.1145/1233912.1233913.

Paul Over, Hoa Dang, and Donna Harman. 2007. DUC in context. *Information Processing and Management* 43(6):1506–1520. https://doi.org/10.1016/j.ipm.2007.01.019.

Karolina Owczarzak, John M Conroy, Hoa Trang Dang, and Ani Nenkova. 2012. An assessment of the accuracy of automatic evaluation in summarization. In *Proceedings of Workshop on Evaluation Metrics and System Comparison for Automatic Summarization*. pages 1–9.

Kishore Papineni, Salim Roukos, Todd Ward, and Wj Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. July, pages 311–318. https://doi.org/10.3115/1073083.1073135.

Rebecca J. Passonneau, Emily Chen, Weiwei Guo, and Dolores Perin. 2013. Automated Pyramid Scoring of Summaries using Distributional Semantics. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. pages 143–147.

Emily Pitler, Annie Louis, and Ani Nenkova. 2004. Automatic Evaluation of Linguistic Quality in Multi-Document Summarization. *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics* pages 544–554. http://www.aclweb.org/anthology/P10-1056.

Peter A Rankel, John M Conroy, Hoa Trang Dang, and Ani Nenkova. 2013. A Decade of Automatic Content Evaluation of News Summaries: Reassessing the State of the Art. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. pages 131–136.

Jonas Sjöbergh. 2007. Older versions of the ROUGEeval summarization evaluation system were easier to fool. *Information Processing and Management* 43(6):1500–1505. https://doi.org/10.1016/j.ipm.2007.01.014.

Rion Snow, Brendan O Connor, Daniel Jurafsky, Andrew Y Ng, Dolores Labs, and Capp St. 2008. Cheap and fast - but is it good? Evaluation non-expert annotiations for natural language tasks. *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)* pages 254–263. https://doi.org/10.1.1.142.8286.

Simone Teufel and Hans Van Halteren. 2004. Evaluating Information Content by Factoid Analysis: Human annotation and stability. In *Proceedings of the 2004 Conference on Empirical Methods on Natural Language Processing*. pages 419–426.

Louis Leon Thurstone. 1927. A law of comparative judgement. *Psychological Review* 34:278–286.

Qian Yang, Rebecca J. Passonneau, and Gerard de Melo. 2016. PEAK: Pyramid Evaluation via Automated Knowledge Extraction. In *Proceedings of the 30th Conference on Artificial Intelligence*. pages 2673–2679.

Ernst Zermelo. 1929. Die Berechnung der Turnier-Ergebnisse als ein Maximumproblem der Wahrscheinlichkeitsrechnung. *Mathematische Zeitschrift* 29:436–460.

Markus Zopf, Eneldo Loza Mencía, and Johannes Fürnkranz. 2016. Beyond Centrality and Structural Features: Learning Information Importance for Text Summarization. In *Proceedings of the 20th Conference on Computational Natural Language Learning*. Association for Computational Linguistics, pages 84–94.

Generating topic-oriented summaries using neural attention

Kundan Krishna
Adobe Research
kunkrish@adobe.com

Balaji Vasan Srinivasan
Adobe Research
balsrini@adobe.com

Abstract

Summarizing a document requires identifying the important parts of the document with an objective of providing a quick overview to a reader. However, a long article can span several topics and a single summary cannot do justice to all the topics. Further, the interests of readers can vary and the notion of importance can change across them. Existing summarization algorithms generate a single summary and are not capable of generating multiple summaries tuned to the interests of the readers. In this paper, we propose an attention based RNN framework to generate multiple summaries of a single document tuned to different topics of interest. Our method outperforms existing baselines and our results suggest that the attention of generative networks can be successfully biased to look at sentences relevant to a topic and effectively used to generate topic-tuned summaries.

1 Introduction

Automatic text summarization is the task of generating/extracting short text snippet that embodies the content of a larger document or a collection of documents in a concise fashion. Traditionally, researchers have used extractive methods for summarization - where a set of sentences is selected from an article and concatenated *as-is* to form the summary. Extractive methods are limited by their inability to paraphrase the content of the article using new sentences and hence are very different from the summaries created by humans, who paraphrase a given article to generate summaries. Recent works on summarization have come up with sequence-to-sequence models for generating summaries in a word-by-word fashion, thus 'generating' new sentences.

As articles get longer, it might span several topics and therefore, a single summary is often insufficient to satisfy the interests of the reader. In these cases, it is desirable to produce summaries that are aligned to the topical interests of reader to enable better consumption. Extractive summarization uses sentence-level features (Yang et al., 2017) that have been leveraged for producing query-focused or topic-based summaries. However, for RNN-based frameworks, such tuned summary generation is non-obvious due to the absence of explicit content based features.

To extend the advantages of abstractive summarization and generate topic-tuned summaries, we propose a neural encoder-decoder based framework which takes an article along with a topic of interest as input and generates a summary tuned to the target topic of interest. Our method works by training a neural framework to pay higher attention to parts of the input articles relevant to given topic. To overcome the lack of dataset containing articles with multiple topic-oriented summaries, we use a novel approach to artificially create a topic-centric training corpus from the CNN/Dailymail dataset (Hermann et al., 2015; Nallapati et al., 2016) for training our model.

Table 1 shows an article with different summaries generated by the proposed approach tuned towards the topics of politics, finance and social aspects. It can be seen that the *business* oriented summary talks about IMF's estimates about increase in taxes and the summary for *politics* talks about how it relates to the government's upcoming budget. The summary for the *social* topic elaborates on universal basic income and social security. Thus, for the same article, the proposed approach is capable of generating different summaries tuned to the input topic of interest.

Proceedings of NAACL-HLT 2018, pages 1697–1705
New Orleans, Louisiana, June 1 - 6, 2018.

Title: IMF backs Universal Basic Income in India, serves Modi govt a political opportunity **Article:** Ahead of Union Budget 2018, the Narendra Modi-led governments last full-year budget to be presented in February, the International Monetary Fund (IMF) has made a strong case for India adopting a fiscally neutral Universal Basic Income by eliminating both food and fuel subsidies ...
Business: imf claim eliminating energy " tax subsidies " would require a increase in fuel taxes and retail fuel prices such as petrol prices and tax of rs400 ($ 6) per tonne on coal consumption ...
Politics: narendra modi-led government 's last full-year budget to be presented in february. the international monetary fund has made a strong case for india adopting a fiscally neutral universal basic income by eliminating both food and fuel subsidies ...
Social: universal basic income is a form of social security guaranteed to citizens and transferred directly to their bank accounts and is being debated globally ...

Table 1: Topic oriented summaries generated by our method for an article (from LiveMint) touching multiple topics

2 Related Work

Traditional methods for summarization (Nenkova and McKeown, 2011) extract key sentences from the source text to construct the summary. Early works on abstractive summarization were focused on sentence compression based approaches (Filippova, 2010; Berg-Kirkpatrick et al., 2011; Banerjee et al., 2015) which connected fragments from multiple sentences to generate novel sentences for the summary or template based approaches that generated summaries by fitting content into a template (Wang and Cardie, 2013; Genest and Lapalme, 2011).

With the advent of deep sequence to sequence models which generated text word-by-word (Sutskever et al., 2014), attention based neural network models have been proposed for summarizing long sentences. Rush et al. (2015) first demonstrated the use of neural networks to generate shorter forms of long sentences. Nallapati et al. (2016) proposed a neural approach for abstractive summarization of large articles by applying the sequence to sequence model. See et al. (2017) further improved the performance on abstractive summarization of articles by introducing the ability to copy words from the source article (Gulcehre et al., 2016) using a pointer network (Vinyals et al., 2015), in addition to generating new words. However, all these frameworks focus on generating a single summary, and do not support topic-tuned summary generation. We use the architecture by See et al. as the starting point for our work and develop a method to generate topic-tuned summaries.

There have been some works on extending extractive summarization towards topical tuning. Lin and Hovy (2000) proposed the idea of extracting topic-based *signature terms* for summarization. Given a topic and a corpus of documents relevant and not relevant to the topic, a set of words characterizing each topic is extracted using a log-likelihood based measure. Sentences which contain these chosen words are assigned more importance while summarizing. Conroy et al. (2006) further extended the method for query-based multi-document summarization by considering sentence overlap with both query terms and topic signature words.

However, all these works rely on identifying sentence level features to compute topic affinities that are leveraged for choosing topic specific sentences for the summary. Since sequence-to-sequence frameworks generate text in a word-by-word fashion, incorporating sentence level statistics is not feasible in this framework. Therefore, we modify the attention of the network to focus on topic-specific parts of the input text to generate the tuned summaries.

3 Topic aware pointer-generator network

Our method builds on top of the pointer-generator network of See et al. which consists of an encoder and a decoder both based on LSTM architecture. Our contribution is a modified encoder to encode the article in a topic-sensitive manner. However, for the sake of completeness, we shall provide an overview of the entire network, and will reuse notations from their work to a large extent.

Given an input article as a sequence of words $a = w_1, w_2 \dots w_n$, and a scaled one-hot topic vector u_t, where t is a topic in a predefined set of topics T, the network produces a summary $s = s_1, s_2, \dots, s_k$ pertaining to the topic t. The topic-vector u_t has a non-zero value $c > 0$ only for the entry corresponding to the desired topic t of the summary. c designates the amount of bias that should be put towards the topic while generating the summary, higher values suggesting higher bias. The input article is passed through an embedding layer to transform them to lower-dimensional embedding $m_1, m_2, \dots, m_n$. The topic one-hot vector u_t is concatenated to each of the embedding to create the sequence $(m_1, u_t), (m_2, u_t), \dots, (m_n, u_t)$. This sequence is passed to a bidirectional LSTM encoder which computes a sequence of hidden states $h_1, h_2, ..h_n$. The final hidden state is passed to a decoder which

is also a LSTM, and it serves as the initial hidden state of the decoder. At each decoding step of the decoder, an attention distribution a^t is calculated over all words of the input article,

$$e_i^t = v^T \tanh(W_h h_i + W_s s_t + b_{att}) \quad (1)$$
$$a^t = \text{softmax}(e^t) \quad (2)$$

Here, s_t is the hidden state of the decoder at timestep t, and v, W_h, W_s and b_{att} are model parameters.

The attention model can be thought of as a probability distribution over words in the source text, which aids the decoder in generating the next word in the summary using words in the source text that have higher attention. Since the encoder has information about what is the topic of interest, the h_i are calculated in such a way that the corresponding areas of the article receiving high attention are relevant to the topic. We demonstrate this later in Section 4.2. The decoder uses this attention to calculate a context vector h_t^* as a weighted sum of the encoder hidden states to determine the next word that is to be generated.

$$h_t^* = \sum_{i=1}^{n} a_i^t h_i, \quad (3)$$

At each decoding time step, the decoder network also gets as input y_t, the last word in the summary generated so far and computes a scalar p_{gen} denoting the probability of the network generating a new word from the vocabulary.

$$p_{gen} = \sigma(w_h^T h_t^* + w_s^T s_t + w_y^T y_t + b_{gen}) \quad (4)$$

where w_h, w_s, w_y, b_{gen} are trained vectors. The network probabilistically decides based on p_{gen}, whether to generate a new word from the vocabulary or copy a word from the source text using the attention distribution.

For each word w in the vocabulary, the model calculates $P_{vocab}(w)$, the probability of the word getting newly generated next. P_{vocab} is calculated by passing a concatenation s_t and h_t^* through a linear transformation with sigmoid activation. Also, for each word w' in the input article, its total attention received yields its probability of being copied. Since some words occur in the vocabulary and also the input article, they will have non-zero probabilities of being newly generated as well as being copied. Therefore, the total probability of w being the next word generated in the summary, denoted by $\mathbf{p}$ is given by,

$$\mathbf{p}(w) = p_{gen} P_{vocab}(w) + (1 - p_{gen}) \sum_{i:w_i=w} a_i^t \quad (5)$$

The second term allows the framework to choose a word to copy from the input text using the attention distribution.

The pointer-generator network further employs a coverage mechanism which modifies the loss function to encourage diversity in attention distributions over time steps. This prevents the network from repeating the same phrases again while generating a summary. To handle out-of-vocabulary words, the pointer-generator network also appends them to the vocabulary before the article is encoded.

The training loss is set to be the average negative log-likelihood of the ground truth summaries. The model is trained using back-propagation and the Adagrad gradient descent algorithm (Duchi et al., 2011).

Alternate to the the proposed architecture, one can append the topic vector to the initial hidden state of the encoder, or the initial state of the decoder. However, in our experiments, these approaches did not produce the desired tuning. Another alternative would be to use different W_h attention matrix for each different topic and learn topic specific attention biases. This again did not perform well, perhaps due insufficient data available to train each topic's W_h.

3.1 Generating the training data

The problem of generating *multiple* different summaries of the *same* document based on topics of interest requires a set of articles, each of them (a) annotated with multiple (u_t, s) pairs, where t is a topic, and s is the summary of a oriented to topic t. However, existing datasets for single-document summarization (Over et al., 2007; Hermann et al., 2015) do not have multiple ground truth summaries oriented to different topics for each document. Moreover, existing datasets for topic-based summarization are focused on multi-document summarization and contain multiple articles tagged with a relevant topic. For example, DUC 2005 dataset (Dang, 2005) for topic based multi-document summarization contains 50 topics with each topic having 25 to 50 documents in it. To address the lack of datasets with multiple summaries of a single document, we propose

an approach create such a dataset artificially. The dataset is created in two steps.

We begin with fixing the set of topics for consideration and learn a word frequency based representation for each topic by using a corpus of articles where each article is labelled with a topic. For our experiment, we use the dataset of news articles tagged with topics like *politics*, *sports*, *education* etc. released at the 2017 KDD Data Science + Journalism Workshop (VoxMedia). We group the articles by the topics, so that all articles having topic t are represented by the set S_t. We represent each topic t as a vector $e_t = (n_1, n_2, \ldots, n_v)$ where $v = |V|$ is the size of vocabulary of words $V = \{w_1, w_2, ..., w_v\}$ and n_i is the number of times word w_i occurs in S_t. We normalize e_t to sum to 1.

We then take a corpus of articles with human generated summaries as a collection of (a, s) pairs. For our purposes, we use the CNN-Dailymail dataset which has articles with summaries. We create an *intermediate* dataset which consists of (a, u_t, s) pairs, where u_t is a one-hot representation of the topic t of summary s. We begin with labeling each summary with a topic by computing the dot-product between summary (in its bag-of-words representation) and the topic vectors extracted in the previous step. Let $< v_s, e_t >$ indicate the dot-product between the bag of words representation of summary s and the topic vector for topic t. For each topic t_i, $sim(s, t_i) =< v_s, e_{t_i} >$ is the similarity of summary s to the topic t_i. Let's say that t_i has the highest similarity to summary s and t_j has the second highest similarity. Then we say that summary s has topic t_i with confidence $c = \frac{sim(s,t_i)}{sim(s,t_j)}$. If the confidence is less than a threshold (set to 1.2 in our experiments), we drop the article and summary from our dataset. This enures that the intermediate dataset does not include summaries with more than one dominant topic. If the confidence is greater than the threshold, we add the triple (a, u_{t_i}, s) to the intermediate dataset, where the vector u_{t_i} keeps the value corresponding to t_i equal to the confidence c, instead of 1 as commonly done in one-hot vectors. This allows us to retain the confidence of s being of topic t_i while training our model.

To generate the final dataset, we follow the steps below:

1. Randomly pick (a_1, u_{t_1}, s_1) and (a_2, u_{t_2}, s_2) from the intermediate dataset such that $t_1 \neq t_2$.

2. Make a new article a' by sequentially picking up lines from a_1 and a_2. Each addition of a new line is done by randomly selecting one of a_1 or a_2 and extracting out a new line from the beginning of it. This ensures that the lines from a_1 occur in the same order in a' as originally in a_1, and the same thing is true for a_2 too. This ensures that the sequential flow of content is retained in the merger.

3. Add (a', u_{t_1}, s_1) to the final dataset.

4. Repeat step 2 to get a new article a'' and add (a'', u_{t_2}, s_2) to the final dataset.

5. Discard (a_1, u_{t_1}, s_1) and (a_2, u_{t_2}, s_2) from the intermediate dataset.

6. Repeat steps $1 - 5$ until the entire intermediate dataset is exhausted or all remaining instances in it have the same topic.

The created final dataset is used to train the proposed neural network for summarization. Since every article in the final dataset is a combination of two original articles, and the target summary to be generated is of one of them, the model must learn to distinguish between content coming from the two original articles. The randomization of position of sentences from each original article while merging, ensures that there is no position-specific bias that the model can use either. Since the two original articles have different topics, and the only information given to the model to hint whose summary is to be generated is the topic of one of them, the model is forced to learn what content is more relevant to the given topic and generate a summary accordingly.

We ended up with $112,360$ articles in our final dataset since many article-summary pairs from the CNNDailymail dataset were dropped due to insufficient confidence about their topics. Out of this, $103,666$ were used for training, $4,720$ for validation, and $3,974$ for testing.

4 Experimental Evaluation

To position our method against existing works, we use the following summarizers as our baselines.

Our first baseline is the **vanilla pointer generator (PG)** described in the original work of See et al. (2017). This method does not consider the

desired topic of summary when generating a summary. For an unbiased evaluation, we use exactly those unmerged article-summary pairs of the CNN/Dailymail dataset for training and validation which were eventually incorporated in the final dataset. Then the trained model is applied to generate summaries of the test set of the final dataset.

Our next baseline is a **frequency-based extraction** method that selects lines from the input article which are strongly aligned to the desired topic u_t. For each sentence, the relevance to each of the predefined topics is calculated using a dot product between their vector representations. The sentence is designated to the topic having the maximum relevance score, and the strength of alignment is the ratio of the the highest and the second highest relevance scores.

We extract all the sentences in the article which are aligned to the target topic, and run it through the pointer-generator network to create the summary. We refer to this baseline as **abstractive summarizer with frequency based extraction (*Freq-Abs*)**. Alternatively, we take k sentences which have the highest strength of alignment with the target topic to create a purely extractive summary. k was set to 3 in accordance with the average number of sentences in the summaries of the training set (2.83). We call this method **extractive summarizer with frequency based extraction (*Freq-Ext*)**.

Our last baseline is a **topic-signature based approach** which also works by extracting sentences from the article which are aligned to the target topic. However, the selection of sentences is based on topic signatures as described by Lin and Hovy (2000) and Conroy et al. (2006) instead of word frequencies. A topic signature is a set of words relevant to the topic. For any given sentence, the number of signature terms of each topic is computed. The sentence is designated to belong to the topic which has the highest number of its signature terms occurring in it.

The topic signature is determined based a set of documents T relevant to the topic, and a set of background documents T' that is indicative of general topics. It is assumed that in T and T', the occurrence of each word w follows a binomial distribution with probability of occurrence p. The likelihood of observing T and T' is calculated under two hypotheses - one where the probability of occurrence of w is p_1 in T and p_2 in T' such that $p_1 > p_2$, and the other where it is p in both T and T'. The ratio of likelihoods is calculated and words for which this ratio is the highest are included as part of the topic's signature.

We extracted topic signatures using the summaries of the training dataset as our corpus. For each topic t, the corresponding summaries form the topic specific corpus T, and the remaining summaries make the background corpus. Table 2 shows a subset of the topic signatures.

Topic	Signature words
business	stock, firm, rate, google, worth, companies
education	children, teacher, parents, schools, test
entertainment	actor, star, league, movie, will, taylor
health	disease, brain, people, risk, league, infection, blood
military	arrested, police, car, officer, year, officers, killed, charged
politics	party, campaign, secretary, democrats, bill, congress
social	church, rich, life, woman, sex, identity, gender, will
sports	scored, united, league, nfl, cup, england, beat, game
technology	app, solar, launch, ipod, app, earth, energy, oil

Table 2: Words from signatures of different topics

Analogous to Freq-Abs and Freq-Ext, we try two alternatives here as well - **abstractive summarizer with signature based extraction (*Sign-Abs*)** and **extractive summarizer with signature based extraction (*Sign-Ext*)**.

4.1 Performance on the created dataset

We used the $3,974$ article-topic-summary tuples from our final dataset to evaluate the performance of the summarizers. The models were given the input article and topic and the generated summary was compared with the ground truth summary. We use ROUGE scores to measure the quality of summaries. ROUGE scores measure the precision, recall and f-measure for the occurrence of n-grams in the generated summary with respect to the reference human generated summary. We use the ROUGE-1, ROUGE-2 and ROUGE-L variants (Lin and Och, 2004) which look at unigram, bigram and longest common subsequence overlaps between generated and reference summaries. Table 3 shows the ROUGE F1 scores for the topic-based summaries generated by different methods. It is easy to see that the proposed method yields the best performance across all the baselines.

Further, we also observed that the summaries generated by our system show abstractive nature as noted in See et al. (2017). Table 4 shows some instances where our model used new words unseen in the article.

Algorithm	ROUGE-1	ROUGE-2	ROUGE-L
PG	26.8	9.2	24.5
Freq-Abs	25.8	8.4	23.4
Freq-Ext	25.5	8.5	22.9
Sign-Abs	26.1	8.5	23.7
Sign-Ext	25.9	8.7	23.3
Our method	**34.1**	**13.6**	**31.2**

Table 3: ROUGE F1 scores obtained by various methods on the final test set

Article: spain 's 2-0 defeat by holland on tuesday brought back bitter memories of their disastrous 2014 world cup , but coach vicente del bosque will not be too worried ... **Summary:** holland **beat** spain 2-0 at the amsterdam arena on tuesday **night**
Article: it 's 11 years since arsenal won the title. they went from invincibles to incapables ... **Summary:** arsene wenger 's **side** have 15 wins in 17 **appearances**

Table 4: Summaries where our model uses new words not seen in the input article

Title: Paul McGowan won't be risked in final six games of the season, as Dundee boss Paul Hartley looks to help troubled midfielder **Military:** dundee rogue paul mcgowan has been handed his third conviction after assaulting a police officer . mcgowan escaped a jail sentence but was placed under a restriction ... **Sports:** paul hartley has warned that paul mcgowan may not feature in any of the team 's remaining six games this season because he will not risk playing the troubled midfielder this season ...
Title: Third suspect arrested in alleged Panama City gang rape **Education:** ryan calhoun has been a student at middle tennessee state university . the two are students and have been " placed on temporary suspension and disciplinary procedures ... **Military:** sheriff 's office : third person has been arrested in the case of an alleged spring break gang rape that was videotaped on a crowded stretch of panama city beach , the bay county , florida , sheriff 's office said . the arrests come after a woman told police ...
Title: University of Michigan will go forward with 'American Sniper' screening **Military:** kyle was fatally shot at a texas shooting range in 2013 . kyle cooper was nominated for an oscar for his portrayal of chris kyle , a navy seal and most lethal sniper in u.s. military history ... **Education:** university of michigan has decided to proceed with a screening of the film " american sniper " despite objections from some students . more than 200 signed a petition ...
Title: The Pope's old iPad sells for $30,500 **Business:** the motorcycle sold for $ 284,000 at auction , more than 10 times its normal sales price . a harley motorcycle jacket signed by francis sold for nearly $ 68,000 . ... **Education:** ... the proceeds will go to a school in montevideo , uruguay . it 's not the first time a papal hand-me-down has gone for big bucks . ...

Table 5: Topic oriented summaries generated by our method for articles from CNN-Dailymail dataset

4.2 Performance on multi-topic articles

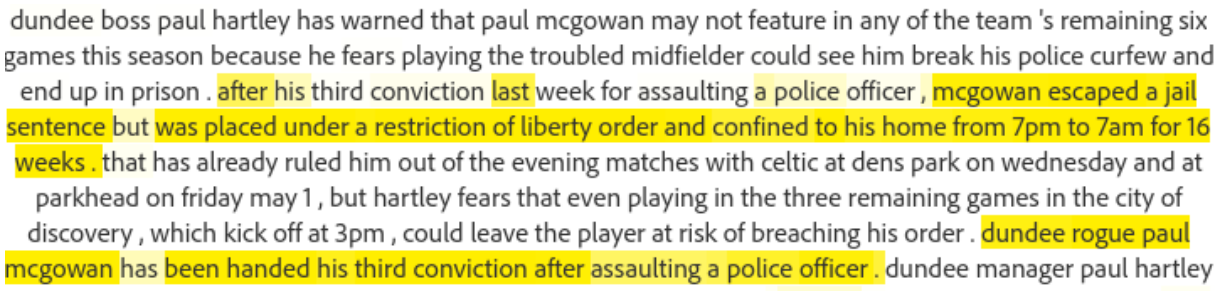
We ran the proposed model on original articles from CNN-DailyMail dataset which were not part of the final dataset used for training. Since the articles were not annotated with topics, we generated summaries for all the different topics. We then detected articles where the method generated different summaries for different topics suggesting the presence of more than one topic in the article. Table 5 shows a few summaries generated by the proposed approach where different summaries were generated aligned to the input topics. The first article talks about the dropping of a player from a football squad. The summary oriented to the *military* topic talks about the assault of a police officer by the player and his criminal history. The *sports* summary talks about the player's fate in the remaining games of the season.

Similarly, we have different summaries for the second article, where the *education* oriented summary talks about the educational affiliations of the suspects and disciplinary procedures, whereas the *military* summary talks about the arrests. Note that these are among the original articles in the dataset which were not used for creating any of the articles in our final data used for training the model.

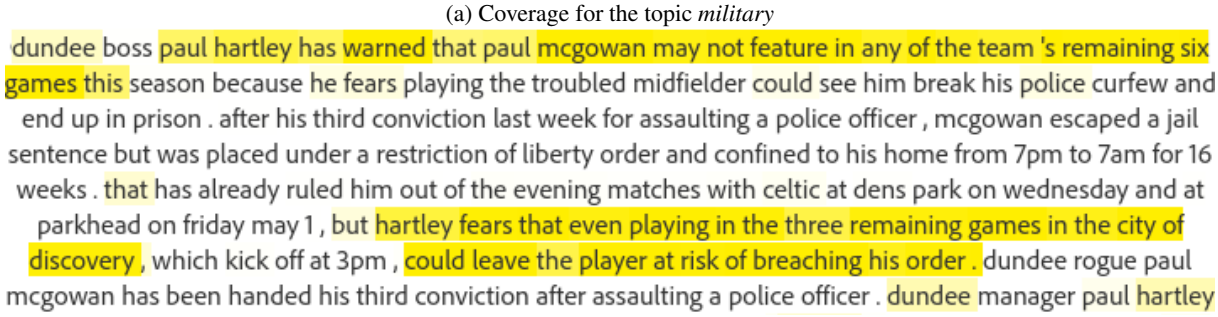
We also observe that the attention distribution varies according to the input topic of summary. We investigate the amount of cumulative attention (defined as coverage by See et al.) that is received by each term summed over all the decoding steps during generation. An example of the variation in attention over an input article is shown in Figure 1. The degree of yellow hue is used to represent the value of coverage for each term. For the topic *military*, words like *jail* and *assualting* receive higher attention, whereas for the topic *sports* words like *games* and *player* are highlighted. Figure 1(b) also shows an example where our model summarizes by skipping a clause (*"which kick off at 3pm"*).

4.3 Human evaluation of performance

Finally, we performed human evaluations of summaries to compare the quality of our method against the baselines. We restrict ourselves to



(a) Coverage for the topic *military*



(b) Coverage for the topic *sports*

Figure 1: Variation in the attention coverage while summarizing an article for different topics

the best performing baseline - the topic signature based abstractive summarizer (Sign-Abs), and the vanilla pointer-generator.

We fetched articles touching multiple topics using the NYTimes Search API[1], which allows to search for articles on NYTimes which appear in the news desk for a topic (the major topic) and are tagged with another topic (the minor topic). For different pairs of topics, we retrieved relevant articles using the API and randomly selected few of them to generate summaries tuned to the two topics using our method and the baselines. We retrieved 18 total articles for the evaluation. Each article had two topics leading to 36 (article,topic) pairs for the summary generation. For each (article,topic) pair, annotators were shown two summaries - one generated by our method and the other by one of the baselines. The task was to choose the summary more relevant to the topic. Each such task was annotated by 10 different annotators. Every annotator was assigned 9 tasks and 1 extra dummy task to check if they were paying attention. Annotations from evaluators who answered incorrectly to the dummy task were discarded. We had a total of 720 annotations from the human evaluation and their summary is shown in Table 6.

The value under "Overall annotations" refers to the fraction of all human responses across all documents which rated the summary produced by our method better than the alternate summary produced by the compared baseline method. Understandably, the proposed approach is comparable in the preference to both the baselines for the major topic (0.5667) - since most standard summaries will cover the primary topic of the input. However, for the minor topic, it can be seen that the proposed approach is better than the baselines.

The value under "Document annotations" indicates the fraction of times the proposed method was preferred by half or more of the annotators for a document-topic pair. It is easy to see that the proposed approach clearly outperforms the baselines under this scenario. The difference in performance is even more significant for summaries generated for the non-major topic since our approach is capable of efficiently generating tuned summaries for minor topics as well.

5 Conclusion

We proposed a method for generating multiple abstractive summaries of a given document oriented towards different topics of interest. Our method works by modifying the attention mechanism of a pointer-generator neural network to make it focus on text relevant to a topic. For training our network, we devised a novel way to create a dataset where articles are tagged with topic oriented summaries. Our method outperformed previous fea-

[1] https://developer.nytimes.com/

Topics	Overall Annotations		Document Annotations	
	vs PG	vs Sign-Abs	vs PG	vs Sign-Abs
All	0.5889	0.6111	0.7222	0.8333
Major	0.5667	0.5667	0.6667	0.7778
Minor	0.6111	0.6556	0.7778	0.8889

Table 6: Evaluation of summaries of the proposed approach against Pointer Generator Framework and Topic Signature based Summarizer by human annotators

ture based methods for topic oriented summarization using word frequencies or log likelihood ratios.

References

Siddhartha Banerjee, Prasenjit Mitra, and Kazunari Sugiyama. 2015. Multi-document abstractive summarization using ilp based multi-sentence compression. In *IJCAI*. pages 1208–1214.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pages 481–490.

John M Conroy, Judith D Schlesinger, and Dianne P O'Leary. 2006. Topic-focused multi-document summarization using an approximate oracle score. In *Proceedings of the COLING/ACL on Main conference poster sessions*. Association for Computational Linguistics, pages 152–159.

Hoa Trang Dang. 2005. Overview of duc 2005. In *Proceedings of the document understanding conference*. volume 2005, pages 1–12.

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

Katja Filippova. 2010. Multi-sentence compression: Finding shortest paths in word graphs. In *Proceedings of the 23rd International Conference on Computational Linguistics*. Association for Computational Linguistics, pages 322–330.

Pierre-Etienne Genest and Guy Lapalme. 2011. Framework for abstractive summarization using text-to-text generation. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*. Association for Computational Linguistics, pages 64–73.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 140–149.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*. pages 1693–1701.

Chin-Yew Lin and Eduard Hovy. 2000. The automated acquisition of topic signatures for text summarization. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, pages 495–501.

Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, page 605.

LiveMint. 2017. IMF backs Universal Basic Income in India, serves Modi govt a political opportunity. [Online; accessed 8-Dec-2017].

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Ça glar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *CoNLL 2016* page 280.

Ani Nenkova and Kathleen McKeown. 2011. Automatic summarization. *Foundations and Trends® in Information Retrieval* 5(2–3):103–233.

Paul Over, Hoa Dang, and Donna Harman. 2007. Duc in context. *Information Processing & Management* 43(6):1506–1520.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 379–389.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 1073–1083.

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

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*. pages 2692–2700.

VoxMedia. 2017. Data science + journalism @ kdd 2017. https://sites.google.com/view/dsandj2017/datasets. [Online; accessed 8-Dec-2017].

Lu Wang and Claire Cardie. 2013. Domain-independent abstract generation for focused meeting summarization. In *ACL (1)*. pages 1395–1405.

Yinfei Yang, Forrest Sheng Bao, and Ani Nenkova. 2017. Detecting (un) important content for single-document news summarization. *EACL 2017* page 707.

Generative Bridging Network for Neural Sequence Prediction

Wenhu Chen,[1] **Guanlin Li,**[2] **Shuo Ren,**[5] **Shujie Liu,**[3] **Zhirui Zhang,**[4] **Mu Li,**[3] **Ming Zhou**[3]
University of California, Santa Barbara[1]
Harbin Institute of Technology[2]
Microsoft Research Asia[3]
University of Science and Technology of China[4]
Beijing University of Aeronautics and Astronautics[5]
wenhuchen@cs.ucsb.edu epsilonlee.green@gmail.com {v-shure, shujliu, v-zhirzh, muli, mingzhou}@microsoft.com

Abstract

In order to alleviate data sparsity and overfitting problems in maximum likelihood estimation (MLE) for sequence prediction tasks, we propose the Generative Bridging Network (GBN), in which a novel bridge module is introduced to assist the training of the sequence prediction model (the generator network). Unlike MLE directly maximizing the conditional likelihood, the bridge extends the point-wise ground truth to a bridge distribution conditioned on it, and the generator is optimized to minimize their KL-divergence. Three different GBNs, namely uniform GBN, language-model GBN and coaching GBN, are proposed to penalize confidence, enhance language smoothness and relieve learning burden. Experiments conducted on two recognized sequence prediction tasks (machine translation and abstractive text summarization) show that our proposed GBNs can yield significant improvements over strong baselines. Furthermore, by analyzing samples drawn from different bridges, expected influences on the generator are verified.

1 Introduction

Sequence prediction has been widely used in tasks where the outputs are sequentially structured and mutually dependent. Recently, massive explorations in this area have been made to solve practical problems, such as machine translation (Bahdanau et al., 2014; Ma et al., 2017; Norouzi et al., 2016), syntactic parsing (Vinyals et al., 2015), spelling correction (Bahdanau et al., 2014), image captioning (Xu et al., 2015) and speech recognition (Chorowski et al., 2015). Armed with modern computation power, deep LSTM (Hochreiter and Schmidhuber, 1997) or GRU (Chung et al., 2014) based neural sequence prediction models have achieved the state-of-the-art performance.

The typical training algorithm for sequence prediction is Maximum Likelihood Estimation (MLE), which maximizes the likelihood of the target sequences conditioned on the source ones:

$$\theta^* = \arg\max_{\theta} \mathop{\mathbb{E}}_{(X,Y^*)\sim D} \log p_\theta(Y^*|X) \quad (1)$$

Despite the popularity of MLE or teacher forcing (Doya, 1992) in neural sequence prediction tasks, two general issues are always haunting: 1). data sparsity and 2). tendency for overfitting, with which can both harm model generalization.

To combat data sparsity, different strategies have been proposed. Most of them try to take advantage of monolingual data (Sennrich et al., 2015; Zhang and Zong, 2016; Cheng et al., 2016). Others try to modify the ground truth target based on derived rules to get more similar examples for training (Norouzi et al., 2016; Ma et al., 2017). To alleviate overfitting, regularization techniques,

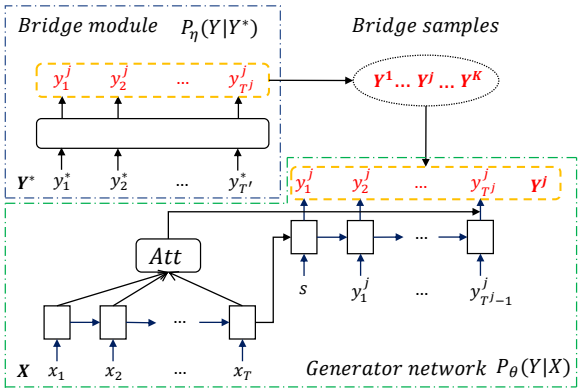


Figure 1: The overall architecture of our novel Generative Bridging Network (GBN). Two main components, namely the generator network and the bridge module, are connected through samples ($Y^1 \dots Y^K$ in red) from the bridge module during training time. (We sometimes call them generator and bridge in brief respectively in the following discussion.) The generator is implemented through an attentive encoder-decoder, where in the figure *Att* represents the attention module.

Proceedings of NAACL-HLT 2018, pages 1706–1715
New Orleans, Louisiana, June 1 - 6, 2018.

such as confidence penalization (Pereyra et al., 2017) and posterior regularization (Zhang et al., 2017), are proposed recently.

As shown in Figure 1, we propose a novel learning architecture, titled Generative Bridging Network (GBN), to combine both of the benefits from synthetic data and regularization. Within the architecture, the bridge module (bridge) first transforms the point-wise ground truth into a bridge distribution, which can be viewed as a target proposer from whom more target examples are drawn to train the generator. By introducing different constraints, the bridge can be set or trained to possess specific property, with which the drawn samples can augment target-side data (alleviate data sparsity) while regularizing the training (avoid overfitting) of the generator network (generator).

In this paper, we introduce three different constraints to build three bridge modules. Together with the generator network, three GBN systems are constructed: 1). a uniform GBN, instantiating the constraint as a uniform distribution to penalize confidence; 2). a language-model GBN, instantiating the constraint as a pre-trained neural language model to increase language smoothness; 3). a coaching GBN, instantiating the constraint as the generator's output distribution to seek a close-to-generator distribution, which enables the bridge to draw easy-to-learn samples for the generator to learn. Without any constraint, our GBN degrades to MLE. The uniform GBN is proved to minimize KL-divergence with a so-called payoff distribution as in reward augmented maximum likelihood or RAML (Norouzi et al., 2016).

Experiments are conducted on two sequence prediction tasks, namely machine translation and abstractive text summarization. On both of them, our proposed GBNs can significantly improve task performance, compared with strong baselines. Among them, the coaching GBN achieves the best. Samples from these three different bridges are demonstrated to confirm the expected impacts they have on the training of the generator. In summary, our contributions are:

- A novel GBN architecture is proposed for sequence prediction to alleviate the data sparsity and overfitting problems, where the bridge module and the generator network are integrated and jointly trained.
- Different constraints are introduced to build GBN variants: uniform GBN, language-model GBN and coaching GBN. Our GBN architecture is proved to be a generalized form of both MLE and RAML.
- All proposed GBN variants outperform the MLE baselines on machine translation and abstractive text summarization. Similar relative improvements are achieved compared to recent state-of-the-art methods in the translation task. We also demonstrate the advantage of our GBNs qualitatively by comparing ground truth and samples from bridges.

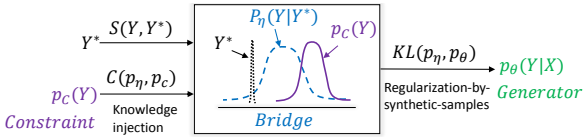


Figure 2: Conceptual interpretation of our Generative Bridging Network (GBN). See detailed discussion in the beginning of Sec. 2.

2 Generative Bridging Network

In this section, we first give a conceptual interpretation of our novel learning architecture which is sketched in Figure 2. Since data augmentation and regularization are two golden solutions for tackling data sparsity and overfitting issues. We are willing to design an architecture which can integrate both of their benefits. The basic idea is to use a so-called bridge which transforms Y^* to an easy-to-sample distribution, and then use this distribution (samples) to train and meanwhile regularize the sequence prediction model (the generator).

The bridge is viewed as a conditional distribution[1] $p_\eta(Y|Y^*)$ to get more target Ys given Y^* so as to construct more training pairs (X, Y). In the meantime, we could inject (empirical) prior knowledge into the bridge through its optimization objective which is inspired by the design of the payoff distribution in RAML. We formulate the optimization objective with two parts in Equation (2): a) an expected similarity score computed through a similarity score function $S(\cdot, Y^*)$ interpolated with b) a knowledge injection constraint[2] $\mathcal{C}(p_\eta(Y|Y^*), p_c(Y))$ where α controls the

[1] η should be treated as an index of the bridge distribution, so it is not necessarily the parameters to be learned.

[2] Note that, in our paper, we specify $\mathcal{C}$ to be KL-divergence between the bridge distribution p_η and certain constraint dis-

strength of the regularization, formally, we write the objective function $L_B(\eta)$ as follows:

$$L_B(\eta) = \mathop{\mathbb{E}}_{Y\sim p_\eta(Y|Y^*)}[-S(Y,Y^*)] + \alpha\mathcal{C}(p_\eta(Y|Y^*), p_c(Y)) \tag{2}$$

Minimizing it empowers the bridge distribution not only to concentrate its mass around the ground truth Y^* but also to adopt certain hope property from $p_c(Y)$. With the constructed bridge distribution, we optimize the generator network $P_\theta(Y|X)$ to match its output distribution towards the bridge distribution by minimizing their KL-divergence:

$$L_G(\theta) = KL(p_\eta(Y|Y^*)||p_\theta(Y|X)) \tag{3}$$

In practice, the KL-divergence is approximated through sampling process detailed in Sec. 2.3. As a matter of fact, the bridge is the crux of the integration: it synthesizes new targets to alleviate data sparsity and then uses the synthetic data as regularization to overcome overfitting. Thus a regularization-by-synthetic-example approach, which is very similar to the prior-incorporation-by-virtual-example method (Niyogi et al., 1998).

2.1 Generator Network

Our generator network is parameterized with the commonly used encoder-decoder architecture (Bahdanau et al., 2014; Cho et al., 2014). The encoder is used to encode the input sequence X to a sequence of hidden states, based on which an attention mechanism is leveraged to compute context vectors at the decoding stage. The context vector together with previous decoder's hidden state and previously predicted label are used, at each time step, to compute the next hidden state and predict an output label.

As claimed in Equation (3), the generator network is not trained to maximize the likelihood of the ground truth but tries best to match the bridge distribution, which is a delegate of the ground truth. We use gradient descent to optimize the KL-divergence with respect to the generator:

$$\nabla L_G(\theta) = \mathop{\mathbb{E}}_{Y\sim p_\eta(Y|Y^*)} \log \nabla p_\theta(Y|X) \tag{4}$$

The optimization process can be viewed as the generator maximizing the likelihood of samples drawn from the bridge. This may alleviate data sparsity and overfitting by posing more unseen scenarios to the generator and may help the generator generalize better in test time.

2.2 Bridge Module[3]

Our bridge module is designed to transform a single target example Y^* to a bridge distribution $p_\eta(Y|Y^*)$. We design its optimization target in Equation (2) to consist of two terms, namely, a concentration requirement and a constraint. The constraint is instantiated as KL-divergence between the bridge and a contraint distribution $p_c(Y)$. We transform Equation (2) as follows, which is convenient for mathematical manipulation later:

$$L_B(\eta) = \mathop{\mathbb{E}}_{Y\sim p_\eta}[-\frac{S(Y,Y^*)}{\tau}] + KL(p_\eta(Y|Y^*)||p_c(Y)) \tag{5}$$

$S(Y,Y^*)$ is a predefined score function which measures similarity between Y and Y^* and peaks when $Y = Y^*$, while $p_c(Y)$ reshapes the bridge distribution. More specifically, the first term ensures that the bridge should concentrate around the ground truth Y^*, and the second introduces willing property which can help regularize the generator. The hyperparameter τ can be interpreted as a temperature which scales the score function. In the following bridge specifications, the score function $S(Y,Y^*)$ is instantiated according to Sec. 3.1.

1. Delta Bridge The delta bridge can be seen as the simplest case where $\alpha = 0$ or no constraint is imposed. The bridge seeks to minimize $\mathop{\mathbb{E}}_{Y\sim p_\eta(Y|Y^*)}[-\frac{S(Y,Y^*)}{\tau}]$. The optimal solution is when the bridge only samples Y^*, thus the Dirac delta distribution is described as follows:

$$p_\eta(Y|Y^*) = \delta_{Y^*}(Y) \tag{6}$$

This exactly corresponds to MLE, where only examples in the dataset are used to train the generator. We regard this case as our baseline.

2. Uniform Bridge The uniform bridge adopts a uniform distribution $U(Y)$ as constraint. This

tribution p_c, however, we believe mathematical form of $\mathcal{C}$ is not restricted, which could motivate further development.

[3] Although we name it bridge module, we explicitly learn it with the generator when a closed-form static solution exists in terms of Equation (5). Otherwise, we will adopt an encoder-decoder to construct a dynamic bridge network.

bridge motivates to include noise into target example, which is similar to label smoothing (Szegedy et al., 2016). The loss function can be written as:

$$L_B(\eta) = \mathop{\mathbb{E}}_{Y\sim p_\eta}[-\frac{S(Y,Y^*)}{\tau}] + KL(p_\eta(Y|Y^*)||U(Y)) \tag{7}$$

We can re-write it as follows by adding a constant to not change the optimization result:

$$L_B(\eta) + C = KL(p_\eta(Y|Y^*)||\frac{\exp\frac{S(Y,Y^*)}{\tau}}{Z}) \tag{8}$$

This bridge is static for having a closed-form solution:

$$p_\eta(Y|Y^*) = \frac{\exp\frac{S(Y,Y^*)}{\tau}}{Z} \tag{9}$$

where Z is the partition function. Note that our uniform bridge corresponds to the payoff distribution described in RAML (Norouzi et al., 2016).

3. Language-model (LM) Bridge The LM bridge utilizes a pretrained neural language model $p_{LM}(Y)$ as constraint, which motivates to propose target examples conforming to language fluency.

$$L_B(\eta) = \mathop{\mathbb{E}}_{Y\sim p_\eta(Y|Y^*)}[-\frac{S(Y,Y^*)}{\tau}] + KL(p_\eta(Y|Y^*)||p_{LM}) \tag{10}$$

Similar to the uniform bridge case, we can re-write the loss function to a KL-divergence:

$$\begin{aligned} &L_B(\eta) + C \\ =&KL(p_\eta(Y|Y^*)||\frac{p_{LM}(Y)\cdot\exp\frac{S(Y,Y^*)}{\tau}}{Z}) \end{aligned} \tag{11}$$

Thus, the LM bridge is also static and can be seen as an extension of the uniform bridge, where the exponentiated similarity score is re-weighted by a pretrained LM score, and renormalized:

$$p(Y|Y^*) = \frac{p_{LM}(Y)\exp\frac{S(Y,Y^*)}{\tau}}{Z} \tag{12}$$

where Z is the partition function. The above equation looks just like the payoff distribution, whereas an additional factor is considered.

4. Coaching Bridge The coaching bridge utilizes the generator's output distribution as constraint, which motivates to generate training samples which are easy to be understood by the generator, so as to relieve its learning burden. The coaching bridge follows the same spirit as the coach proposed in Imitation-via-Coaching (He et al., 2012), which, in reinforcement learning vocabulary, advocates to guide the policy (generator) with easy-to-learn action trajectories and let it gradually approach the oracle when the optimal action is hard to achieve.

$$L_B(\eta) = \mathop{\mathbb{E}}_{Y\sim p_\eta}[-\frac{S(Y,Y^*)}{\tau}] + KL(p_\theta(Y|X)||p_\eta(Y|Y^*)) \tag{13}$$

Since the KL constraint is a moving target when the generator is updated, the coaching bridge should not remain static. Therefore, we perform iterative optimization to train the bridge and the generator jointly. Formally, the derivatives for the coaching bridge are written as follows:

$$\begin{aligned} \nabla L_B(\eta) = &\mathop{\mathbb{E}}_{Y\sim p_\eta}[-\frac{S(Y,Y^*)}{\tau}\nabla\log p_\eta(Y|Y^*)] \\ &+ \mathop{\mathbb{E}}_{Y\sim p_\theta}\nabla\log p_\eta(Y|Y^*) \end{aligned} \tag{14}$$

The first term corresponds to the policy gradient algorithm described in REINFORCE (Williams, 1992), where the coefficient $-S(Y,Y^*)/\tau$ corresponds to reward function. Due to the mutual dependence between bridge module and generator network, we design an iterative training strategy, i.e. the two networks take turns to update their own parameters treating the other as fixed.

2.3 Training

The training of the above three variants is illustrated in Figure 3. Since the proposed bridges can be divided into static ones, which only require pretraining, and dynamic ones, which require continual training with the generator, we describe their training process in details respectively.

2.3.1 Stratified-Sampled Training

Since closed-formed optimal distributions can be found for uniform/LM GBNs, we only need to draw samples from the static bridge distributions to train our sequence generator. Unfortunately,

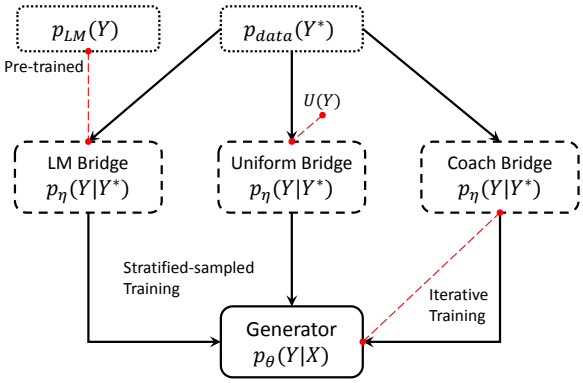


Figure 3: The training processes of the three different variants of our GBN architecture (Sec. 2.3).

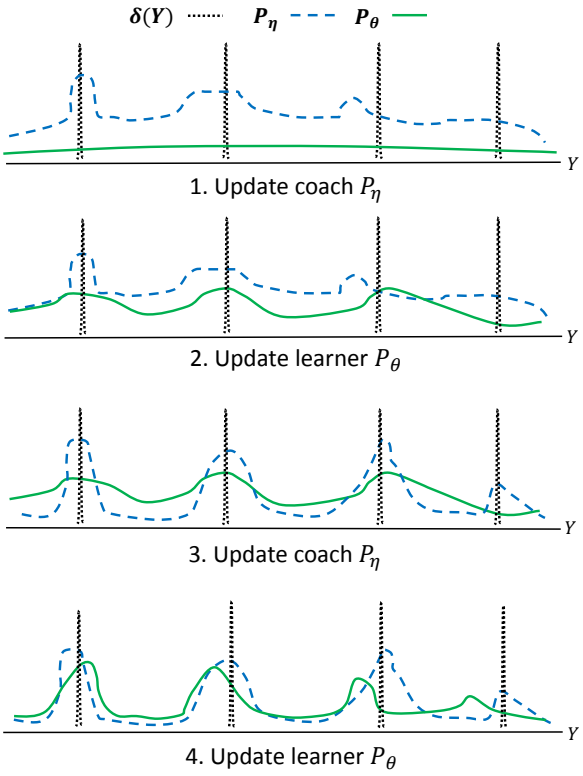


Figure 4: Four iterative updates of the coaching bridge and the generator. In an early stage, the pre-trained generator P_θ may not put mass on some ground truth target points within the output space, shown by $\delta(Y)$. The coaching bridge is first updated with Equation (14) to locate in between the Dirac delta distribution and the generator's output distribution. Then, by sampling from the coaching bridge for approximating Equation (4), target samples which demonstrate easy-to-learn sequence segments facilitate the generator to be optimized to achieve closeness with the coaching bridge. Then this process repeats until the generator converges.

due to the intractability of these bridge distributions, direct sampling is infeasible. Therefore, we follow Norouzi et al. (2016); Ma et al. (2017) and adopt stratified sampling to approximate the direct sampling process. Given a sentence Y^*, we first sample an edit distance m, and then randomly select m positions to replace the original tokens. The difference between the uniform and the LM bridge lies in that the uniform bridge replaces labels by drawing substitutions from a uniform distribution, while LM bridge takes the history as condition and draws substitutions from its step-wise distribution.

2.3.2 Iterative Training

Since the KL-constraint is a moving target for the coaching bridge, an iterative training strategy is designed to alternately update both the generator and the bridge (Algorithm 1). We first pre-train both the generator and the bridge and then start to alternately update their parameters. Figure 4 intuitively demonstrates the intertwined optimization effects over the coaching bridge and the generator. We hypothesize that iterative training with easy-to-learn guidance could benefit gradient update, thus result in better local minimum.

3 Experiment

We select machine translation and abstractive text summarization as benchmarks to verify our GBN framework.

3.1 Similarity Score Function

In our experiments, instead of directly using BLEU or ROUGE as reward to guide the bridge network's policy search, we design a simple surrogate n-gram matching reward as follows:

$$S(Y, Y^*) = 0.4{*}N_4{+}0.3{*}N_3{+}0.2{*}N_2{+}0.1{*}N_1 \quad (15)$$

N_n represents the n-gram matching score between Y and Y^*. In order to alleviate reward sparsity at sequence level, we further decompose the global reward $S(Y, Y^*)$ as a series of local rewards at every time step. Formally, we write the step-wise reward $s(y_t|y_{1:t-1}, Y^*)$ as follows:

$$s(y_t|y_{1:t-1}, Y^*) = \begin{cases} 1.0; N(y_{1:t}, y_{t-3:t}) \leq N(Y^*, y_{t-3:t}) \\ 0.6; N(y_{1:t}, y_{t-2:t}) \leq N(Y^*, y_{t-2:t}) \\ 0.3; N(y_{1:t}, y_{t-1:t}) \leq N(Y^*, y_{t-1:t}) \\ 0.1; N(y_{1:t}, y_t) \leq N(Y^*, y_t) \\ 0.0; otherwise \end{cases} \quad (16)$$

where $N(Y, \tilde{Y})$ represents the occurrence of subsequence $\tilde{Y}$ in whole sequence Y. Specifically, if

Algorithm 1 Training Coaching GBN

procedure PRE-TRAINING
Initialize $p_\theta(Y|X)$ and $p_\eta(Y|Y^*)$ with random weights θ and η
Pre-train $p_\theta(Y|X)$ to predict Y^* given X
Use pre-trained $p_\theta(Y|X)$ to generate $\hat{Y}$ given X
Pre-train $p_\eta(Y|Y^*)$ to predict $\hat{Y}$ given Y^*
end procedure
procedure ITERATIVE-TRAINING
while Not Converged **do**
Receive a random example (X, Y^*)
if Bridge-step **then**
Draw samples Y from $p_\theta(Y|X)$
Update bridge via Equation (14)
else if Generator-step **then**
Draw samples Y from $p_\eta(Y|Y^*)$
Update generator via Equation (4)
end if
end while
end procedure

a certain sub-sequence $y_{t-n+1:t}$ from Y appears less times than in the reference Y^*, y_t receives reward. Formally, we rewrite the step-level gradient for each sampled Y as follows:

$$
\begin{aligned}
&-\frac{S(Y,Y^*)}{\tau}\nabla \log p_\eta(Y|Y^*) \\
&=\sum_t -\frac{s(y_t|y_{1:t-1},Y^*)}{\tau}\cdot\nabla \log p_\eta(y_t|y_{1:t-1},Y^*)
\end{aligned}
\tag{17}
$$

3.2 Machine Translation

Dataset We follow Ranzato et al. (2015); Bahdanau et al. (2016) and select German-English machine translation track of the IWSLT 2014 evaluation campaign. The corpus contains sentence-wise aligned subtitles of TED and TEDx talks. We use Moses toolkit (Koehn et al., 2007) and remove sentences longer than 50 words as well as lower-casing. The evaluation metric is BLEU (Papineni et al., 2002) computed via the multi-bleu.perl.

System Setting We use a unified GRU-based RNN (Chung et al., 2014) for both the generator and the coaching bridge. In order to compare with existing papers, we use a similar system setting with 512 RNN hidden units and 256 as embedding size. We use attentive encoder-decoder to build our system (Bahdanau et al., 2014). During training, we apply ADADELTA (Zeiler, 2012) with $\epsilon = 10^{-6}$ and $\rho = 0.95$ to optimize parameters of the generator and the coaching bridge. During decoding, a beam size of 8 is used to approximate the full search space. An important hyper-parameter for our experiments is the temperature τ. For the uniform/LM bridge, we follow Norouzi et al. (2016) to adopt an optimal temperature $\tau = 0.8$. And for the coaching bridge, we test hyper-parameters from $\tau \in \{0.8, 1.0, 1.2\}$. Besides comparing with our fine-tuned baseline, other systems for comparison of relative BLEU improvement are: MIXER (Ranzato et al., 2015), BSO (Wiseman and Rush, 2016), AC (Bahdanau et al., 2016), Softmax-Q (Ma et al., 2017).

Methods	Baseline	Model
MIXER	20.10	21.81 +1.71
BSO	24.03	26.36 +2.33
AC	27.56	28.53 +0.97
Softmax-Q	27.66	28.77 +1.11
Uniform GBN ($\tau = 0.8$)	29.10	29.80 +0.70
LM GBN ($\tau = 0.8$)		29.90 +0.80
Coaching GBN ($\tau = 0.8$)		29.98 +0.88
Coaching GBN ($\tau = 1.2$)		30.15 +1.05
Coaching GBN ($\tau = 1.0$)		**30.18** +1.08

Table 1: Comparison with existing works on IWSLT-2014 German-English Machine Translation Task.

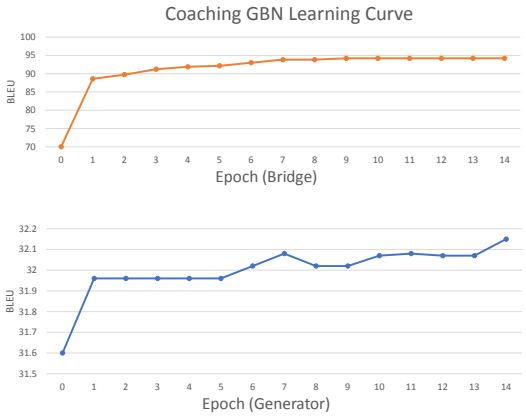


Figure 5: Coaching GBN's learning curve on IWSLT German-English Dev set.

Results The experimental results are summarized in Table 1. We can observe that our fine-tuned MLE baseline (29.10) is already over-

Methods	RG-1	RG-2	RG-L
ABS	29.55	11.32	26.42
ABS+	29.76	11.88	26.96
Luong-NMT	33.10	14.45	30.71
SAEASS	**36.15**	**17.54**	**33.63**
seq2seq+att	34.04	15.95	31.68
Uniform GBN ($\tau = 0.8$)	34.10	16.70	31.75
LM GBN ($\tau = 0.8$)	34.32	16.88	31.89
Coaching GBN ($\tau = 0.8$)	34.49	16.70	31.95
Coaching GBN ($\tau = 1.2$)	34.83	16.83	32.25
Coaching GBN ($\tau = 1.0$)	**35.26**	**17.22**	**32.67**

Table 2: Full length ROUGE F1 evaluation results on the English Gigaword test set used by (Rush et al., 2015). RG in the Table denotes ROUGE. Results for comparison are taken from SAEASS (Zhou et al., 2017).

competing other systems and our proposed GBN can yield a further improvement. We also observe that LM GBN and coaching GBN have both achieved better performance than Uniform GBN, which confirms that better regularization effects are achieved, and the generators become more robust and generalize better. We draw the learning curve of both the bridge and the generator in Figure 5 to demonstrate how they cooperate during training. We can easily observe the interaction between them: as the generator makes progress, the coaching bridge also improves itself to propose harsher targets for the generator to learn.

3.3 Abstractive Text Summarization

Dataset We follow the previous works by Rush et al. (2015); Zhou et al. (2017) and use the same corpus from Annotated English Gigaword dataset (Napoles et al., 2012). In order to be comparable, we use the same script [4] released by Rush et al. (2015) to pre-process and extract the training and validation sets. For the test set, we use the English Gigaword, released by Rush et al. (2015), and evaluate our system through ROUGE (Lin, 2004). Following previous works, we employ ROUGE-1, ROUGE-2, and ROUGE-L as the evaluation metrics in the reported experimental results.

[4]https://github.com/facebookarchive/NAMAS

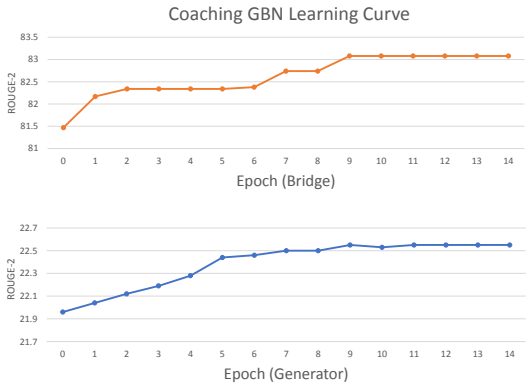


Figure 6: Coaching GBN's learning curve on Abstractive Text Summarization Dev set.

System Setting We follow Zhou et al. (2017); Rush et al. (2015) to set input and output vocabularies to 119,504 and 68,883 respectively, and we also set the word embedding size to 300 and all GRU hidden state size to 512. Then we adopt dropout (Srivastava et al., 2014) with probability $p = 0.5$ strategy in our output layer. We use attention-based sequence-to-sequence model (Bahdanau et al., 2014; Cho et al., 2014) as our baseline and reproduce the results of the baseline reported in Zhou et al. (2017). As stated, the attentive encoder-decode architecture can already outperform existing ABS/ABS+ systems (Rush et al., 2015). In coaching GBN, due to the fact that the input of abstractive summarization X contains more information than the summary target Y^*, directly training the bridge $p_\eta(Y|Y^*)$ to understand the generator $p_\theta(Y|X)$ is infeasible. Therefore, we re-design the coaching bridge to receive both source and target input X, Y and we enlarge its vocabulary size to 88,883 to encompass more information about the source side. In Uniform/LM GBN experiments, we also fix the hyper-parameter $\tau = 0.8$ as the optimal setting.

Results The experimental results are summarized in Table 2. We can observe a significant improvement via our GBN systems. Similarly, the coaching GBN system achieves the strongest performance among all, which again reflects our assumption that more sophisticated regularization can benefit generator's training. We draw the learning curve of the coaching GBN in Figure 6 to demonstrate how the bridge and the generator promote each other.

4 Analysis

By introducing different constraints into the bridge module, the bridge distribution will propose different training samples for the generator to learn. From Table 3, we can observe that most samples still reserve their original meaning. The uniform bridge simply performs random replacement without considering any linguistic constraint. The LM bridge strives to smooth reference sentence with high-frequent words. And the coaching bridge simplifies difficult expressions to relieve generator's learning burden. From our experimental results, the more rational and aggressive diversification from the coaching GBN clearly benefits generator the most and helps the generator generalize to more unseen scenarios.

System	**Uniform GBN**
Property	Random Replacement
Reference	the question is , is it worth it ?
Bridge	the question lemon , was it worth it ?
System	**Language-model GBN**
Property	Word Replacement
Reference	now how can this help us ?
Bridge	so how can this help us ?
System	**Coaching GBN**
Property	Reordering
Reference	i need to have a health care lexicon .
Bridge	i need a lexicon for health care .
Property	Simplification
Reference	this is the way that most of us were taught to tie our shoes .
Bridge	most of us learned to bind our shoes .

Table 3: Qualitative analysis for three different bridge distributions.

5 Related Literature

5.1 Data Augmentation and Self-training

In order to resolve the data sparsity problem in Neural Machine Translation (NMT), many works have been conducted to augment the dataset. The most popular strategy is via self-learning, which incorporates the self-generated data directly into training. Zhang and Zong (2016) and Sennrich et al. (2015) both use self-learning to leverage massive monolingual data for NMT training. Our bridge can take advantage of the parallel training data only, instead of external monolingual ones to synthesize new training data.

5.2 Reward Augmented Maximum Likelihood

Reward augmented maximum likelihood or RAML (Norouzi et al., 2016) proposes to integrate task-level reward into MLE training by using an exponentiated payoff distribution. KL divergence between the payoff distribution and the generator's output distribution are minimized to achieve an optimal task-level reward. Following this work, Ma et al. (2017) introduces softmax Q-Distribution to interpret RAML and reveals its relation with Bayesian decision theory. These two works both alleviate data sparsity problem by augmenting target examples based on the ground truth. Our method draws inspiration from them but seeks to propose the more general Generative Bridging Network, which can transform the ground truth into different bridge distributions, from where samples are drawn will account for different interpretable factors.

5.3 Coaching

Our coaching GBN system is inspired by imitation learning by coaching (He et al., 2012). Instead of directly behavior cloning the oracle, they advocate learning hope actions as targets from a coach which is interpolated between learner's policy and the environment loss. As the learner makes progress, the targets provided by the coach will become harsher to gradually improve the learner. Similarly, our proposed coaching GBN is motivated to construct an easy-to-learn bridge distribution which lies in between the ground truth and the generator. Our experimental results confirm its effectiveness to relieve the learning burden.

6 Conclusion

In this paper, we present the Generative Bridging Network (GBN) to overcome data sparsity and overfitting issues with Maximum Likelihood Estimation in neural sequence prediction. Our implemented systems prove to significantly improve the performance, compared with strong baselines. We believe the concept of bridge distribution can be applicable to a wide range of distribution matching tasks in probabilistic learning. In the future, we intend to explore more about GBN's applications as well as its provable computational and statistical guarantees.

References

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Semi-supervised learning for neural machine translation. *arXiv preprint arXiv:1606.04596* .

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In *Advances in Neural Information Processing Systems*. pages 577–585.

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

Kenji Doya. 1992. Bifurcations in the learning of recurrent neural networks. In *Circuits and Systems, 1992. ISCAS'92. Proceedings., 1992 IEEE International Symposium on*. IEEE, volume 6, pages 2777–2780.

He He, Jason Eisner, and Hal Daume. 2012. Imitation learning by coaching. In *Advances in Neural Information Processing Systems*. pages 3149–3157.

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

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*. Association for Computational Linguistics, pages 177–180.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*. Barcelona, Spain, volume 8.

Xuezhe Ma, Pengcheng Yin, Jingzhou Liu, Graham Neubig, and Eduard Hovy. 2017. Softmax q-distribution estimation for structured prediction: A theoretical interpretation for raml. *arXiv preprint arXiv:1705.07136* .

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*. Association for Computational Linguistics, pages 95–100.

Partha Niyogi, Federico Girosi, and Tomaso Poggio. 1998. Incorporating prior information in machine learning by creating virtual examples. *Proceedings of the IEEE* 86(11):2196–2209.

Mohammad Norouzi, Samy Bengio, Navdeep Jaitly, Mike Schuster, Yonghui Wu, Dale Schuurmans, et al. 2016. Reward augmented maximum likelihood for neural structured prediction. In *Advances In Neural Information Processing Systems*. pages 1723–1731.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 311–318.

Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548* .

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

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685* .

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709* .

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 2818–2826.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*. pages 2773–2781.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Sam Wiseman and Alexander M Rush. 2016. Sequence-to-sequence learning as beam-search optimization. *arXiv preprint arXiv:1606.02960* .

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*. pages 2048–2057.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

Jiacheng Zhang, Yang Liu, Huanbo Luan, Jingfang Xu, and Maosong Sun. 2017. Prior knowledge integration for neural machine translation using posterior regularization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 1514–1523.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. In *EMNLP*. pages 1535–1545.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. *arXiv preprint arXiv:1704.07073* .

Higher-order Syntactic Attention Network for Long Sentence Compression

Hidetaka Kamigaito[♡], **Katsuhiko Hayashi**[♢], **Tsutomu Hirao**[♠] **and Masaaki Nagata**[♠]

[♡] Institute of Innovative Research, Tokyo Institute of Technology
[♢] Institute of Scientific and Industrial Research, Osaka University
[♠] NTT Communication Science Laboratories, NTT Corporation

kamigaito@lr.pi.titech.ac.jp, katsuhiko-h@sanken.osaka-u.ac.jp,
{hirao.tsutomu,nagata.masaaki}@lab.ntt.co.jp

Abstract

Sentence compression methods based on LSTM can generate fluent compressed sentences. However, the performance of these methods is significantly degraded when compressing long sentences since it does not explicitly handle syntactic features. To solve this problem, we propose a higher-order syntactic attention network (HiSAN) that can handle higher-order dependency features as an attention distribution on LSTM hidden states. Furthermore, to avoid the influence of incorrect parse results, we train HiSAN by maximizing the probability of a correct output together with the attention distribution. Experiments on the Google sentence compression dataset show that our method achieved the best performance in terms of F_1 as well as ROUGE-1,2 and L scores, 83.2, 82.9, 75.8 and 82.7, respectively. In subjective evaluations, HiSAN outperformed baseline methods in both readability and informativeness.

1 Introduction

Sentence compression is the task of compressing long sentences into short and concise ones by deleting words. To generate compressed sentences that are grammatical, many researchers (Jing, 2000; Knight and Marcu, 2000; Berg-Kirkpatrick et al., 2011; Filippova and Altun, 2013) have adopted tree trimming methods. Even though Filippova and Altun (2013) reported the best results on this task, automatic parse errors greatly degrade the performances of these tree trimming methods.

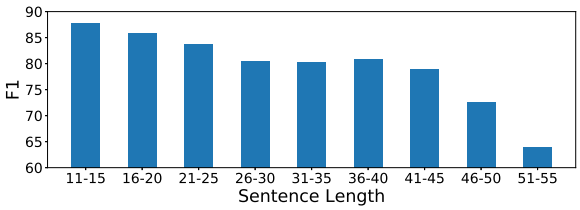


Figure 1: F_1 scores of LSTM-based sentence compression method for each sentence length.[1]

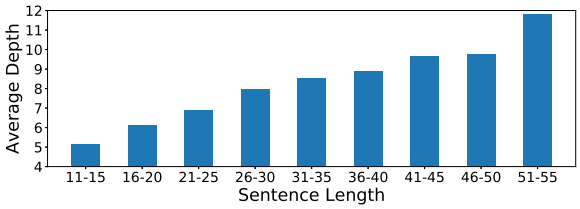


Figure 2: Average tree depths for each sentence length.[2]

Recently, Filippova et al. (2015) proposed an LSTM sequence-to-sequence (Seq2Seq) based sentence compression method that can generate fluent sentences without utilizing any syntactic features. Therefore, Seq2Seq based sentence compression is a promising alternative to tree trimming.

However, as reported for a machine translation task (Cho et al., 2014; Pouget-Abadie et al., 2014; Koehn and Knowles, 2017), the longer the input sentences are, the worse the Seq2Seq performances become. We also observed this problem in the sentence compression task. As shown in Figure 1, the performance of Seq2Seq is degraded when compressing long sentences. In particular, the performance significantly falls if sentence length exceeds 26 words. This is an important problem, because sentences longer than the average sentence length (=28 words) accounts for 42% of the Google sentence compression dataset.

As shown in Figure 2, long sentences have deep

[1] We used an LSTM-based sentence compression method (Filippova et al., 2015) in the evaluation setting as described in Section 4.1.

[2] We treat the maximum distance from root node to the leaf node as dependency tree depth.

Proceedings of NAACL-HLT 2018, pages 1716–1726
New Orleans, Louisiana, June 1 - 6, 2018.

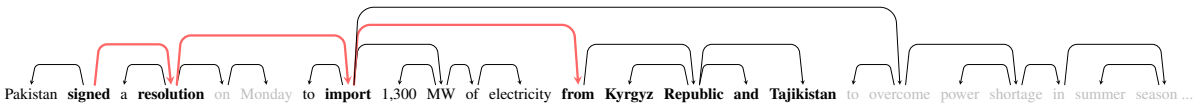


Figure 3: An example compressed sentence and its dependency tree. The words colored by gray represent deleted words.

dependency trees, which have long distances from root node to words at leaf nodes. Therefore, improving compression performance for sentences with such deep dependency trees can help to compress longer sentences.

To deal with sentences that have deep dependency trees, we focus on the chains of dependency relationships. Figure 3 shows an example of a compressed sentence with its dependency tree. The topic of this sentence is *import agreement related to electricity*. Thus, to generate informative compression, the compressed sentence must retain the *country name*. In this example, the compressed sentence should keep the phrase "from Kyrgyz Republic and Tajikistan". Thus, the compressed sentence must also keep the dependency chain "import", "resolution" and "signed" because the phrase is a child of this chain. By considering such higher-order dependency chains, the system can implement informative compression. As can be seen from the example in Figure 3, tracking a higher-order dependency chain for each word would help to compress long sentences. This paper refers to such dependency relationships by the expression "d-length dependency chains".

To handle a d-length dependency chain for sentence compression with LSTM, we propose the higher-order syntactic attention network (HiSAN). HiSAN computes the deletion probability for a given word based on the d-length dependency chain starting from the word. The d-length dependency chain is represented as an attention distribution, learned using automatic parse trees. To alleviate the influence of parse errors in automatic parse trees, we learn the attention distribution together with deletion probability.

Evaluation results on the Google sentence compression dataset (Filippova and Altun, 2013) show that HiSAN achieved the best F_1, ROUGE-1,2 and L scores 83.2, 82.9, 75.8 and 82.7, respectively. In particular, HiSAN attained remarkable compression performance with long sentences. In human evaluations, HiSAN also outperformed the baseline methods.

2 Baseline Sequence-to-Sequence Method

Sentence compression can be regarded as a tagging task, where given a sequence of input tokens $\mathbf{x} = (x_0, ..., x_n)$, a system assigns output label y_t, which is one of three types of specific labels ("keep,""delete," or"end of sentence") to each input token x_t $(1 \leq t \leq n)$.

The LSTM-based approaches for sentence compression are mostly based on the bi-LSTM based tagging method (Tagger) (Klerke et al., 2016; Wang et al., 2017; Chen and Pan, 2017) or Seq2Seq (Filippova et al., 2015; Tran et al., 2016). Tagger independently predicts labels in a point estimation manner, whereas Seq2Seq predicts labels by considering previously predicted labels. Since Seq2Seq is more expressive than Tagger, we built HiSAN on the baseline Seq2Seq model.

Our baseline Seq2Seq is a version of Filippova et al. (2015) extended through the addition of bi-LSTM, an input feeding approach (Vinyals et al., 2015; Luong et al., 2015), and a monotonic hard attention method (Yao and Zweig, 2015; Tran et al., 2016). As described in the evaluations section, this baseline achieved comparable or even better scores than the state-of-the-art scores reported in Filippova et al. (2015). The baseline Seq2Seq model consists of embedding, encoder, decoder, and output layers.

In the embedding layer, the input tokens $\mathbf{x}$ are converted to the embeddings $\mathbf{e}$. As reported in Wang et al. (2017), syntactic features are important for learning a generalizable embedding for sentence compression. Following their results, we also introduce syntactic features into the embedding layer. Specifically, we combine the surface token embedding w_i, POS embedding p_i, and dependency relation label embedding r_i into a single vector as follows:

$$e_i = [w_i, p_i, r_i], \quad (1)$$

where [] represents vector concatenation, and e_i is an embedding of token x_i.

The encoder layer converts the embedding $\mathbf{e}$ into a sequence of hidden states $\mathbf{h} = (h_0, ..., h_n)$

using a stacked bidirectional-LSTM (bi-LSTM) as follows:

$$h_i = \left[\overrightarrow{h_i}, \overleftarrow{h_i}\right] \quad (2)$$

$$\overrightarrow{h}_i = LSTM_{\overrightarrow{\theta}}(\overrightarrow{h}_{i-1}, e_i) \quad (3)$$

$$\overleftarrow{h}_i = LSTM_{\overleftarrow{\theta}}(\overleftarrow{h}_{i-1}, e_i), \quad (4)$$

where $LSTM_{\overrightarrow{\theta}}$ and $LSTM_{\overleftarrow{\theta}}$ represent forward and backward LSTM, respectively. The final state of the backward LSTM $\overleftarrow{h}_0$ is inherited by the decoder as its initial state.

In the decoder layer, the concatenation of a 3-bit one-hot vector which is determined by previously predicted label y_{t-1}, previous final hidden state d_{t-1} (explained later), and the input embedding of x_t, is encoded into the decoder hidden state $\overrightarrow{s}_t$ using stacked forward LSTMs.

Contrary to the original softmax attention method, we can deterministically focus on one encoder hidden state h_t (Yao and Zweig, 2015) to predict y_t in the sentence compression task (Tran et al., 2016).[3]

In the output layer, label probability is calculated as follows:

$$P(y_t \mid y_{<t}, \mathbf{x}) = softmax(W_o \cdot d_t) \cdot \delta_{y_t}, \quad (5)$$

$$d_t = [h_t, \overrightarrow{s}_t] \quad (6)$$

where W_o is a weight matrix of the softmax layer and δ_{y_t} is a binary vector where the y_t-th element is set to 1 and the other elements to 0.

3 Higher-order Syntactic Attention Network

The key component of HiSAN is its attention module. Unlike the baseline Seq2Seq, HiSAN employs a packed d-length dependency chain as distributions in the attention module. Section 3.1 explains the packed d-length dependency chain. Section 3.2 describes the network structure of our attention module, and Section 3.3 explains the learning method of HiSAN.

3.1 Packed d-length Dependency Chain

The probability for a packed d-length dependency chain is obtained from a dependency graph, which is an edge-factored dependency score matrix (Hashimoto and Tsuruoka, 2017; Zhang et al., 2017). First, we explain the dependency graph. Figure 4 (a) shows an example of the dependency graph. HiSAN represents a dependency graph as an attention distribution generated by the attention module. A probability for each dependency edge is obtained from the attention distribution.

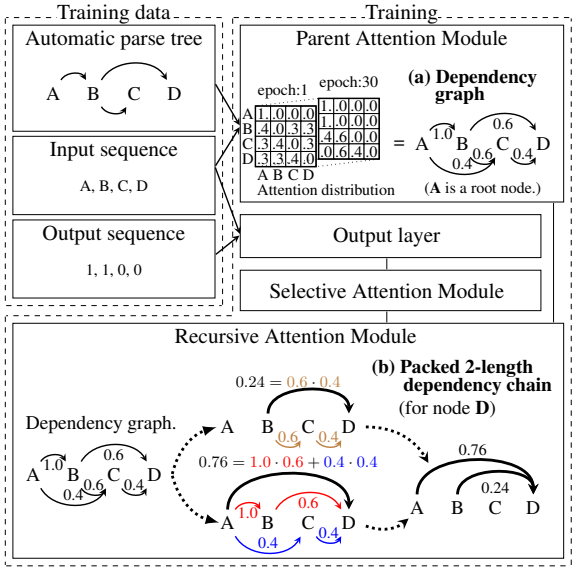


Figure 4: Dependency structures used in our higher-order syntactic attention network.

Figure 4 (b) shows an example of the packed d-length dependency chain. With our recursive attention module, the probability for a packed d-length dependency chain is computed as the sum of probabilities for each path yielded by recursively tracking from a word to its d-th ancestor. The probability for each path is calculated as the product of the probabilities of tracked edges. The probability for the chain can represent several d-length dependency chains compactly, and so alleviates the influence of incorrect parse results. This is the advantage of using dependency graphs.

3.2 Network Architecture

Figure 5 shows the prediction process of HiSAN. In this figure, HiSAN predicts output label y_7 from the input sentence. The prediction process of HiSAN is as follows.

1. **Parent Attention** module calculates $P_{parent}(x_j|x_t, \mathbf{x})$, the probability of x_j being the parent of x_t, by using h_j and h_t. This probability is calculated for all pairs of x_j, x_t. The arc in Figure 5 shows the most probable dependency parent for each child token.

[3] This is because the output length is the same as the input length, and each x_t can be assigned to each y_t in a one-to-one correspondence.

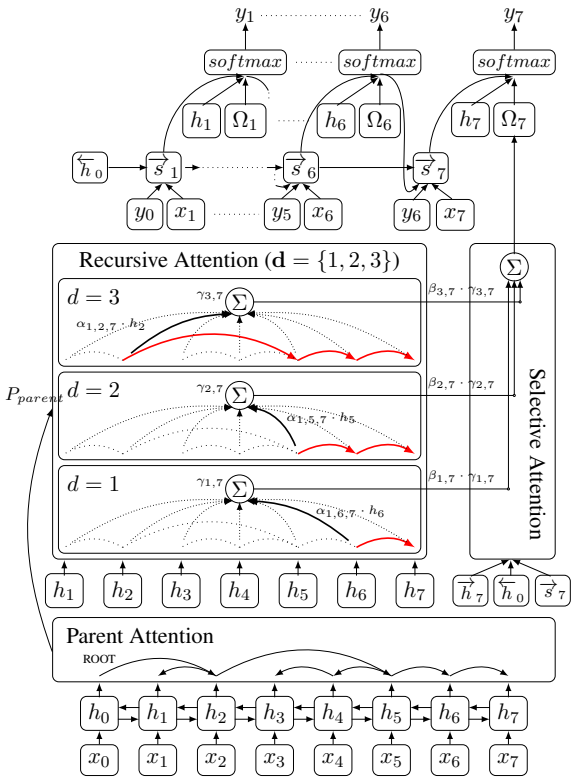


Figure 5: Prediction process of our higher-order syntactic attention network.

2. **Recursive Attention** module calculates $\alpha_{d,t,j}$, the probability of x_j being the d-th order parent (d denotes the chain length) of x_t, by recursively using $P_{parent}(x_j|x_t, \mathbf{x})$. $\alpha_{d,t,j}$ is also treated as an attention distribution, and used to calculate $\gamma_{d,t}$, the weighted sum of $\mathbf{h}$ for each length d. For example, a 3-length dependency chain of word x_7 with highest probability is x_6-x_5-x_2. The encoder hidden states h_6, h_5 and h_2, which correspond to the dependency chain, are weighted by calculated parent probabilities $\alpha_{1,7,6}$, $\alpha_{2,7,5}$ and $\alpha_{3,7,2}$, respectively, and then fed to the selective attention module.

3. **Selective Attention** module calculates weight $\beta_{d,t}$ from its length, $d \in \mathbf{d}$, for each $\gamma_{d,t}$. $\mathbf{d}$ represents a group of chain lengths. $\beta_{d,t}$ is calculated by encoder and decoder hidden states. Each $\beta_{d,t} \cdot \gamma_{d,t}$ is summed to Ω_t, the output of the selective attention module.

4. Finally, the calculated Ω_t is concatenated and input to the output layer.

Details of each module are explained in the following subsection.

3.2.1 Parent Attention Module

Zhang et al. (2017) formalized dependency parsing as the problem of independently selecting the parent of each word in a sentence. They produced a distribution over possible parents for each child word by using the attention layer on bi-LSTM hidden layers.

In a dependency tree, a parent has more than one child. Under this constraint, dependency parsing is represented as follows. Given sentence $S = (x_0, x_1, ..., x_n)$, the parent of x_j is selected from $S \setminus x_i$ for each token $S \setminus x_0$. Note that x_0 denotes the root node. The probability of token x_j being the parent of token x_t in sentence $\mathbf{x}$ is calculated as follows:

$$P_{parent}(x_j|x_t, \mathbf{x}) = softmax(g(h_{j'}, h_t)) \cdot \delta_{x_j}, \quad (7)$$

$$g(h_{j'}, h_t) = v_a^T \cdot tanh(U_a \cdot h_{j'} + W_a \cdot h_t), \quad (8)$$

where v_a, U_a and W_a are weight matrices of g.

Different from the attention based dependency parser, $P_{parent}(x_j|x_t, \mathbf{x})$ is jointly learned with output label probability $P(\mathbf{y} \mid \mathbf{x})$ in the training phase. Training details are given in Section 3.3.

3.2.2 Recursive Attention Module

The recursive attention module recursively calculates $\alpha_{d,t,j}$, the probability of x_j being the d-th order parent of x_t, as follows:

$$\alpha_{d,t,j} = \begin{cases} \sum_{k=1}^{n} \alpha_{d-1,t,k} \cdot \alpha_{1,k,j} & (d > 1) \\ P_{parent}(x_j|x_t, \mathbf{x}) & (d = 1) \end{cases}. \quad (9)$$

Furthermore, in a dependency parse tree, *root* should not have any parent, and a token should not depend on itself. In order to satisfy these rules, we impose the following constraints on $\alpha_{1,t,j}$:

$$\alpha_{1,t,j} = \begin{cases} 1 & (t = 0 \wedge j = 0) \\ 0 & (t = 0 \wedge j > 0) \\ 0 & (t \neq 0 \wedge t = j) \end{cases} \quad (10)$$

The 1st and 2nd lines of Eq. (10) represent the case that the parent of *root* is also *root*. These constraints imply that *root* does not have any parent. The 3rd line of Eq. (10) prevents a token from depending on itself. Because the 1st line of Eq. (9) is similar to the definition of matrix multiplication, Eq. (9) can be efficiently computed on a CPU and GPU[4].

[4]In training, HiSAN with 1- and 3-length dependency chains took 25 and 26 minutes, respectively, per epoch on an Intel Xeon E5-2697 v3 2.60 GHz.

By recursively using the single attention distribution, it is no longer necessary to prepare additional attention distributions for each order when computing the probability of higher order parents. Furthermore, since it is not necessary to learn multiple attention distributions, it becomes unnecessary to use hyper parameters for adjusting the weight of each distribution in training. Finally, this method can avoid the problem of sparse higher-order dependency relations in the training dataset.

The above calculated $\alpha_{d,t,j}$ is used to weight the bi-LSTM hidden layer $\mathbf{h}$ as follows:

$$\gamma_{d,t} = \sum_{k=j}^{n} \alpha_{d,t,j} \cdot h_j. \quad (11)$$

Note that $\gamma_{d,t}$ is inherited by the selective attention module, as explained in the next section.

3.2.3 Selective Attention Module

To select suitable dependency orders of the input sentence, the selective attention module weights and sums the hidden states $\gamma_{d,t}$ to Ω_t by using weighting parameter $\beta_{d,t}$, according to the current context as follows:

$$\beta_{d,t} = softmax(W_c \cdot c_t) \cdot \delta_d, \quad (12)$$

$$\Omega_t = \sum_{d \in \{0, \mathbf{d}\}} \beta_{d,t} \cdot \gamma_{d,t}, \quad (13)$$

where W_c is the weight matrix of the softmax layer, $\mathbf{d}$ is a group of chain lengths, c_t is a vector representing the current context, $\gamma_{0,t}$ is a zero-vector, and $\beta_{0,t}$ indicates the weight when the method does not use the dependency features. Context vector c_t is calculated as $c_t = [\overleftarrow{h}_0, \overrightarrow{h}_n, \overrightarrow{s}_t]$ using the current decoder hidden state $\overrightarrow{s}_t$.

The calculated Ω_t is concatenated and input to the output layer. In detail, d_t in Eq. (5) is replaced by concatenated vector $d'_t = [h_t, \Omega_t, \overrightarrow{s}_t]$; furthermore, instead of d_t, d'_t is also fed to the input of the decoder LSTM at $t+1$.

3.3 Objective Function

To alleviate the influence of parse errors, we jointly update the 1st-order attention distribution $\alpha_{1,t,k}$ and label probability $P(\mathbf{y}|\mathbf{x})$ (Kamigaito et al., 2017). The 1st-order attention distribution is learned by dependency parse trees. If $a_{t,j} = 1$ is an edge between parent word w_j and child w_t on a dependency tree ($a_{t,j} = 0$ denotes that w_j is not a parent of w_t.), the objective function of our method can be defined as:

$$-logP(\mathbf{y}|\mathbf{x}) - \lambda \cdot \sum_{j=1}^{n} \sum_{t=1}^{n} a_{t,j} \cdot log\alpha_{1,t,j}, \quad (14)$$

where λ is a hyper-parameter that controls the importance of the output labels and parse trees in the training dataset.

4 Evaluations

4.1 Evaluation Settings

4.1.1 Dataset

This evaluation used the Google sentence compression dataset (Filippova and Altun, 2013)[5]. This dataset contains information of compression labels, part-of-speech (POS) tags, dependency parents and dependency relation labels for each sentence.

We used the first and last 1,000 sentences of `comp-data.eval.json` as our test and development datasets, respectively. Note that our test dataset is compatible wth that used in previous studies (Filippova et al., 2015; Tran et al., 2016; Klerke et al., 2016; Wang et al., 2017).

In this paper, we trained the following baselines and HiSAN on all sentences of `sent-comp.train*.json` (total 200,000 sentences)[6,7,8].

In our experiments, we replaced rare words that appear fewer than 10 times in our training dataset with a special token ⟨UNK⟩. After this filtering, the input vocabulary size was 23,168.

4.1.2 Baseline Methods

For a fair comparison of HiSAN, we used the input features described in Eq. (1) for the following baseline methods:

[5]https://github.com/google-research-datasets/sentence-compression

[6]Note that Filippova et al. (2015) used 2,000,000 sentences for training their method, but these datasets are not publicly available.

[7]We also demonstrate an experimental evaluation on a small training set (total 8,000 sentences), that was used in previous research. The results of this setting are listed in our supplemental material.

[8]Note that the large training dataset lacks periods at the end of compressed sentences. To unify the form of compressed sentences in small and large settings, we added periods to the end of compressed sentences in the large training dataset.

Tagger: A method that regards sentence compression as a tagging task based on bi-LSTM (Klerke et al., 2016; Wang et al., 2017).

Tagger+ILP: An extension of **Tagger** that integrates ILP (Integer Linear Programming)-based dependency tree trimming (Wang et al., 2017). We set their positive parameter λ to 0.2.

Bi-LSTM: A method that regards sentence compression as a sequence-to-sequence translation task proposed by (Filippova et al., 2015). For a fair comparison, we replaced their one-directional LSTM with the more expressive bi-LSTM in the encoder part. The initial state of the decoder is set to the sum of the final states of the forward and backward LSTMs.

Bi-LSTM-Dep: An extension of Bi-LSTM that exploits features obtained from a dependency tree (named LSTM-PAR-PRES in Filippova et al. (2015)). Following their work, we fed the word embedding and the predicted label of a dependency parent word to the current decoder input of **Bi-LSTM**.

Base: Our baseline Seq2Seq method described in Section 2.

Attn: An extension of the softmax based attention method (Luong et al., 2015). We replaced h_t in Eq. (6) with the weighted sum calculated by the commonly used concat attention (Luong et al., 2015).

HiSAN-Dep: A variant of HiSAN that utilizes the pipeline approach. We fix $\alpha^{1,j,t}$ to 1.0 if x_j is a parent of x_t in the input dependency parse tree, 0.0 otherwise. In this baseline, $\mathbf{d} = \{1\}$ was used.

4.1.3 Training Details

Following the previous work (Wang et al., 2017), the dimensions of the word embeddings, LSTM layers, and attention layer were set to 100. For the Tagger-style methods, the depth of the LSTM layer was set to 3, and for the Seq2Seq-style methods, the depth of the LSTM layer was set to 2. In this setting, all methods have a total of six LSTM-layers. The dimensions of POS and the dependency-relation label embeddings were set to 40. All parameters were initialized by Glorot and Bengio (2010)'s method. For all methods, we applied Dropout (Srivastava et al., 2014) to the input of the LSTM layers. All dropout rates were set to 0.3.

During training, the learning rate was tuned with Adam (Kingma and Ba, 2014). The initial learning rate was set to 0.001. The maximum number of training epochs was set to 30. The hyper-parameter λ was set to 1.0 in the supervised attention setting. All gradients were averaged in each mini-batch. The maximum mini-batch size was set to 16. The order of mini-batches was shuffled at the end of each training epoch. The clipping threshold of the gradient was set to 5.0. We selected trained models with early stopping based on maximizing per-sentence accuracy (i.e., how many compressions could be fully reproduced) of the development data set.

To obtain a compressed sentence, we used greedy decoding, rather than beam decoding, as the latter attained no gain in the development dataset. All methods were written in C++ on Dynet (Neubig et al., 2017).

4.2 Automatic Evaluation

In the automatic evaluation, we used token level F_1-measure ($\mathbf{F_1}$) as well as recall of ROUGE-1, ROUGE-2 and ROUGE-L (Lin and Och, 2004)[9] as evaluation measures. We used $\Delta C = \mathit{system\ compression\ ratio} - \mathit{gold\ compression\ ratio}$ to evaluate how close the compression ratio of system outputs was to that of gold compressed sentences. The average compression ratio of the gold compression for input sentence was 39.8. We used micro-average for F_1-measure and compression ratio[10], and macro-average for ROUGE scores, respectively.

To verify the benefits of our methods on long sentences, we additionally report scores on sentences longer than the average sentence length ($= 28$) in the test set. The average compression ratio of the gold compression for longer input sentences was 31.4.

All results are reported as the average scores of five trials. In each trial, different random choices were used to generate the initial values of the embeddings and the order of mini-batch processing.

Table 1 shows the results. HiSANs outperformed the other methods. In particular, HiSAN ($d = \{1, 2, 4\}$) achieved the best score on F_1,

[9] We used the ROUGE-1.5.5 script with option "-n 2 -m -d -a".

[10] We also report the macro-average of F_1-measure and compression ratio in our supplemental material.

[11] Note that we used average of all metrics to decide the best score of the development dataset. The results are listed in our supplemental material.

		ALL					LONG				
		$\mathbf{F_1}$	ROUGE			ΔC	$\mathbf{F_1}$	ROUGE			ΔC
			1	*2*	*L*			*1*	*2*	*L*	
Tagger		82.8	81.1	72.4	80.9	-3.0	80.4	78.7	69.7	78.4	-2.8
Tagger+ILP		79.0	76.1	64.6	75.8	-4.1	74.3	73.7	62.1	73.2	-3.6
Bi-LSTM		81.9	81.1	73.7	80.9	-2.2	78.9	78.4	70.4	78.0	-2.1
Bi-LSTM-Dep		82.3	81.5	74.1	81.3	-2.1	79.6	78.9	71.0	78.5	-1.9
Attn		82.4	81.6	74.3	81.4	-2.3	79.8	79.4	71.4	78.7	-2.2
Base		82.7	81.9	74.7	81.7	-2.4	80.1	79.1	71.7	79.0	-2.3
HiSAN-Dep	($\mathbf{d} = \{1\}$)	82.7	82.1	74.9	81.9	-2.2	80.1	79.7	72.0	79.3	-1.9
HiSAN-Dep	($\mathbf{d} = \{1,2\}$)	82.7	81.9	74.6	81.7	-2.4	80.5	79.9	72.3	79.5	-2.2
	($\mathbf{d} = \{1,2,3\}$)	83.1	82.0	74.9	81.8	-2.5	80.5	79.5	71.9	79.2	-2.3
	($\mathbf{d} = \{1,2,4\}$)$*$	82.9	82.1	74.9	81.9	-2.4	80.4	79.7	72.1	79.4	-2.1
	($\mathbf{d} = \{1,2,3,4\}$)	82.7	82.1	74.8	81.9	-2.2	80.1	79.6	71.9	79.3	-2.0
HiSAN	($\mathbf{d} = \{1\}$)	83.0	81.7	74.5	81.5	-2.9	80.6	79.6	72.1	79.3	-2.5
	($\mathbf{d} = \{1,2\}$)	83.1	82.3	75.1	82.2	-2.1	**80.9**	80.5	72.8	80.2	-1.9
	($\mathbf{d} = \{1,2,3\}$)	82.9	82.5	75.2	82.1	-2.1	80.7	80.2	72.6	79.9	-2.1
	($\mathbf{d} = \{1,2,4\}$)$*$	**83.2**	**82.9**	**75.8**	**82.7**	**−1.7**	**80.9**	**80.6**	**73.2**	**80.3**	**−1.8**
	($\mathbf{d} = \{1,2,3,4\}$)	82.7	82.0	74.7	81.8	-2.3	80.6	79.8	72.6	79.5	-2.3

Table 1: Results of automatic evaluation. **ALL** and **LONG** represent, respectively, the results in all sentences and long sentences (longer than average length 28) in the test dataset. $\mathbf{d}$ represents the groups of d-length dependency chains. $*$ indicates the model that achieved the best score among the same methods with different $\mathbf{d}$ in the development dataset[11]. Bold values indicate the best scores.

ROUGE, and ΔC in all settings. The F_1 scores of HiSAN (**ALL**) were higher than the current state-of-the-art score of .82, reported by Filippova et al. (2015). The improvements in F_1 and ROUGE scores from the baselines methods in the **LONG** setting are larger than those in the **ALL** setting. ¿From these results, we can conclude that d-length dependency chains are effective for sentence compression, especially in the case of longer than average sentences. HiSAN ($d = \{1\}$) outperformed **HiSAN-Dep** in F_1 scores in **ALL** and **LONG** settings. This result shows the effectiveness of joint learning the dependency parse tree and the output labels.

4.3 Human Evaluation

In the human evaluation, we compared the baselines with our method, which achieved the highest F_1 score in the automatic evaluations. We used the first 100 sentences that were longer than the average sentence length ($= 28$) in the test set for human evaluation. Similar to Filippova et al. (2015), the compressed sentence was rated by five raters who were asked to select a rating on a five-point Likert scale, ranging from one to five for readability (**Read**) and for informativeness (**Info**). We report the average of these scores from the five raters. To investigate the differences between the methods, we also compared the baseline methods and HiSAN using the sentences for which the methods yielded different compressed sentences.

		Read	Info	CR
All *(100)*	Tagger	4.54	3.41	30.9
	Base	4.64	3.45	31.1
	HiSAN (d=$\{1,2,4\}$)	**4.68**	**3.52**	31.6
Diff *(41)*	Base	4.79	3.46	29.4
	HiSAN (d=$\{1,2,4\}$)	**4.89**	**3.64**	30.6

Table 2: Results of human evaluations. *All* denotes results for all sentences in the test set, and *Diff* denotes results for the sentences for which the methods yielded different compressed sentences. Parentheses ( ) denote sentence size. **CR** denotes the compression ratio. The average gold compression ratio for input sentence in *All* and *Diff* were 32.1 and 31.5, respectively. Other notations are similar to those in Table 1.

Table 2 shows the results. HiSAN ($d = \{1,2,4\}$) achieved better results than the baselines in terms of both readability and informativeness. The results agree with those obtained from the automatic evaluations. ¿From the results on the sentences whose compressed sentences were different between Base and HiSAN ($d = \{1,2,4\}$), we can clearly observe the improvement attained by HiSAN (d=$\{1,2,4\}$) in informativeness.

Input	Pakistan signed a resolution on Monday to import 1,300 MW of electricity from Kyrgyz Republic and Tajikistan to overcome power shortage in summer season, said an official press release .
Gold	Pakistan signed a resolution to import 1,300 MW of electricity from Kyrgyz Republic and Tajikistan .
Tagger	Pakistan signed a resolution to import 1,300 MW of electricity Tajikistan to overcome shortage .
Tagger-ILP	Pakistan signed resolution to import MW said .
Base	Pakistan signed a resolution to import 1,300 MW of electricity .
HiSAN-Dep ($d=\{1\}$)	Pakistan signed a resolution to import 1,300 MW of electricity .
HiSAN ($d=\{1,2,4\}$)	Pakistan signed a resolution to import 1,300 MW of electricity from Kyrgyz Republic and Tajikistan .
Input	US whistleblower Bradley Manning , charged with releasing over 700,000 battlefield reports from Iraq and Afghanistan to Wikileaks , received a sentence of 35 years in prison from a military court Wednesday .
Gold	Bradley Manning received a sentence of 35 years in prison .
Tagger	Bradley Manning received a sentence of 35 years .
Tagger-ILP	Bradley Manning received a sentence of years .
Base	Bradley Manning received a sentence of 35 years .
HiSAN-Dep ($d=\{1\}$)	Bradley Manning charged with releasing over 700,000 battlefield reports to Wikileaks received .
HiSAN ($d=\{1,2,4\}$)	Bradley Manning received a sentence of 35 years in prison .

Table 3: Example sentences and compressions.

5 Analysis

Table 3 shows examples of source sentences and their compressed variants output by baseline and HiSAN ($d = \{1, 2, 4\}$).

For both examples, the compressed sentence output by **Base** is grammatically correct. However, the informativeness is inferior to that attained by HiSAN ($d = \{1, 2, 4\}$). The compressed sentence output by **HiSAN-Dep** in the second example lacks both readability and informativeness. We believe that this compression failure is caused by incorrect parse results, because **HiSAN-Dep** employs the features obtained from the dependency tree in the pipeline procedure.

As reported in recent papers (Klerke et al., 2016; Wang et al., 2017), the F_1 scores of **Tagger** match or exceed those of the Seq2Seq-based methods. The compressed sentence of the first example in Table 3 output by **Tagger** is ungrammatical. We believe that this is mainly because **Tagger** cannot consider the predicted labels of the previous words. **Tagger-ILP** outputs grammatically incorrect compressed sentences in both examples. This result indicates that THE ILP constraint based on the parent-child relationships between words is insufficient to generate fluent sentences.

Compared with these baselines, HiSAN ($d = \{1, 2, 4\}$) output compressed sentences that were fluent and had higher informativeness. This observation, which confirmed our expectations, is supported by the automatic and human evaluation results.

	$\mathbf{F_1}$	ROUGE			ΔC
		1	*2*	*L*	
Tagger	**82.8**	80.6	72.2	80.3	-3.2
Tagger+ILP	77.5	74.7	64.1	74.3	-4.6
Bi-LSTM	81.3	80.4	73.3	80.1	-2.2
Bi-LSTM-Dep	81.5	80.7	73.5	80.3	-2.1
Attn	81.9	81.0	73.9	80.6	-2.3
Base	82.1	81.0	73.9	80.7	-2.5
HiSAN-Dep					
$\mathbf{d} = \{1\}$	82.3	81.2	74.3	80.9	-2.4
$\mathbf{d} = \{1, 2\}$	81.9	80.8	73.8	80.4	-2.6
$\mathbf{d} = \{1, 2, 3\}$	82.6	81.2	74.3	80.9	-2.6
$\mathbf{d} = \{1, 2, 4\}$	82.0	80.7	73.7	80.4	-2.7
$\mathbf{d} = \{1, 2, 3, 4\}$	82.1	81.0	74.1	80.7	-2.5
HiSAN					
$\mathbf{d} = \{1\}$	82.7	81.4	74.5	81.1	-2.8
$\mathbf{d} = \{1, 2\}$	82.6	81.8	74.9	81.5	-2.1
$\mathbf{d} = \{1, 2, 3\}$	82.6	81.8	74.9	81.5	-2.3
$\mathbf{d} = \{1, 2, 4\}$	**82.8**	**82.2**	**75.5**	**82.0**	**−2.0**
$\mathbf{d} = \{1, 2, 3, 4\}$	82.4	81.3	74.4	81.0	-2.4

Table 4: Results of automatic evaluation using sentences with deep dependency trees (deeper than average depth 8). Bold results indicate the best scores.

We confirm that the compression performance of HiSAN actually improves if the sentences have deep dependency trees. Table 4 shows the automatic evaluation results for sentences with deep dependency trees. We can observe that HiSAN

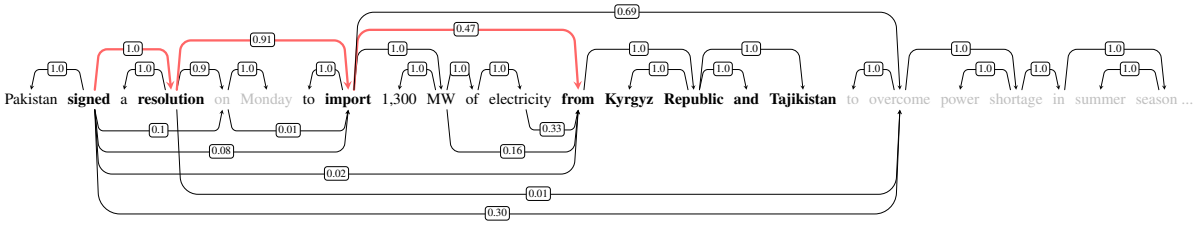


Figure 6: An example compressed sentence and its dependency graph of HiSAN $\mathbf{d} = \{1, 2, 4\}$. The words colored by gray represent deleted words. The numbers for each arc represent a probabilistic weight of a relationship between a parent and child words. The arcs contained in the parsed dependency tree are located on the top side. The arcs not contained in the parsed dependency tree are located on the bottom side.

with higher-order dependency chains has better compression performance if the sentences have deep dependency trees.

Figure 6 shows a compressed sentence and its dependency graph as determined by HiSAN $\mathbf{d} = \{1, 2, 4\}$. Almost all arcs with large probabilistic weights are contained in the parsed dependency trees. Interestingly, some arcs not contained in the parsed dependency trees connecting words which are connected by the dependency chains in the parsed dependency tree (colored by red). Considering the training dataset does not contain such dependency relationships, we can estimate that these arcs are learned in support of compressing sentences. This result meets our expectation that the dependency chain information is necessary for compressing sentences accurately.

6 Related Work

Several neural network based methods for sentence compression use syntactic features. Filippova et al. (2015) employs the features obtained from automatic parse trees in the LSTM-based encoder-decoder in a pipeline manner. Wang et al. (2017) trims dependency trees based on the scores predicted by an LSTM-based tagger. Although these methods can consider dependency relationships between words, the pipeline approach and the 1st-order dependency relationship fail to compress longer than average sentences.

Several recent machine translation studies also utilize syntactic features in Seq2Seq models. Eriguchi et al. (2017); Aharoni and Goldberg (2017) incorporate syntactic features of the target language in the decoder part of Seq2Seq. Both methods outperformed Seq2Seq without syntactic features in terms of translation quality. However, both methods fail to provide an entire parse tree until the decoding phase is finished. Thus, these methods cannot track all possible parents for each word within the decoding process. Similar to HiSAN, Hashimoto and Tsuruoka (2017) use dependency features as attention distributions, but different from HiSAN, they use pre-trained dependency relations, and do not take into account the chains of dependencies. Marcheggiani and Titov (2017); Bastings et al. (2017) consider higher-order dependency relationships in Seq2Seq by incorporating a graph convolution technique (Kipf and Welling, 2016) into the encoder. However, the dependency information of the graph convolution technique is still given in pipeline manner.

Unlike the above methods, HiSAN can capture higher-order dependency features using d-length dependency chains without relying on pipeline processing.

7 Conclusion

In this paper, we incorporated higher-order dependency features into Seq2Seq to compress sentences of all lengths.

Experiments on the Google sentence compression test data showed that our higher-order syntactic attention network (HiSAN) achieved the better performance than baseline methods on F_1 as well as ROUGE-1,2 and L scores 83.2, 82.9, 75.8 and 82.7, respectively. Of particular importance, challenged with longer than average sentences, HiSAN outperformed the baseline methods in terms of F_1, ROUGE-1,2 and L scores. Furthermore, HiSAN also outperformed the previous methods for both readability and informativeness in human evaluations.

From the evaluation results, we conclude that HiSAN is an effective tool for the sentence compression task.

References

Roee Aharoni and Yoav Goldberg. 2017. Towards string-to-tree neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 132–140. http://aclweb.org/anthology/P17-2021.

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Simaan. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 1957–1967. https://www.aclweb.org/anthology/D17-1209.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, pages 481–490. http://www.aclweb.org/anthology/P11-1049.

Yanju Chen and Rong Pan. 2017. Automatic emphatic information extraction from aligned acoustic data and its application on sentence compression. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*. pages 3422–3428.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Association for Computational Linguistics, Doha, Qatar, pages 103–111. http://www.aclweb.org/anthology/W14-4012.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 72–78. http://aclweb.org/anthology/P17-2012.

Katja Filippova, Enrique Alfonseca, Carlos A. Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 360–368. http://aclweb.org/anthology/D15-1042.

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 1481–1491. http://www.aclweb.org/anthology/D13-1155.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. pages 249–256.

Kazuma Hashimoto and Yoshimasa Tsuruoka. 2017. Neural machine translation with source-side latent graph parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 125–135. https://www.aclweb.org/anthology/D17-1012.

Hongyan Jing. 2000. Sentence reduction for automatic text summarization. In *Proceedings of the Sixth Conference on Applied Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 310–315. https://doi.org/10.3115/974147.974190.

Hidetaka Kamigaito, Katsuhiko Hayashi, Tsutomu Hirao, Hiroya Takamura, Manabu Okumura, and Masaaki Nagata. 2017. Supervised attention for sequence-to-sequence constituency parsing. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Asian Federation of Natural Language Processing, Taipei, Taiwan, pages 7–12. http://www.aclweb.org/anthology/I17-2002.

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

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

Sigrid Klerke, Yoav Goldberg, and Anders Søgaard. 2016. Improving sentence compression by learning to predict gaze. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 1528–1533. http://www.aclweb.org/anthology/N16-1179.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization-step one: Sentence compression. *AAAI/IAAI* 2000:703–710.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*. Association for Computational Linguistics, Vancouver, pages 28–39. http://www.aclweb.org/anthology/W17-3204.

Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*. Barcelona, Spain, pages 605–612. https://doi.org/10.3115/1218955.1219032.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1412–1421. http://aclweb.org/anthology/D15-1166.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 1506–1515. https://www.aclweb.org/anthology/D17-1159.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980* .

J. Pouget-Abadie, D. Bahdanau, B. van Merrienboer, K. Cho, and Y. Bengio. 2014. Overcoming the Curse of Sentence Length for Neural Machine Translation using Automatic Segmentation. *ArXiv e-prints* .

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

Nhi-Thao Tran, Viet-Thang Luong, Ngan Luu-Thuy Nguyen, and Minh-Quoc Nghiem. 2016. Effective attention-based neural architectures for sentence compression with bidirectional long short-term memory. In *Proceedings of the Seventh Symposium on Information and Communication Technology*. ACM, New York, NY, USA, SoICT '16, pages 123–130. https://doi.org/10.1145/3011077.3011111.

Oriol Vinyals, Ł ukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, Curran Associates, Inc., pages 2773–2781. http://papers.nips.cc/paper/5635-grammar-as-a-foreign-language.pdf.

Liangguo Wang, Jing Jiang, Hai Leong Chieu, Chen Hui Ong, Dandan Song, and Lejian Liao. 2017. Can syntax help? improving an lstm-based sentence compression model for new domains. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1385–1393. http://aclweb.org/anthology/P17-1127.

Kaisheng Yao and Geoffrey Zweig. 2015. Sequence-to-sequence neural net models for grapheme-to-phoneme conversion. In *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015*. pages 3330–3334. http://www.isca-speech.org/archive/interspeech_2015/i15_3330.html.

Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. 2017. Dependency parsing as head selection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, Valencia, Spain, pages 665–676. http://www.aclweb.org/anthology/E17-1063.

Neural Storyline Extraction Model for Storyline Generation from News Articles

Deyu Zhou[†] **Linsen Guo**[†] **Yulan He**[§]
[†] School of Computer Science and Engineering, Key Laboratory of Computer Network and Information Integration, Ministry of Education, Southeast University, China
[§] School of Engineering and Applied Science, Aston University, UK
{d.zhou,guolinsen}@seu.edu.cn, y.he@cantab.net

Abstract

Storyline generation aims to extract events described on news articles under a certain topic and reveal how those events evolve over time. Most existing approaches first train supervised models to extract events from news articles published in different time periods and then link relevant events into coherent stories. They are domain dependent and cannot deal with unseen event types. To tackle this problem, approaches based on probabilistic graphic models jointly model the generations of events and storylines without annotated data. However, the parameter inference procedure is too complex and models often require long time to converge. In this paper, we propose a novel neural network based approach to extract structured representations and evolution patterns of storylines without using annotated data. In this model, title and main body of a news article are assumed to share the similar storyline distribution. Moreover, similar documents described in neighboring time periods are assumed to share similar storyline distributions. Based on these assumptions, structured representations and evolution patterns of storylines can be extracted. The proposed model has been evaluated on three news corpora and the experimental results show that it outperforms state-of-the-art approaches accuracy and efficiency.

1 Introduction

With the development of the internet, massive information about current events is generated and propagated continuously on online news media sites. It is difficult for the public to digest such large volumes of information effectively. Storyline generation, aiming at summarizing the development of certain related events, has been intensively studied recently (Diao and Jiang, 2014).

In general, storyline can be considered as an event cluster where event-related news articles are ordered and clustered depending on both content and temporal similarity. Different ways of calculating content and temporal similarity can be used to cluster related events (Yan et al., 2011; Huang and Huang, 2013). Bayesian non-parametric models could also be used to tackle this problem by describing the storyline generating process using probabilistic graphical models (Li and Cardie, 2014; Diao and Jiang, 2014). Nevertheless, most existing approaches extract events independently and link relevant events in a post-processing step. More recently, Zhou et al. (2016) proposed a non-parametric generative model to extract storylines which is combined with Chinese Restaurant Processes (CRPs) to determine the number of storylines automatically. However, the parameter inference procedure is too complex and the model requires long time to converge. This makes it impractical to be deployed in real-world applications.

Recently, deep learning techniques have been successfully applied to various natural language processing tasks. Several approaches (Mikolov et al., 2013; Le and Mikolov, 2014) such as word2vec have been proved efficient in representing rich syntactic and semantic information in text. Therefore, it would be interesting to combine the advantage of both probabilistic graphical model and deep neural networks. There have been some efforts in exploring this in recent years. For example, Yang et al. (2015) proposed a gaussian mixture neural topic model incorporating both the ordering of words and the semantic meaning of sentences into a topic model. Cao et al. (2015) explained topic models from the perspective of neural networks and proposed a neural topic model where the representation of words and documents are combined into a unified framework. However, to the best of our knowledge, there is no attempt in extracting structured repre-

Proceedings of NAACL-HLT 2018, pages 1727–1736
New Orleans, Louisiana, June 1 - 6, 2018.

sentation of storylines from text using neural network based approaches.

In this paper, we propose a novel neural model for storyline generation without the use of any annotated data. In specific, we assume that the storyline distributions of a document's title and its main body are similar. A pairwise ranking approach is used to optimize the model. We also assume that similar documents described in neighboring time periods should share similar storyline distributions. Hence, the model learned in the previous time period can be used for guiding the learning of the model in the current period. Based on the two assumptions, relevant events can be extracted and linked. Furthermore, storyline filtering based on confidence scores is performed. This makes it possible to generate new storylines.

The main contributions of this paper are summarized below:

- We propose a novel neural network based model to extract structured representations and evolution patterns of storylines. To the best of our knowledge, it is the first attempt to perform storyline generation based on neural network without any annotated data.
- The proposed approach has been evaluated on three corpora and a significant improvement on F-measure is achieved when compared to the state-of-the-art approaches. Moreover, the proposed approach only requires a faction of the training time in comparison with the second best approach.

2 Related Work

Considering storyline as hidden topic, storyline extraction can be casted into the topic detection and tracking (TDT) problem. One popular way to deal with TDT is through topic models. However, traditional topic models such as Latent Dirichlet Allocation (LDA) (Blei et al., 2003) do not detect the dynamics of topic over time. Griffiths and Steyvers (2004) clustered texts using LDA and then mapped the topics into corresponding time periods. Blei and Lafferty (2006) developed a dynamic topic model which captures the evolution of topics in a sequentially organized corpus of documents by using Gaussian time series on the natural parameter of the multinomial topics and logistic normal topic proportion models. Unlike early work that relied on Markov assumptions or discretization of time, Wang and McCallum (2006) proposed a topic-over-time (TOT) model where each topic is associated with a continuous distribution over timestamps. For each document, the mixture distribution over topics is influenced by both word co-occurrences and the document's timestamp. As a storyline might include more than one topic, Kawamae (2011) made an improvement over TOT and proposed a trend analysis model which generates storylines based on the model trained in the previous time period. Ahmed and Xing (2008) employed Recurrent Chinese Restaurant Processes (RCRPs) to cluster texts from discrete time slice while the number of clusters can grows automatically with the data at each epoch. Following this, many approaches were proposed for storyline extraction by combining RCRP with LDA (Ahmed et al., 2011a,b; Ahmed and Xing, 2013). Considering dependencies among clusters in different time periods, a distance-dependent CRP model was proposed by (Blei and Frazier, 2011) which defines a weight function to quantify the dependency in different clusters. Huang et al. (2015) proposed a Dynamic Chinese Restaurant Process (DCRP) model which considers the birth, survival and death of a storyline.

Recently, there have been increasing interests in exploring neural network based approaches for topic detection from text. These approaches can be divided into two categories, solely based on neural networks and a combination of topic models and neural networks. For the first category, topic distributions of documents are modeled by a hidden layer in neural networks. For example, Hinton and Salakhutdinov (2009) proposed a two layer probabilistic graphical model which is a generalization of the restricted Boltzmann machine, called a "Replicate Softmax". It can be used to automatically extract low-dimensional latent semantic representations from a large unstructured collection of documents. Larochelle and Lauly (2012) proposed a neural autoregressive topic model to compute the hidden units of the network efficiently. There are also many approaches trying to combine neural networks with topic models. For example, Yang et al. (2015) presented a Gaussian mixture neural topic model which incorporates both the ordering of words and the semantic meaning of sentences into topic modeling. To make the neural network based model more interpretable, Cao et

al. (2015) explained topic models from the perspective of neural networks and proposed a neural topic model where the representation of words and documents are combined into a unified framework. Tian et al. (2016) proposed a sentence level recurrent topic model assuming the generation of each word within a sentence is dependent on both the topic of the sentence and the the historical context of its preceding words in the sentence. Wan et al. (2012) introduced a hybrid model which combines a neural networks with a latent topic models. The neural network provides a low dimensional embedding for the input data while the subsequent distribution is captured by the topic model. However, most of the aforementioned models are solely for topic detection. They do not consider evolutionary topic clustering for storyline generation.

3 Methodology

To model the generation of a storyline in consecutive time periods from a stream of documents, we propose a neural network based approach, called Neural Storyline Extraction Model (NSEM), as shown in Figure 1. In this model, we have the following assumptions:

Assumption 1: *for a document, the storyline distribution of its title and main body should be similar.*

In general, for any given document, its title and main body should discuss the same storyline. Although title may exist metaphor and metonymy to catch the reader's eye ball, the key entities and words will not change such as name, location and so on. Therefore, it is reasonable to assume that the title h and its main body d of a document share a similar storyline distribution. The storyline distributions of title and main body are denoted as $\boldsymbol{p}(\boldsymbol{s}_h)$ and $\boldsymbol{p}(\boldsymbol{s}_d)$. Hence, $\boldsymbol{p}(\boldsymbol{s}_h)$ and $\boldsymbol{p}(\boldsymbol{s}_d)$ should be similar. Based on this assumption, documents at time period t can be clustered into several storylines in such a way. Let h_{pos} denotes the correct title to the main body d (positive example), and h_{neg} denotes an irrelevant title (negative example), the similarity of the storyline distribution derived from the main body d and that obtained from the correct title h_{pos} should be far more greater than that obtained from irrelevant titles h_{neg}, i.e. $sim(\boldsymbol{p}(\boldsymbol{s}_d), \boldsymbol{p}(\boldsymbol{s}_{h_{pos}})) \gg sim(\boldsymbol{p}(\boldsymbol{s}_d), \boldsymbol{p}(\boldsymbol{s}_{h_{neg}}))$. Different similarity metrics can be used to measure the similarity between two distributions.

Assumption 2: *for similar documents in neighboring time periods, they should share similar storyline distribution.*

It is assumed that similar documents in the neighboring time periods tend to share the same storyline. For example, a document with the title "Indian Election 2014: What are minorities to do?" and another document in the next time period with the title "The efficiency of Indian elections is time tested" should belong to the same storyline "*India election*". Based on this assumption, events extracted in different time period can be linked into storylines. As main body contains more information than title, we only use the storyline distribution of the main body, $\boldsymbol{p}(\boldsymbol{s}_d)$, in order to simplify the model structure. The learned information in the previous time period is used to supervise the learning in the current time period.

Based on the above two assumptions, the proposed NSEM as shown in Figure 1 contains the following four layers: (1) *Input layer* shown at the left bottom part of Figure 1, takes d, h_{pos} and h_{neg} as the input and transforms these texts into vectors; (2) *Main body-Storyline layer* and *Title-Storyline layer*, both are designed to generate storyline distributions; (3) *Similarity layer* aims to calculate the similarity between the storyline distribution of the main body and that of the title. In the top part of Figure 1, the model learned in previous time period is used to guide the storyline distribution learning in current time period. We explain the structure and function of each layer of NSEM in more details below:

Input Layer (d, h): the input layer aims to represent the main body d and title h with distributed embedding $\vec{d}$ and $\vec{h}$. Let the subscript pos denotes the relevant title h_{pos} (positive example) and subscript neg denotes an irrelevant title h_{neg} (negative example). For news articles, we pay more attention to the key elements of events such as location l, person p , organization o and keywords w. Thus an event is described by a quadruple $\langle l, p, o, w\rangle$. We extract these elements from the main body and concatenate their word embeddings as the feature vector $\vec{d} = [\vec{l}, \vec{p}, \vec{o}, \vec{w}]$. We obtain the title feature $\vec{h}$ in the same way.

We first identify named entities and treat those named entities with multi-word expressions (e.g., "Donald Trump") as single tokens. Then we train word2vec (Mikolov et al., 2013) to represent each entity with a 100-dimensional embedding vector. We also filter out less important keywords and en-

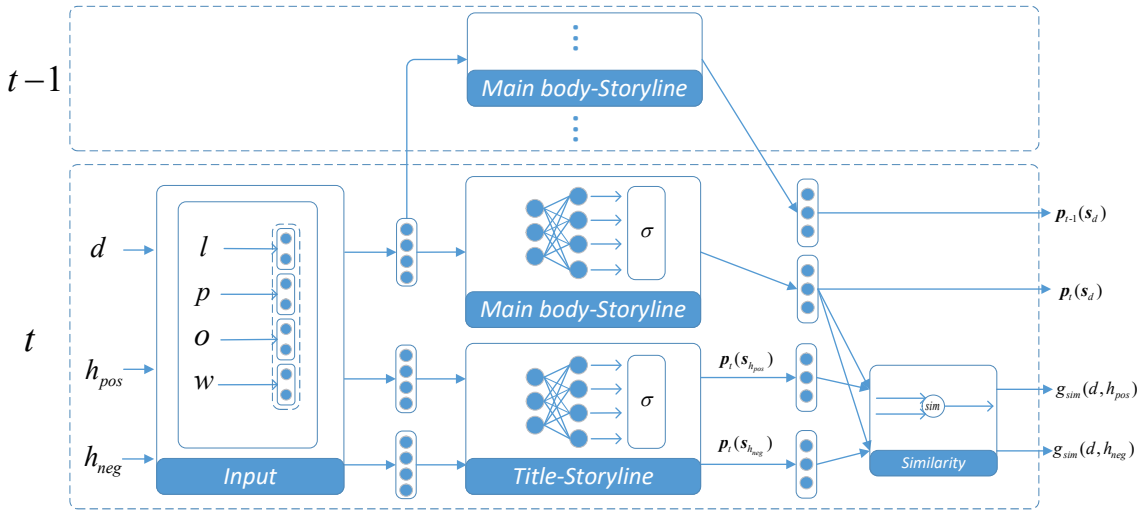


Figure 1: Overall architecture of the Neural Storyline Extraction Model (NSEM).

tities based on some criteria such as TFIDF. For a document containing more than one entity for the same event element type, for example, a document might contain mentions of different locations, we calculate the weighted sum of all location embeddings according to their occurrence number. If a certain event element is missing from a document, we set it to "null". After concatenating the four key event elements, each document or title is represented by a 400-dimensional embedding vector.

Main body-Storyline Layer ($\boldsymbol{p}(\boldsymbol{s}_d) \in \mathbb{R}^{1\times S}$): this layer aims to represent the storyline distribution $\boldsymbol{p}(\boldsymbol{s}_d)$ of main body d. Suppose there are a total of S storylines, the storyline distribution $\boldsymbol{p}(\boldsymbol{s}_d)$ is a S-dimensional vector, denoted as $\boldsymbol{p}(\boldsymbol{s}_d) = \{p(s_d = 1), \cdots, p(s_d = S)\}$. It can be formulated as below:

$$\boldsymbol{p}(\boldsymbol{s}_d) = f(\vec{d} \cdot \boldsymbol{W}_1 + \boldsymbol{b}_1) \quad (1)$$

where $\boldsymbol{W_1} \in \mathbb{R}^{K\times S}$ denotes the weight matrix, $\boldsymbol{b}$ denote the bias, $K = 400$ is the dimension of the document representation, and f denotes the activation function. Here we use the Softmax function. The probability of the main body d belonging to the storyline i can be written below:

$$p(s_d = i) = \frac{\exp(\vec{d} \cdot \boldsymbol{W}_{1i} + \boldsymbol{b}_{1i})}{\sum_{i=1}^{S} \exp(\vec{d} \cdot \boldsymbol{W}_{1i} + \boldsymbol{b}_{1i})} \quad (2)$$

Title-Storyline Layer ($\boldsymbol{p}(\boldsymbol{s}_h) \in \mathbb{R}^{1\times S}$): this layer aims to represent the storyline distribution $\boldsymbol{p}(\boldsymbol{s}_h)$ of title h. Similar to the Main body-Storyline layer, we can obtain $\boldsymbol{p}(\boldsymbol{s}_h)$ and $p(s_h = i)$ of title h in the following way:

$$\boldsymbol{p}(\boldsymbol{s}_h) = f(\vec{h} \cdot \boldsymbol{W}_2 + \boldsymbol{b}_2) \quad (3)$$

$$p(s_h = i) = \frac{\exp(\vec{h} \cdot \boldsymbol{W}_{2i} + \boldsymbol{b}_{2i})}{\sum_{i=1}^{S} \exp(\vec{h} \cdot \boldsymbol{W}_{2i} + \boldsymbol{b}_{2i})} \quad (4)$$

Similarity Layer ($g_{sim} \in \mathbb{R}$): this layer aims to calculate the similarity of the distributions between $\boldsymbol{p}(\boldsymbol{s}_d)$ and $\boldsymbol{p}(\boldsymbol{s}_h)$. The similarity score g_{sim} is calculated by the Kullback-Leibler (KL) divergence:

$$g_{sim}(d, h) = -\sum \boldsymbol{p}(\boldsymbol{s}_d) \log \frac{\boldsymbol{p}(\boldsymbol{s}_h)}{\boldsymbol{p}(\boldsymbol{s}_d)} \quad (5)$$

The similarity can be also calculated by other metric methods.

3.1 Storyline Construction

Different from the common way which link relevant events into storyline, we extract it in a unified framework. According to our second assumption, for the current time period t, we employ the storyline generation results in the previous time period $t-1$ as constraints to guide the storyline generation process in t. For a document d_t (we only use the main body here) in the time period t, we first use the model trained in $t-1$ to predict its storyline distribution $\boldsymbol{p}_{t-1}(\boldsymbol{s}_{d_t})$. Hence when we learn $\boldsymbol{p}_t(\boldsymbol{s}_{d_t})$, we would expect it to be similar to $\boldsymbol{p}_{t-1}(\boldsymbol{s}_{d_t})$. By doing so, we can link relevant events in different time periods together. For cases where intermittent storylines are observed, i.e., the related events occur initially, but disappear in certain time periods and re-occur later, we select

documents randomly from all previous time periods and make them participate in the learning of current model.

3.2 Training

Our first assumption assumes that *for a document, its title and main body should share similar storyline distributions*. Hence, we use a pairwise ranking approach (Collobert et al., 2011) to optimize $\boldsymbol{p}(\boldsymbol{s}_d)$ and $\boldsymbol{p}(\boldsymbol{s}_h)$. The basic idea is that the storyline distribution of the main body d should be more similar to that of the relevant title than irrelevant ones. We first define the loss function as below:

$$\mathcal{L}_1(d, h_{pos}, h_{neg}) = \max(0, \Omega - g_{sim}(d, h_{pos}) + g_{sim}(d, h_{neg})) \quad (6)$$

where Ω denotes the margin parameter, h_{pos} denotes the relevant title and h_{neg} denotes an irrelevant title. We choose titles whose elements $\langle l, p, o, k\rangle$ have no intersection with those positive titles from the current time period as negative examples.

Our second assumption assume that *for similar documents in neighboring time periods, they should share similar storyline distribution*. Hence, the model learned in the previous time period can be used for guiding the learning of the model in the current period. Hence, when constructing storyline for the main body d in current time period t, we use the model in previous time period $t-1$ and predict the storyline distribution $\boldsymbol{p}_{t-1}(\boldsymbol{s}_d)$. Then we measure current storyline distribution $\boldsymbol{p}_t(\boldsymbol{s}_d)$ and predicted distribution $\boldsymbol{p}_{t-1}(\boldsymbol{s}_d)$ by KL divergence which can be defined as below:

$$\mathcal{L}_2(d) = \sum \boldsymbol{p}_{t-1}(\boldsymbol{s}_d) \log \frac{\boldsymbol{p}_t(\boldsymbol{s}_d)}{\boldsymbol{p}_{t-1}(\boldsymbol{s}_d)} \quad (7)$$

Therefore, the final objective function is to minimize:

$$\mathcal{L} = \sum_d (\alpha \mathcal{L}_1(d, h_{pos}, h_{neg}) + \beta \mathcal{L}_2(d)) \quad (8)$$

where α and β are the weights controlling the contributions of the two loss terms.

For the start time period, we only use $\mathcal{L}_1$ to optimize our model. Let Φ_t denote the model parameter in the time period t. Based on the model structure and the loss function described above, the training procedure for NSEM is given in Algorithm 1.

Algorithm 1 Training procedure for NSEM at the time period t

Require: main bodies $\boldsymbol{d}$; titles $\boldsymbol{h}$; model parameter Φ_{t-1} at the time period $t-1$

```
Initialize Φ_t
for d ∈ d do
    Calculate its storyline distribution based on Φ_{t-1}
end for
repeat
    for every minibatch M in (d, h) do
        for every pair (d_i, h_{i,pos}) in minibatch M do
            Calculate the storyline distribution p(s_{d_i})
            Calculate the storyline distribution p(s_{h_{i,pos}})
            Sample an irrelevant title h_{i,neg} where h_{i,neg} ∩ h_{i,pos} = ∅
            Calculate the storyline distribution p(s_{h_{i,neg}})
            Calculate L_1(d_i, h_{i,pos}, h_{i,neg})
            Calculate L_2(d_i)
        end for
        Calculate minibatch loss L_M = Σ_{d_i}(αL_1 + βL_2) and gradients ∇_{Φ_t} L_M
        Update model parameter Φ_t
    end for
until Convergence
```

3.3 Post-processing

As the number of storylines at each time period is assumed to be the same, some newly emerging storylines might be incorrectly linked with previous storylines. Therefore, post-processing is needed to filter out such erroneous linkings. We assume that if a current storyline does not have any key element in common with previously extracted storyline, it should be flagged as a new storyline. We define the **Coverage** of the storyline s as below:

$$Coverage(s, t, M) = (element)_s^t \cap (element)_s^{t-M} \quad (9)$$

where $(element)_s^t$ denotes the set of event elements in the time period t for storyline s and $(element)_s^{t-M}$ denote the set of event elements in the last M time periods for storyline s. If the coverage $Coverage(s, t, M)$ is less than a threshold N, the current storyline s is considered as a new one. For example, if the current storyline' $Coverage$ with index 5 is less than N, then previ-

ous storyline with index 5 stops at current period and the current storyline with index 5 is a new one.

4 Experiments

4.1 Setup

To evaluate the proposed approach, we use the three datasets as in (Zhou et al., 2016). The statistics of the three datasets are presented in Table 4.1. Among which the Dataset III includes 30 different types of manually annotated storylines which are categorized into four types: (1) long-term storylines which last for more than 2 weeks; (2) short-term storylines which last for less than 1 week; (3) intermittent storylines which last for more than 2 weeks in total, but stop for a time and then appear again; (4) new storylines which emerge in the middle of the period, not at the beginning.

Datasets	Documents	Storylines	Dates
I	526,587	N/A	1-30 May 2014
II	101,654	77	1-7 May 2014
III	23,376	30	1-30 May 2014

Table 1: Statistics of the three datasets.

In our experiments, we used the Stanford named entity recognizer [1] for identifying the named entities. In addition, we removed common stopwords and only kept tokens which are verbs, nouns, or adjectives from these news articles.

We chose the following four methods as the baseline approaches.

1. DLDA (Blei and Lafferty, 2006): the dynamic LDA is based on the Markovian assumption that the topic-word distribution at the current time period is only influenced by the topic-word distribution in the previous time period. Moreover, topic-word distributions are linked across time periods by a Markovian chain.

2. RCRP (Ahmed et al., 2011a): it is a non-parametric model for evolutionary clustering based on RCRP, which assumes that the past story popularity is a good prior for current popularity.

3. SDM (Zhou et al., 2015): it assumes that the number of storylines is fixed and the storyline is modeled as a joint distribution over entities and keywords. The dependency of different stories of the same storyline at different time periods is captured by modifying Dirichlet priors.

4. DSEM (Zhou et al., 2016): this model is integrated with CRPs so that the number of storylines can be determined automatically without human intervention. Moreover, per-token Metropolis-Hastings sampler based on light LDA (Yuan et al., 2015) is used to reduce sampling complexity.

For DLDA, SDM and our model NSEM, the storyline number is set to 100 on both Dataset I-I and III. In consideration of the dependency to the historical storyline distributions, the number of past epochs M is set to 7 for both SDM and DSEM. For RCRP, the hyperparameter α is set to 1. For our model NSEM, the threshold Ω is set to 0.5 and the loss weight α and β are set to 1 and 0.5 respectively. In postprocess step, we empirically set the N to 7.

To evaluate the performance of the proposed approach, we use precision, recall and F-measure which are commonly used in evaluating information extraction systems. The precision is calculated based on the following criteria: 1) The entities and keywords extracted refer to the same storyline; 2) The duration of the storyline is correct. We assume that the start date (or end date) of a storyline is the publication date of the first (or last) related news article.

As there is no gold standard available for Dataset I, we do manual examination with the experimental result. We search for the same period of news and compare it with our results in the criteria.

4.2 Experimental Results

The experimental results of the proposed approach in comparison to the baselines on Dataset I, II and III are presented in Table 2. For Dataset I, as it is hard to know the ground-truth of storylines, we only report the precision value by manually examining the extracted storylines.

It can be observed from Table 2 that the proposed approach achieves the best performance on the three datasets. In specific, for Dataset I, NSEM extracts more storylines and with a higher precision value. For Dataset II containing 77 storylines, NSEM extracts 81 storylines among which

[1] https://nlp.stanford.edu/software/CRF-NER.html

Dataset I			
Method	Precision(%)	# of extracted storylines	
SDM	70.20	104	
DSEM	75.43	114	
NSEM	**76.58**	**121**	
Dataset II			
Method	Precision(%)	Recall(%)	F-measure(%)
DLDA	62.67	61.03	61.84
RCRP	67.11	66.23	66.67
SDM	70.67	68.80	69.27
DSEM	73.17	77.92	75.47
NSEM	**75.31**	**79.22**	**77.22**
Dataset III			
Method	Precision(%)	Recall(%)	F-measure(%)
DLDA	46.16	43.33	42.86
RCRP	61.54	53.33	57.14
SDM	54.17	43.33	48.15
DSEM	75.00	70.00	72.41
NSEM	**77.78**	70.00	**73.69**

Table 2: Performance comparison of the storyline extraction results on Dataset I, II and III.

61 are correct and outperforms DSEM with 2% in F-measure. For dataset III consisting of 30 storylines, NSEM extracted 27 storylines among which 21 are correct. Although its recall value is the same as DSEM, its precision value is nearly 3% higher which results in better F-measure.

4.3 Impact of the Number of Storylines S

The proposed approach needs to preset the number of storylines. To study the impact of the number of storylines on the performance of the proposed model, we conducted experiments Dataset III with different numbers of storylines S varying between 25 and 150. Table 3 shows the performance of storyline extraction with different value of S. It can be observed that both precision and recall of NSEM increase with the increasing number of storylines until it reaches 100. If further increasing S, the precision/recall have slight change and the F-measure become relatively stable.

4.4 Structured Browsing

We illustrate the evolution of storylines using structured browsing. The structured information of the storylines such as locations, persons, entities, keywords are presented, together with titles of some related documents. The number of related documents for each storyline is also depicted to allow an easy visualization of storyline popularity over time. Figure 2 illustrates three different types of storylines including "Apple vs Samsung", "Pistorious shoot Steenkamp" and "Egypt election".

Dataset III			
S	Precision(%)	Recall(%)	F-measure(%)
25	66.67	33.33	44.44
50	73.08	46.67	56.96
75	76.92	53.33	62.99
100	77.78	70.00	73.69
125	78.13	73.33	75.65
150	78.79	70.00	74.13

Table 3: The performances of NSEM with different S.

For the first storyline "Apple vs Samsung", it starts at the beginning of the month and only lasts for 9 days. Three representative epochs are highlighted. From the extracted organizations, "Apple, Samsung", and keywords, "patent, infringe", it can be easily deduced that this is about "Apple and Samsung infringed patents".

For the storyline "Pistorious shoot Steenkamp", it is an intermittent storyline which lasts for more than 2 weeks but with no related news articles in some of the days in between. From Figure 2, it can be observed that the storyline ceases for 2 days in Day 10 and 11. From the structured representation of the early storylines, it can be observed that there is a shooting event about Pistorious and Steenkamp in South African. After 2 day's silence, in Day 13, public attention was raised once again since Pistorius applied for mental tests.

For the last storyline "Egypt election", it starts in Day 20 and continues beyond the end of May. From the key event elements, location "Egypt" and keywords "presidential, election", it can be easily inferred that there was a presidential election in Egypt. It can also be observed that Sisi and Morsi were both candidates for the Egypt's presidential election from persons extracted, "Sisi, Morsi" in Day 26. In Day 29, the storyline reached to the climax since Sisi won the election, which can be discovered from the title "Sisi elected #Egypt president by landslide".

4.5 Time Complexity

To explore the efficiency of the proposed approach, we conducted an experiment by comparing the proposed approach NSEM with DSEM. DSEM employs the Metropolis-Hastings sampler to

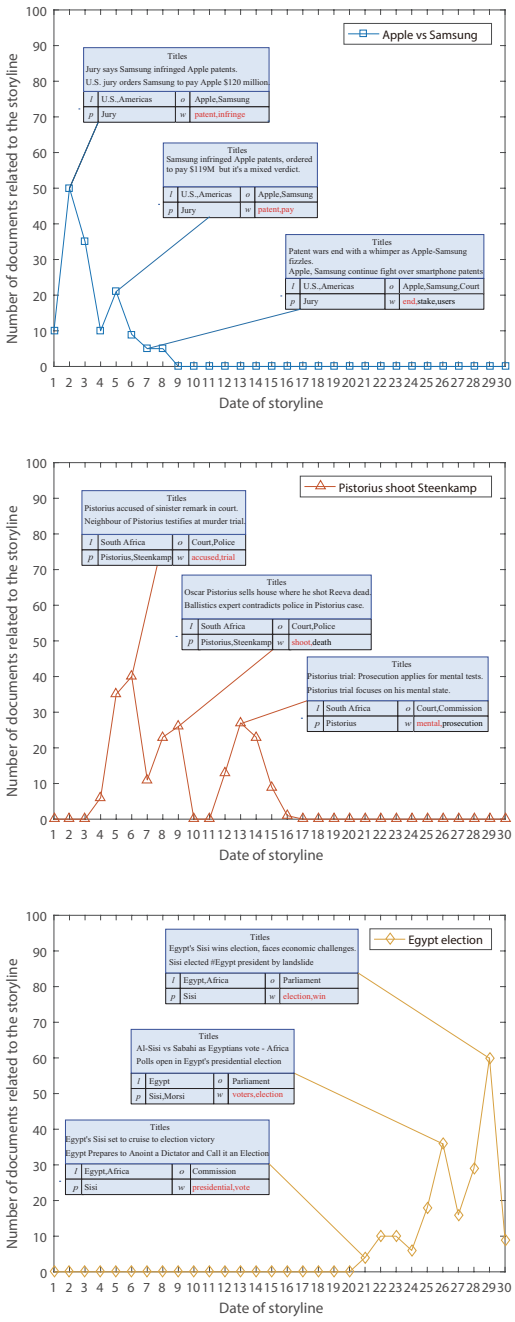


Figure 2: The structured representations of three example storylines.

boost the sampling complexity in order to achieve faster convergence. We train both models on training data varying from 1,000 to 10,000 documents. Figure 3 illustrates the logarithm of time consumed for each training set. It can be observed that NSEM trains 30 times faster compared to DSEM, showing the advantage of using a neural network based approach in comparison with a Bayesian model based method.

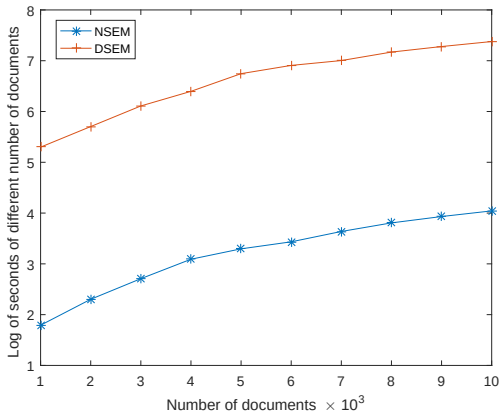


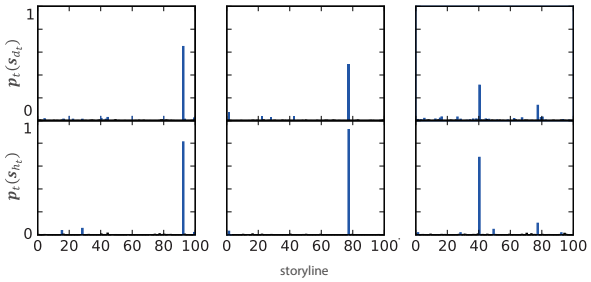
Figure 3: Comparison of training time between NSEM and DSEM.

4.6 Visualization of the Learned Distribution

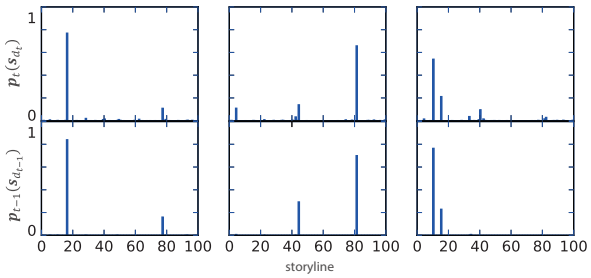
Our proposed model is based on the two distribution similarity assumptions which we presented in the Methodology section. To investigate the quality of the learned storyline distribution, we conducted an experiment on Dataset III where the storyline number S is set to 100. We randomly choose three documents and calculate the storyline distribution of theirs title and main body based on our learned NSEM. We also randomly select three pairs similar documents in different time periods and draw their main body storyline distributions based on the learned NSEM. It can be observed from Figure 4 that the storyline distributions of the title and the main body of a document are similar. Moreover, the storyline distributions of two similar documents in different time periods are also similar.

5 Conclusions and Future Work

In this paper, we have proposed a neural network based storyline extraction model, called NSEM, to extract structured representations of storyline from news articles. NSEM was designed based on the two assumptions about the similarity of storyline distributions of the title and the main body of the same document, and the similarity of storyline distributions of similar documents in different time periods. Experimental results show that our proposed model outperforms the state-of-the-art approaches and only requires a fraction of training time. In future work, we will explore the extension of our proposed model to cater for varying number of storylines automatically and also better deal with intermittent storylines.



(a) Storyline distributions of title and main body.



(b) Storyline distributions of similar documents in different time periods.

Figure 4: Visualization of the learned storyline distributions.

Acknowledgments

This work was funded by the National Natural Science Foundation of China (61772132), the Natural Science Foundation of Jiangsu Province of China (BK20161430) and Innovate UK (103652).

References

Amr Ahmed, Qirong Ho, Jacob Eisenstein, Eric Xing, Alexander J Smola, and Choon Hui Teo. 2011a. Unified analysis of streaming news. In *Proceedings of the 20th international conference on World wide web*. ACM, pages 267–276.

Amr Ahmed, Qirong Ho, Choon Hui Teo, Jacob Eisenstein, Alex Smola, and Eric Xing. 2011b. Online inference for the infinite topic-cluster model: Storylines from streaming text. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. pages 101–109.

Amr Ahmed and Eric Xing. 2008. Dynamic nonparametric mixture models and the recurrent chinese restaurant process: with applications to evolutionary clustering. In *Proceedings of the 2008 SIAM International Conference on Data Mining*. SIAM, pages 219–230.

Amr Ahmed and Eric P Xing. 2013. Scalable dynamic nonparametric bayesian models of content and users. In *IJCAI*. pages 3111–3115.

David M Blei and Peter I Frazier. 2011. Distance dependent chinese restaurant processes. *Journal of Machine Learning Research* 12(Aug):2461–2488.

David M Blei and John D Lafferty. 2006. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*. ACM, pages 113–120.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3(Jan):993–1022.

Ziqiang Cao, Sujian Li, Yang Liu, Wenjie Li, and Heng Ji. 2015. A novel neural topic model and its supervised extension. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*. pages 2210–2216.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(Aug):2493–2537.

Qiming Diao and Jing Jiang. 2014. Recurrent chinese restaurant process with a duration-based discount for event identification from twitter. In *Proceedings of the 2014 SIAM International Conference on Data Mining*. SIAM, pages 388–397.

Thomas L Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National academy of Sciences* 101(suppl 1):5228–5235.

Geoffrey E Hinton and Ruslan R Salakhutdinov. 2009. Replicated softmax: an undirected topic model. In *Advances in neural information processing systems*. pages 1607–1614.

Lifu Huang and Lian'en Huang. 2013. Optimized event storyline generation based on mixture-event-aspect model. In *EMNLP*. pages 726–735.

Rui Huang, Fengyuan Zhu, and Pheng-Ann Heng. 2015. The dynamic chinese restaurant process via birth and death processes. In *AAAI*. pages 2687–2693.

Noriaki Kawamae. 2011. Trend analysis model: trend consists of temporal words, topics, and timestamps. In *Proceedings of the fourth ACM international conference on Web search and data mining*. ACM, pages 317–326.

Hugo Larochelle and Stanislas Lauly. 2012. A neural autoregressive topic model. In *Advances in Neural Information Processing Systems*. pages 2708–2716.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. pages 1188–1196.

Jiwei Li and Claire Cardie. 2014. Timeline generation: Tracking individuals on twitter. In *Proceedings of the 23rd international conference on World wide web*. ACM, pages 643–652.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Computer Science* .

Fei Tian, Bin Gao, Di He, and Tie-Yan Liu. 2016. Sentence level recurrent topic model: Letting topics speak for themselves. *arXiv preprint arXiv:1604.02038* .

Li Wan, Leo Zhu, and Rob Fergus. 2012. A hybrid neural network-latent topic model. In *International Conference on Artificial Intelligence and Statistics*. pages 1287–1294.

Xuerui Wang and Andrew McCallum. 2006. Topics over time: a non-markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 424–433.

Rui Yan, Liang Kong, Congrui Huang, Xiaojun Wan, Xiaoming Li, and Yan Zhang. 2011. Timeline generation through evolutionary trans-temporal summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 433–443.

Min Yang, Tianyi Cui, and Wenting Tu. 2015. Ordering-sensitive and semantic-aware topic modeling. In *AAAI*. pages 2353–2360.

Jinhui Yuan, Fei Gao, Qirong Ho, Wei Dai, Jinliang Wei, Xun Zheng, Eric Po Xing, Tie-Yan Liu, and Wei-Ying Ma. 2015. Lightlda: Big topic models on modest computer clusters. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, pages 1351–1361.

Deyu Zhou, Haiyang Xu, Xin-Yu Dai, and Yulan He. 2016. Unsupervised storyline extraction from news articles. In *IJCAI*. pages 3014–3021.

Deyu Zhou, Haiyang Xu, and Yulan He. 2015. An unsupervised bayesian modelling approach for storyline detection on news articles. In *EMNLP*. pages 1943–1948.

Provable Fast Greedy Compressive Summarization with Any Monotone Submodular Function

Shinsaku Sakaue Tsutomu Hirao Masaaki Nishino Masaaki Nagata
NTT Communication Science Laboratories
{sakaue.shinsaku, hirao.tsutomu}@lab.ntt.co.jp
{nishino.masaaki, nagata.masaaki}@lab.ntt.co.jp

Abstract

Submodular maximization with the greedy algorithm has been studied as an effective approach to *extractive summarization*. This approach is known to have three advantages: its applicability to many useful submodular objective functions, the efficiency of the greedy algorithm, and the provable performance guarantee. However, when it comes to *compressive summarization*, we are currently missing a counterpart of the extractive method based on submodularity. In this paper, we propose a fast greedy method for compressive summarization. Our method is applicable to any monotone submodular objective function, including many functions well-suited for document summarization. We provide an approximation guarantee of our greedy algorithm. Experiments show that our method is about 100 to 400 times faster than an existing method based on integer-linear-programming (ILP) formulations and that our method empirically achieves more than 95%-approximation.

1 Introduction

Automatic document summarization continues to be a seminal subject of study in natural language processing and information retrieval (Luhn, 1958; Edmundson, 1969; Cheng and Lapata, 2016; Peyrard and Eckle-Kohler, 2017). Owing to the recent advances in data collection, the size of document data to be summarized has been exploding, which has been bringing a drastic increase in the demand for fast summarization systems.

Extractive summarization is a widely used approach to designing fast summarization systems. With this approach, we construct a summary by extracting some sentences from the original document(s). The extractive approach is not only fast but also has the potential to achieve state-of-the-art ROUGE scores (Lin, 2004), which was revealed by Hirao et al. (2017b). In many existing methods, sentences are extracted by solving various subset selection problems: for example, the *knapsack problem* (McDonald, 2007), *maximum coverage problem* (Filatova and Hatzivassiloglou, 2004; Takamura and Okumura, 2009a), *budgeted median problem* (Takamura and Okumura, 2009b), and *submodular maximization problem* (Lin and Bilmes, 2010). Of particular interest, the method based on submodular maximization has three advantages: (1) Many objective functions used for document summarization are known to be *monotone* and *submodular* (Lin and Bilmes, 2011; J Kurisinkel et al., 2016); examples of such functions include the *coverage function*, *diversity reward function*, and ROUGE. Therefore, the method can deliver high performance by using monotone submodular objective functions that are suitable for the given tasks. (2) The efficient greedy algorithm is effective for the submodular maximization problem, which provides fast summarization systems. (3) Theoretical performance guarantees of the greedy algorithm can be proved; for example, a $\frac{1}{2}(1 - e^{-1})$-approximation guarantee can be obtained.

Although the above extractive methods successfully obtain summaries with high ROUGE scores, they have the following shortcoming: A long sentence typically has redundant parts, which means a summary constructed simply by extracting some sentences often includes many redundant parts. As a result, if the limitation placed on summary length is tight, the extractive approach cannot yield an informative summary.

Compressive summarization is known to be effective in overcoming this problem. With this approach, a summary is constructed with some

Proceedings of NAACL-HLT 2018, pages 1737–1746
New Orleans, Louisiana, June 1 - 6, 2018.

compressed sentences, and thus we can obtain a concise and informative summary. To make compressed sentences, the dependency-tree-based approach (Filippova and Strube, 2008) is often used, which is advantageous in that each compressed sentence preserves its original dependency relations. Specifically, given a set of dependency trees constructed for sentences in the original documents, a summary is obtained by extracting some rooted subtrees; each subtree corresponds to a compressed sentence. Different from the extractive summarization, the dependency relations in each sentence must be taken into account, and hence the aforementioned extractive methods cannot be applied to compressive summarization. A number of methods have been proposed for compressive summarization (Berg-Kirkpatrick et al., 2011; Almeida and Martins, 2013; Morita et al., 2013; Kikuchi et al., 2014; Hirao et al., 2017a). These methods formulate summarization as a type of combinatorial optimization problem with a tree constraint, and they obtain summaries by solving the problem. Unfortunately, the existing methods have two drawbacks: (1) The class of objective functions to which they are applicable is limited; for example, they work only with the linear function or coverage function. As a result, the performance of these methods cannot be improved by elaborating the objective functions. (2) They contain costly procedures as their building blocks: integer-linear-programming (ILP) solvers, dynamic programming (DP) algorithms, and so on. Therefore, they are not fast enough to be applied to large-scale document data. In a nutshell, compressive summarization is currently missing a fast method that is applicable to a wide variety of objective functions.

1.1 Our Contribution

In this paper, we propose a submodularity-based greedy method for compressive summarization. Our method is, so to speak, a compressive counterpart of the greedy method for extractive summarization (Lin and Bilmes, 2010). Similar to the extractive method, our method has the three key advantages:

1. Our method works with any monotone submodular objective function, a wide class of useful objective functions, examples of which include the coverage function, ROUGE, and many others (Lin and Bilmes, 2011; J Kurisinkel et al., 2016).

2. Our method is faster than existing compressive summarization methods since it employs the efficient greedy algorithm. Specifically, given a set, V, of all textual units contained in the document data and a summary length limitation value, L, our method requires at most $O(L|V|)$ objective function evaluations. Experiments show that our method is about 100 to 400 times faster than the ILP-based method implemented with `CPLEX`.

3. A theoretical guarantee of our method can be proved; specifically, a $\frac{1}{2}(1 - e^{-1/\lambda})$-approximation guarantee can be obtained, where λ is a parameter defined from given document data (a definition is shown later). This result generalizes the $\frac{1}{2}(1 - e^{-1})$-approximation of the greedy algorithm for submodular maximization with a knapsack constraint (Leskovec et al., 2007). In experiments, our method achieved more that 95%-approximation. Furthermore, our method attained ROUGE_1 scores comparable to those of the ILP-based method.

1.2 Related Work

There are many existing methods for compressive summarization (Berg-Kirkpatrick et al., 2011; Almeida and Martins, 2013; Morita et al., 2013; Kikuchi et al., 2014; Hirao et al., 2017a), and they attempt to create summaries by solving optimization problems with a tree and length constraints. Unfortunately, these methods accept only a few objective functions.

A common approach is to use ILP formulations. Berg-Kirkpatrick et al. (2011) formulate the problem as an ILP with the coverage objective function, which is solved by using an ILP solver. Almeida and Martins (2013) also employs an ILP formulation and solves the problem via an algorithm based on *dual decomposition*, which runs faster than an ILP solver.[1] These ILP-based methods are optimal in terms of objective function values. However, it is hard to apply them to large-scale document data since to solve ILPs often takes long computation time.

[1]Their method was observed to be about 25 times faster than `GLPK`, a commonly used free ILP solver. On the other hand, `CPLEX`, which is a commercial ILP solver used in our experiments, was observed to be about 3 to 20 times faster than `GLPK`, and our method is about 100 to 400 times faster than `CPLEX`. Consequently, our method is estimated to be about 12 to 320 times faster than their method.

In an attempt to uncover the potential power of dependency-tree-based compressive summarization, Hirao et al. (2017a) solved ILPs with the ROUGE objective function with an ILP solver. Their method obtains summaries by directly maximizing the ROUGE score for given reference summaries (i.e., any other methods cannot achieve higher ROUGE scores than their method). The resulting summaries, called *oracle summaries*, were revealed to attain substantially high rouge scores, which implies that there remains much room for further research into compressive summarization.

A greedy method with a DP algorithm (Morita et al., 2013) is probably the closest one to our idea. Their method iteratively chooses compressed sentences in a greedy manner, for which a DP algorithm is employed. Thanks to the submodularity of their objective function, their method enjoys a $\frac{1}{2}(1-e^{-1})$-approximation guarantee. However, because of the costly DP procedure, their method is less scalable than the standard greedy methods such as the extractive method (Lin and Bilmes, 2010) and ours. Moreover, it is applicable only to objective functions that are designed for their problem settings; for example, it cannot use ROUGE as an objective function.

1.3 Overview of Our Approach

A high-level sketch of our approach is as follows: As in many existing works, we formulate the compressive summarization task as a combinatorial optimization problem with a tree constraint, which we call the *submodular tree knapsack problem* (STKP). STKP is generally NP-hard; in fact, it includes the knapsack problem and maximum coverage problem as special cases. Unfortunately, as we will see later, a naive greedy algorithm for STKP does not offer any approximation guarantee in general. The main difficulty with STKP is that its tree constraint is too complex. To avoid dealing with the complex constraint directly, we transform STKP into a special case of the *submodular cost submodular knapsack problem* (SCSKP) (Iyer and Bilmes, 2013). For general SCSKP, no approximation guarantee has been proved. Fortunately, in our case, a $\frac{1}{2}(1-e^{-1/\lambda})$-approximation can be proved by exploiting the structure of the resulting SCSKP. Thus we obtain a fast greedy method for compressive summarization, which works with various monotone submodular objective functions and enjoys an approximation guarantee.

2 Submodularity

Given finite set V (e.g., a set of chunks), set function $g : 2^V \to \mathbb{R}$ is said to be *submodular* if $g(A \cup B) + g(A \cap B) \leq g(A) + g(B)$ holds for any $A, B \subseteq V$. We define $g(A \mid B) \coloneqq g(A \cup B) - g(B)$. The submodularity is also characterized by the following *diminishing return property*: $g(\{v\} \mid A) \geq g(\{v\} \mid B)$ for any $A \subseteq B$ and $v \in V \backslash B$. Set function g is *monotone* if $g(A) \leq g(B)$ for any $A \subseteq B$. In this paper, we focus on monotone submodular functions such that $g(\emptyset) = 0$. The submodularity and monotonicity are a natural fit for document summarization; intuitively, the marginal gain, $g(\{v\} \mid S)$, of adding new chunk $v \in V$ to summary $S \subseteq V$ is small if S already has many chunks (submodularity), and a summary becomes more informative as it gets more chunks (monotonicity). In fact, as in (Lin and Bilmes, 2011), many objective functions well-suited for document summarization have submodularity and monotonicity; examples of such functions include the coverage function, diversity reward function, and ROUGE, to name a few.

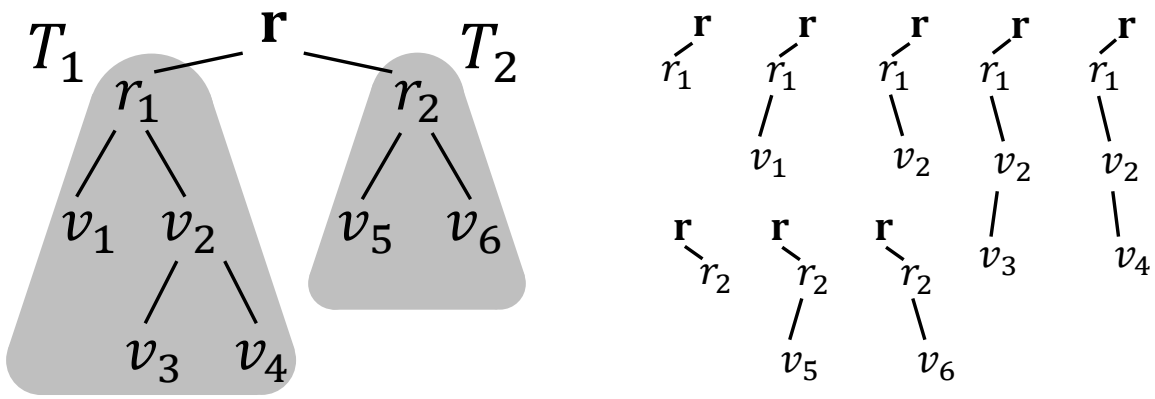
3 Problem Statements

We formulate the summarization task as the following subtree extraction problem called STKP hereafter. In what follows, we let $[M] \coloneqq \{1, \ldots, M\}$ for any positive integer M.

We attempt to summarize document data consisting of N sentences. Each sentence forms a dependency tree, which can be constructed by using existing methods (e.g., (Filippova and Strube, 2008; Filippova and Altun, 2013)). For convenience, we call the dependency tree of a sentence the *sentence tree*. The i-th sentence ($i \in [N]$) yields sentence tree $T_i = (V_i, E_i)$ rooted at $r_i \in V_i$, where V_i is a set of textual units (e.g., words or chunks) contained in the i-th sentence, and edges in E_i represent their dependency relations. We define a *document tree* with a dummy root vertex $\mathbf{r}$ as $\mathbf{T} \coloneqq (\{\mathbf{r}\} \cup V, E)$, where V and E are vertex and edge sets, respectively, defined as follows:

$$V \coloneqq \bigcup_{i \in [N]} V_i, \quad E \coloneqq \bigcup_{i \in [N]} \{E_i \cup \{(\mathbf{r}, r_i)\}\}.$$

Namely, V is the set of all textual units contained in the document data, and edges in E represent the dependency relations as well as the relations between $\mathbf{r}$ and r_i, with which the multiple sentence



(a) Document tree $\mathbf{T} = (\{\mathbf{r}\} \cup V, E)$

(b) Set of all paths rooted at $\mathbf{r}$

Figure 1: Illustration of the problem reformulation. The left figure is a document tree rooted at $\mathbf{r}$; it consists of two sentence trees, T_1 and T_2, rooted at r_1 and r_2, respectively. We have $V = \{r_1, r_2, v_1, \ldots, v_6\}$. The right figure shows $\mathcal{P}$, the set of all paths rooted at $\mathbf{r}$. Note that $|V| = |\mathcal{P}|$ holds. With our method, the greedy algorithm is performed over $\mathcal{P}$, which requires at most $O(|V|)$ objective function evaluations in each iteration.

trees form a single document tree. Figure 1 (a) illustrates an example of a document tree.

Given document tree $\mathbf{T}$, a summary preserves the original dependency relations if it forms a subtree rooted at $\mathbf{r}$ in $\mathbf{T}$. Therefore, our aim is to find a rooted subtree of $\mathbf{T}$ that includes informative textual units. For each $v \in V$, the length of v is denoted by $\ell_v \geq 0$; for example, ℓ_v is the number of words or characters in chunk v. If $S \subseteq V$ is a subset of the textual units included in an obtained summary, its total length must be less than or equal to the given length limitation value $L \geq 0$; namely, the following knapsack constraint must be satisfied: $\sum_{v \in S} \ell_v \leq L$. The quality of summary S is evaluated by a monotone submodular function g. Consequently, compressive summarization is formulated as STKP:

$$\begin{aligned} \underset{S \subseteq V}{\text{maximize}} \quad & g(S) \qquad (1) \\ \text{subject to} \quad & \sum_{v \in S} \ell_v \leq L, \\ & S \cup \{\mathbf{r}\} \text{ forms a subtree in } \mathbf{T}. \end{aligned}$$

At first glance, it may seem that the following naive greedy approach works well for this problem: Starting from root $\mathbf{r}$, we sequentially add the most beneficial child to the current solution until the knapsack constraint is violated. Unfortunately, the approximation ratio of this method can become arbitrarily bad since it may miss beneficial vertices that are far from $\mathbf{r}$; if such missed vertices are more beneficial than those added to the solution by a considerable margin, the resulting approximation ratio is almost equal to zero. To avoid this difficulty, we reformulate STKP in the next section.

4 Proposed Method

We observed that the naive greedy algorithm does not work well for STKP (1) due to the complex tree constraint. We circumvent this difficulty by transforming STKP into a special case of the submodular cost submodular knapsack problem (SCSKP). We then provide a greedy algorithm for SCSKP. An approximation guarantee of the greedy algorithm is also presented.

4.1 Problem Reformulation

We show that STKP can be transformed into SCSKP. Let $\mathcal{P}$ be a set of all paths that connect $v \in V$ to $\mathbf{r}$. Note that there is a one-to-one correspondence between $v \in V$ and $p \in \mathcal{P}$ that connects v to $\mathbf{r}$, and hence $|\mathcal{P}| = |V|$. We define $V_p \subseteq V$ as the set of vertices that are included in $p \in \mathcal{P}$, and we let $V_X \coloneqq \bigcup_{p \in X} V_p$ for any $X \subseteq \mathcal{P}$. If $X \subseteq \mathcal{P}$, then $V_X \cup \{\mathbf{r}\}$ forms a subtree in $\mathbf{T}$. Conversely, if $S \cup \{\mathbf{r}\}$ forms a subtree in $\mathbf{T}$ ($S \subseteq V$), there exists $X \subseteq \mathcal{P}$ such that $V_X = S$. Thus STKP (1) can be transformed into the following maximization

Algorithm 1 Greedy

$U \leftarrow \mathcal{P}$, $X \leftarrow \emptyset$
while $U \neq \emptyset$ **do**
  $p = \mathrm{argmax}_{p' \in U} \frac{f(p'|X)}{c(p'|X)}$
  if $c(X + p) \leq L$ **then**
    $X \leftarrow X + p$
  end if
  $U \leftarrow U - p$
end while
$\hat{p} = \mathrm{argmax}_{p' \in \mathcal{P}} f(p')$
return $Y = \mathrm{argmax}_{X' \in \{X, \hat{p}\}} f(X')$

problem on $\mathcal{P}$:

$$\begin{aligned} \underset{X \subseteq \mathcal{P}}{\text{maximize}} \quad & f(X) \coloneqq g(V_X) \\ \text{subject to} \quad & c(X) \coloneqq \sum_{v \in V_X} \ell_v \leq L. \end{aligned} \tag{2}$$

We here suppose that $c(p) \leq L$ holds for all $p \in \mathcal{P}$; any $p \in \mathcal{P}$ violating this condition can be removed in advance since no feasible solution includes such p. The set functions f and c are monotone submodular functions defined on $\mathcal{P}$ (see the Appendix), and thus the above problem is SCSKP. Figure 1 illustrates how to transform STKP into SCSKP.

4.2 Greedy Algorithm

We provide a greedy algorithm for SCSKP (2). In what follows, given any $X, Y \subseteq \mathcal{P}$, we define the binary operators $+$ and $-$ on $\mathcal{P}$ as

$$\begin{aligned} X + Y &\coloneqq \{p \in \mathcal{P} : p \in X \text{ and/or } p \in Y\}, \\ X - Y &\coloneqq \{p \in \mathcal{P} : p \in X \text{ and } p \notin Y\}. \end{aligned}$$

Namely, they are the union and subtraction of two subsets defined on $\mathcal{P}$. We sometimes abuse the notation and regard $p \in \mathcal{P}$ as a subset of $\mathcal{P}$; for example, we let $X + p = X + \{p\}$ for any $X \subseteq \mathcal{P}$ and $p \in \mathcal{P}$. Furthermore, we define $f(X \mid Y) \coloneqq f(X + Y) - f(Y)$ and $c(X \mid Y) \coloneqq c(X + Y) - c(Y)$ for any $X, Y \subseteq \mathcal{P}$.

Algorithm 1 presents a concise description of the greedy algorithm for SCSKP (2). In practice, function evaluations in the above greedy algorithm can be reduced by using the technique provided in (Leskovec et al., 2007) with some modifications. The resulting greedy algorithm requires at most $O(L|V|)$ function evaluations.

Different from the naive greedy algorithm explained in Section 3, the above greedy algorithm is performed on the set of all rooted paths, $\mathcal{P}$. Thus, even if beneficial vertices are far from $\mathbf{r}$, rooted paths that include such beneficial vertices are considered as candidates to be chosen in each iteration. As a result, we get the following performance guarantee for Algorithm 1; we define λ_i as the number of leaves in T_i for $i \in [N]$, and we let $\lambda \coloneqq \max_{i \in [N]} \lambda_i$.

Theorem 1. *If $Y \subseteq \mathcal{P}$ is the output of Algorithm 1 and $X^* \subseteq \mathcal{P}$ is an optimal solution for SCSKP (2), then we have $f(Y) \geq \frac{1}{2}(1 - e^{-1/\lambda}) f(X^*)$.*

Proof. See the Appendix. □

In other words, Algorithm 1 enjoys a $\frac{1}{2}(1 - e^{-1/\lambda})$-approximation guarantee. Notably, if the values of λ_i ($i \in [N]$) are bounded by a small constant for all N sentences, the performance guarantee does not deteriorate no matter how many sentences are in the document data. This implies that our method works effectively for summarizing large-scale document data that comprises many sentences.

4.3 Relation with Existing Work

We first see some existing results. For submodular maximization with a size constraint (i.e., $|S|$ must be at most a certain value), the greedy algorithm has been proved to achieve $(1 - e^{-1})$-approximation (Nemhauser et al., 1978). Khuller et al. (1999) studied the maximum coverage problem with a knapsack constraint, and proved that the greedy algorithm achieves $(1 - e^{-1/2})$-approximation. They also showed that $(1 - e^{-1})$-approximation can be obtained by executing the greedy algorithm $O(|V|^3)$ times, and this result was generalized to the case with a submodular objective function (Sviridenko, 2004). The greedy algorithm for submodular maximization with a knapsack constraint is known to achieve $\frac{1}{2}(1 - e^{-1})$-approximation (Leskovec et al., 2007). Lin and Bilmes (2010) stated that $(1 - e^{-1/2})$-approximation can be obtained with the greedy algorithm, but a mistake in their proof was pointed out by Morita et al. (2013).[2]

Unlike the above problem settings, submodular maximization with a tree constraint has only a few literatures. Krause et al. (2006) studied submodular maximization over a graph with a knapsack and tree constraints, but their algorithm, called *pSPIEL*,

[2]Probably, this mistake can be fixed with the techniques used in (Khuller et al., 1999).

requires a complicated preprocessing step and imposes some assumptions on the problem, which do not hold in most summarization tasks. Iyer and Bilmes (2013) addressed SCSKP, a more general problem setting. Their algorithm is, however, more expensive than the greedy algorithm, and it only achieves a *bi-criterion* approximation guarantee (i.e., not only the objective value but also the magnitude of constraint violation is approximated); if we use this algorithm for document summarization, a resulting summary may violate the length limitation.

We turn to the relation between our result and the existing ones. We consider submodular maximization with a knapsack constraint. This problem can be formulated as an STKP on a *star graph*, whose vertex and edge sets are $\{\mathbf{r}, r_1, \dots, r_N\}$ and $\{(\mathbf{r}, r_1), \dots, (\mathbf{r}, r_N)\}$, respectively (i.e., every leaf corresponds to an element in $V = \{r_1, \dots, r_N\}$). In this case, we have $\lambda = 1$, and thus we obtain a $\frac{1}{2}(1 - e^{-1})$-approximation guarantee, matching the result of (Leskovec et al., 2007).[3]

5 Objective Functions

As presented in (Lin and Bilmes, 2011), many objective functions used for document summarization are known to be monotone and submodular. Below we list examples of the functions that will be used in the experiments.

Coverage Function

To use the coverage function is a simple but powerful approach to document summarization, and so it appears in many existing works (e.g., (Filatova and Hatzivassiloglou, 2004; Takamura and Okumura, 2009a; Berg-Kirkpatrick et al., 2011)). Let M be the number of distinct words in the document data, and suppose that they are indexed with $j \in [M]$. We let w_j ($j \in [M]$) be the weight value of the j-th word. Given summary $S \subseteq V$, the coverage function $\mathrm{COV}(S)$ is defined as follows:

$$\mathrm{COV}(S) := \sum_{j=1}^{M} w_j z_j,$$

where $z_j \in \{0, 1\}$ is a binary decision variable that indicates whether the j-th word is included in S or not; more precisely, $z_j = 1$ if and only if at least one textual unit in S contains the j-th word.

[3] We also tried to obtain an approximation guarantee that corresponds to the $(1 - e^{-1/2})$-approximation (Khuller et al., 1999; Lin and Bilmes, 2010), but it was not straightforward to apply their techniques to our case.

Coverage Function with Rewords

A summary obtained with the above coverage function often consists of many overly-compressed sentences, which typically leads to low readability. Morita et al. (2013) addressed this problem by adding a positive reward term to the coverage function. Given summary S, let $b_{r_i} \in \{0, 1\}$ ($i \in [N]$) be a binary decision variable that indicates whether r_i, the root node of sentence tree T_i, is included in S or not. Note that, if $S \cup \{\mathbf{r}\}$ forms a rooted subtree in $\mathbf{T}$, we have $b_{r_i} = 1$ if and only if at least one textual unit in the i-th sentence appears in S. With these additional variables, the modified coverage function can be written as

$$\mathrm{COVR}(S) := \mathrm{COV}(S) + \gamma \left(\sum_{v \in S} \ell_v - \sum_{i=1}^{N} b_{r_i} \right),$$

where $\gamma \geq 0$ is a parameter that controls the rate of sentence compression. The value of $\sum_{i=1}^{N} b_{r_i}$ is equal to the number of sentences whose textual unit(s) is used in S. Therefore, a summary that consists of fewer sentences tends to get a higher objective value, thus enhancing readability.

ROUGE

ROUGE (Lin, 2004) is widely used for summarization evaluation, and it is known to be highly correlated with human evaluation. Furthermore, ROUGE is known to be monotone and submodular (Lin and Bilmes, 2011). Specifically, given K reference summaries $R_1, \dots, R_K \subseteq V$ and function $\mathrm{C}_e(S)$, which counts the number of times that n-gram e occurs in summary $S \subseteq V$, the ROUGE_n score function is defined as

$$\mathrm{ROUGE}_n(S) := \frac{\sum_{k=1}^{K} \sum_{e \in R_k} \min\{\mathrm{C}_e(S), \mathrm{C}_e(R_k)\}}{\sum_{k=1}^{K} \sum_{e \in R_k} \mathrm{C}_e(R_k)}.$$

6 Experiments

We applied our method to compressive summarization tasks with the three kinds of objective functions: the coverage function, the one with rewards, and ROUGE_1. To benchmark our method, we also applied the ILP-based method to the tasks. These two methods were compared in terms of achieved approximation ratios, ROUGE_1 scores, and running times.

Objective function	Method	Approximation ratio	ROUGE$_1$	Time (ms)
Coverage	Greedy	0.964	**0.347**	**1.34**
	ILP	**1.00**	0.346	231
Coverage with rewards	Greedy	0.967	**0.334**	**1.44**
	ILP	**1.00**	0.332	552
ROUGE$_1$	Greedy	0.985	0.468	**0.759**
	ILP (oracle)	**1.00**	**0.494**	92.1

Table 1: Approximation ratios, ROUGE$_1$ scores, and running times for our method (Greedy) and the ILP-based method (ILP); the average values over the 50 topics are presented. The two methods are applied to compressive summarization tasks with three types of objective functions: Coverage, Coverage with rewards, and ROUGE$_1$. Summaries obtained with the ILP-based method and ROUGE$_1$ objective function are oracle summaries.

6.1 Settings

In the following experiments, we regard V as the set of all chunks in the document data. For each chunk $v \in V$, we let ℓ_v be the number of words contained in v, and we set the length limitation, L, to 100. For the coverage function and the one with rewards, the weight values w_j ($j \in [M]$) were estimated by logistic regression (Yih et al., 2007) trained on the DUC-2003 dataset. For the coverage function with rewards, we set the parameter, γ, to 0.9.

The experiments were conducted on the DUC-2004 dataset for multiple document summarization evaluation, which is a commonly used benchmark dataset. The dataset consists of 50 topics, each of which has 10 newspaper articles. The dependency trees for this dataset were obtained as follows: We first applied the Stanford parser (de Marneffe et al., 2006) to all sentences in the dataset in order to obtain dependency relations between words. We then applied Filippova's rules (Filippova and Strube, 2008; Filippova and Altun, 2013) to the obtained relations so as to construct trees that represent dependency relations between chunks. To obtain summaries with high readability, we treated a set of chunks connected with certain relations (e.g., subject–object) as a single chunk.

Our algorithm was implemented in C++ and compiled with `GCC version 4.8.5`. The ILP-based method solved ILPs with `CPLEX ver. 12.5.1.0`, a widely used commercial ILP solver. The details of ILP formulations for the three objective functions are presented in the Appendix. All experiments were conducted on a Linux machine (CPU: Intel Xeon E5-2620 v4 2.10GHz and 32GB RAM).

6.2 Results

Table 1 summarizes the comparisons of the achieved approximation ratios, ROUGE$_1$ scores and running times. The ILP-based method are always optimal in terms of objective values (i.e., 100%-approximation is attained), and our method achieved more than 95%-approximation. We observed that the maximum number, λ, of leaves in a sentence tree was about 22 on average, which leads to a 2.2%-approximation guarantee of our algorithm. Therefore, our method empirically performs much better than the theoretical guarantee; this is often the case with the greedy algorithm for submodular maximization problems, in particular when the problems have complex constraints. The ROUGE$_1$ scores of our method are comparable to those of the ILP-based method. With the coverage function and the one with rewards, it happened that our method attained slightly higher ROUGE$_1$ scores than those of ILP-based methods;[4] note that this result is possible since the objective values and ROUGE$_1$ scores are not completely correlated. The results on approximation ratios and ROUGE$_1$ scores imply that our method compares favorably with the ILP-based method in terms of empirical performance. With regard to the running times, our method substantially outperformed the ILP-based method. Specifically, our method was about 170, 380, and 120 times faster than the ILP-based one for the coverage function, the one with rewards, and the ROUGE$_1$ objective function, respectively.

Table 2 shows examples of the summaries obtained by our method and the ILP-based method; both methods used the coverage function with rewards as an objective function. We see that

[4] Similar results were observed in (Takamura and Okumura, 2009a).

Greedy: Yeltsin suffered from disease and had a heart attack followed by multiple bypass surgery in the months. Russian President Boris Yeltsin cut short a trip to Central Asia on Monday due to a respiratory infection that revived questions about his health and ability to lead Russia through a sustained economic crisis. Doctors insisted that Yeltsin fly home ahead of schedule. The prime minister reiterated Wednesday that Yeltsin has plans to resign early elections. Russia's Constitutional Court opened hearings Thursday on whether Boris Yeltsin can seek a term. Sources in Primakov's office said the cancellation was due to concerns.
ILP: Russian President Boris Yeltsin cut short a trip to a respiratory infection that revived questions about his health and ability to lead Russia through a economic crisis. Yeltsin was spending outside Moscow his spokesman Dmitry Yakushkin told reporters. Doctors insisted Monday that Yeltsin fly home from Central Asia ahead of schedule because he was suffering. Yeltsin falls ill speculation arises. The prime minister reiterated Wednesday that Yeltsin has plans to resign early elections. Russia's Constitutional Court opened hearings Thursday on whether Boris Yeltsin can seek a term. Sources in Primakov's office said the cancellation was due to concerns.

Table 2: Summaries obtained with our greedy method (upper) and the ILP-based method (lower) for topic:D31032. To obtain these summaries, both methods used the coverage function with rewards as an objective function.

both methods successfully created informative summaries that preserve original dependency relations. The readability of obtained summaries is unfortunately not high enough. Note that not only our method but also most compressive summarization methods suffer this problem; in fact, there is little difference between the two summaries obtained with our method and the optimal ILP-based method with regard to readability. To conclude, the empirical performance of our method matches that of the ILP-based method, while running about 100 to 400 times faster.

7 Conclusion and Discussion

We proposed a fast greedy method for compressive summarization. Our method works with any monotone submodular objective function; examples of such functions include the coverage function, ROUGE, and many others. The $\frac{1}{2}(1 - e^{-1/\lambda})$-approximation guarantee of our method was proved, which generalizes the $\frac{1}{2}(1 - e^{-1})$-approximation for submodular maximization with a knapsack constraint. Experiments showed that our greedy method empirically achieves more than 95%-approximation and that it runs about 100 to 400 times faster than the ILP-based method implemented with `CPLEX`. With the coverage function and its variant, our method attained as high ROUGE_1 scores as the ILP-based method.

As mentioned above, current compressive summarization systems often fail to achieve high readability, and one possible approach to this problem is to develop better objective functions. Since our method is applicable to various monotone submodular objective functions and can find almost optimal solutions efficiently, our method would be helpful in testing the performance of newly proposed objective functions. Thus we believe that our method is useful for advancing the study into compressive summarization.

Interestingly, STKP can be seen as a variant of *DR-submodular* maximization (Soma and Yoshida, 2017), which is a submodular maximization problem defined over integer lattice. The constraint that appears in DR-submodular maximization is somewhat easier to deal with than that of our problem; exploiting this, Soma and Yoshida (2017) developed a polynomial-time algorithm that achieves roughly $\frac{1}{2}$-approximation. The techniques studied in this field may be useful to develop better algorithms for STKP, which we leave for future work.

References

Miguel Almeida and Andre Martins. 2013. Fast and robust compressive summarization with dual decomposition and multi-task learning. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 196–206. http://www.aclweb.org/anthology/P13-1020.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 481–490. http://www.aclweb.org/anthology/P11-1049.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 484–494. https://doi.org/10.18653/v1/P16-1046.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*. European Language Resources Association, pages 449–454. http://www.aclweb.org/anthology/L06-1260.

Harold P. Edmundson. 1969. New methods in automatic extracting. *Journal of the ACM* 16(2):264–285. https://doi.org/10.1145/321510.321519.

Elena Filatova and Vasileios Hatzivassiloglou. 2004. A formal model for information selection in multi-sentence text extraction. In *Proceedings of the 20th International Conference on Computational Linguistics*. http://www.aclweb.org/anthology/C04-1057.

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1481–1491. http://www.aclweb.org/anthology/D13-1155.

Katja Filippova and Michael Strube. 2008. Dependency tree based sentence compression. In *Proceedings of the 5th International Natural Language Generation Conference*. Association for Computational Linguistics, pages 25–32. http://www.aclweb.org/anthology/W08-1105.

Tsutomu Hirao, Masaaki Nishino, and Masaaki Nagata. 2017a. Oracle summaries of compressive summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 275–280. https://doi.org/10.18653/v1/P17-2043.

Tsutomu Hirao, Masaaki Nishino, Jun Suzuki, and Masaaki Nagata. 2017b. Enumeration of extractive oracle summaries. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, pages 386–396. http://www.aclweb.org/anthology/E17-1037.

Rishabh Iyer and Jeff Bilmes. 2013. Submodular optimization with submodular cover and submodular knapsack constraints. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. Curran Associates Inc., pages 2436–2444. http://dl.acm.org/citation.cfm?id=2999792.2999884.

Litton J Kurisinkel, Pruthwik Mishra, Vigneshwaran Muralidaran, Vasudeva Varma, and Dipti Misra Sharma. 2016. Non-decreasing sub-modular function for comprehensible summarization. In *Proceedings of the NAACL Student Research Workshop*. Association for Computational Linguistics, pages 94–101. https://doi.org/10.18653/v1/N16-2014.

Samir Khuller, Anna Moss, and Joseph S. Naor. 1999. The budgeted maximum coverage problem. *Information Processing Letters* 70(1):39–45. https://doi.org/10.1016/S0020-0190(99)00031-9.

Yuta Kikuchi, Tsutomu Hirao, Hiroya Takamura, Manabu Okumura, and Masaaki Nagata. 2014. Single document summarization based on nested tree structure. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 315–320. https://doi.org/10.3115/v1/P14-2052.

Andreas Krause, Carlos Guestrin, Anupam Gupta, and Jon Kleinberg. 2006. Near-optimal sensor placements: Maximizing information while minimizing communication cost. In *Proceedings of the 5th International Conference on Information Processing in Sensor Networks*. ACM, pages 2–10. https://doi.org/10.1145/1127777.1127782.

Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. 2007. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pages 420–429. https://doi.org/10.1145/1281192.1281239.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Proceedings of Workshop on Text Summarization Branches Out*. pages 74–81. http://www.aclweb.org/anthology/W04-1013.

Hui Lin and Jeff Bilmes. 2010. Multi-document summarization via budgeted maximization of submodular functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 912–920. http://www.aclweb.org/anthology/N10-1134.

Hui Lin and Jeff Bilmes. 2011. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Lin-*

guistics: Human Language Technologies. Association for Computational Linguistics, pages 510–520. http://www.aclweb.org/anthology/P11-1052.

Hans P. Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of Research and Development* 2(2):159–165. https://doi.org/10.1147/rd.22.0159.

Ryan McDonald. 2007. A study of global inference algorithms in multi-document summarization. In *Proceedings of the 29th European Conference on IR Research*. Springer-Verlag, pages 557–564. http://dl.acm.org/citation.cfm?id=1763653.1763720.

Hajime Morita, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2013. Subtree extractive summarization via submodular maximization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1023–1032. http://www.aclweb.org/anthology/P13-1101.

George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. 1978. An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming* 14(1):265–294. https://doi.org/10.1007/BF01588971.

Maxime Peyrard and Judith Eckle-Kohler. 2017. Supervised learning of automatic pyramid for optimization-based multi-document summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1084–1094. https://doi.org/10.18653/v1/P17-1100.

Tasuku Soma and Yuichi Yoshida. 2017. Non-monotone DR-submodular function maximization. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*. pages 898–904. https://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14483.

Maxim Sviridenko. 2004. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters* 32(1):41–43. https://doi.org/10.1016/S0167-6377(03)00062-2.

Hiroya Takamura and Manabu Okumura. 2009a. Text summarization model based on maximum coverage problem and its variant. In *Proceedings of the 12th Conference of the European Chapter of the ACL*. Association for Computational Linguistics, pages 781–789. http://www.aclweb.org/anthology/E09-1089.

Hiroya Takamura and Manabu Okumura. 2009b. Text summarization model based on the budgeted median problem. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*. ACM, pages 1589–1592. https://doi.org/10.1145/1645953.1646179.

Wen-tau Yih, Joshua Goodman, Lucy Vanderwende, and Hisami Suzuki. 2007. Multi-document summarization by maximizing informative content-words. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann Publishers Inc., pages 1776–1782. http://dl.acm.org/citation.cfm?id=1625275.1625563.

Ranking Sentences for Extractive Summarization with Reinforcement Learning

Shashi Narayan **Shay B. Cohen** **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh, EH8 9AB
shashi.narayan@ed.ac.uk, {scohen,mlap}@inf.ed.ac.uk

Abstract

Single document summarization is the task of producing a shorter version of a document while preserving its principal information content. In this paper we conceptualize extractive summarization as a sentence ranking task and propose a novel training algorithm which globally optimizes the ROUGE evaluation metric through a reinforcement learning objective. We use our algorithm to train a neural summarization model on the CNN and DailyMail datasets and demonstrate experimentally that it outperforms state-of-the-art extractive and abstractive systems when evaluated automatically and by humans.[1]

1 Introduction

Automatic summarization has enjoyed wide popularity in natural language processing due to its potential for various information access applications. Examples include tools which aid users navigate and digest web content (e.g., news, social media, product reviews), question answering, and personalized recommendation engines. Single document summarization — the task of producing a shorter version of a document while preserving its information content — is perhaps the most basic of summarization tasks that have been identified over the years (see Nenkova and McKeown, 2011 for a comprehensive overview).

Modern approaches to single document summarization are data-driven, taking advantage of the success of neural network architectures and their ability to learn continuous features without recourse to preprocessing tools or linguistic annotations. *Abstractive* summarization involves various text rewriting operations (e.g., substitution, deletion, reordering) and has been recently framed as a sequence-to-sequence problem (Sutskever et al., 2014). Central in most approaches (Rush et al., 2015; Chen et al., 2016; Nallapati et al., 2016; See et al., 2017; Tan and Wan, 2017; Paulus et al., 2017) is an encoder-decoder architecture modeled by recurrent neural networks. The encoder reads the source sequence into a list of continuous-space representations from which the decoder generates the target sequence. An attention mechanism (Bahdanau et al., 2015) is often used to locate the region of focus during decoding.

Extractive systems create a summary by identifying (and subsequently concatenating) the most important sentences in a document. A few recent approaches (Cheng and Lapata, 2016; Nallapati et al., 2017; Narayan et al., 2017; Yasunaga et al., 2017) conceptualize extractive summarization as a sequence labeling task in which each label specifies whether each document sentence should be included in the summary. Existing models rely on recurrent neural networks to derive a meaning representation of the document which is then used to label each sentence, taking the previously labeled sentences into account. These models are typically trained using cross-entropy loss in order to maximize the likelihood of the ground-truth labels and do not necessarily *learn to rank* sentences based on their importance due to the absence of a ranking-based objective. Another discrepancy comes from the mismatch between the learning objective and the evaluation criterion, namely ROUGE (Lin and Hovy, 2003), which takes the entire summary into account.

In this paper we argue that cross-entropy training is not optimal for extractive summarization. Models trained this way are prone to generating verbose summaries with unnecessarily long sentences and redundant information. We propose to overcome these difficulties by globally optimizing the ROUGE evaluation metric and learning to rank sentences for summary generation through a reinforcement learning objective. Similar to previous work (Cheng and Lapata, 2016; Narayan et al., 2017; Nallapati et al., 2017), our neural summarization model consists of a hierarchical docu-

[1] Our code and data are available here: https://github.com/shashiongithub/Refresh.

Proceedings of NAACL-HLT 2018, pages 1747–1759
New Orleans, Louisiana, June 1 - 6, 2018.

ment encoder and a hierarchical sentence extractor. During training, it combines the maximum-likelihood cross-entropy loss with rewards from policy gradient reinforcement learning to directly optimize the evaluation metric relevant for the summarization task. We show that this global optimization framework renders extractive models better at discriminating among sentences for the final summary; a sentence is ranked high for selection if it often occurs in high scoring summaries.

We report results on the CNN and DailyMail news highlights datasets (Hermann et al., 2015) which have been recently used as testbeds for the evaluation of neural summarization systems. Experimental results show that when evaluated automatically (in terms of ROUGE), our model outperforms state-of-the-art extractive *and* abstractive systems. We also conduct two human evaluations in order to assess (a) which type of summary participants prefer (we compare extractive and abstractive systems) and (b) how much key information from the document is preserved in the summary (we ask participants to answer questions pertaining to the content in the document by reading system summaries). Both evaluations overwhelmingly show that human subjects find our summaries more informative and complete.

Our contributions in this work are three-fold: a novel application of reinforcement learning to sentence ranking for extractive summarization; corroborated by analysis and empirical results showing that cross-entropy training is not well-suited to the summarization task; and large scale user studies following two evaluation paradigms which demonstrate that state-of-the-art abstractive systems lag behind extractive ones when the latter are globally trained.

2 Summarization as Sentence Ranking

Given a document D consisting of a sequence of sentences $(s_1, s_2, ..., s_n)$, an extractive summarizer aims to produce a summary $\mathcal{S}$ by selecting m sentences from D (where $m < n$). For each sentence $s_i \in D$, we predict a label $y_i \in \{0,1\}$ (where 1 means that s_i should be included in the summary) and assign a score $p(y_i|s_i, D, \theta)$ quantifying s_i's relevance to the summary. The model learns to assign $p(1|s_i, D, \theta) > p(1|s_j, D, \theta)$ when sentence s_i is more relevant than s_j. Model parameters are denoted by θ. We estimate $p(y_i|s_i, D, \theta)$ using a neural network model and assemble a summary $\mathcal{S}$ by selecting m sentences with top $p(1|s_i, D, \theta)$ scores.

Our architecture resembles those previously proposed in the literature (Cheng and Lapata, 2016; Nallapati et al., 2017; Narayan et al., 2017). The main components include a sentence encoder, a document encoder, and a sentence extractor (see the left block of Figure 1) which we describe in more detail below.

Sentence Encoder A core component of our model is a convolutional sentence encoder which encodes sentences into continuous representations. In recent years, CNNs have proven useful for various NLP tasks (Collobert et al., 2011; Kim, 2014; Kalchbrenner et al., 2014; Zhang et al., 2015; Lei et al., 2015; Kim et al., 2016; Cheng and Lapata, 2016) because of their effectiveness in identifying salient patterns in the input (Xu et al., 2015). In the case of summarization, CNNs can identify named-entities and events that correlate with the gold summary.

We use temporal narrow convolution by applying a kernel filter K of width h to a window of h words in sentence s to produce a new feature. This filter is applied to each possible window of words in s to produce a feature map $f \in R^{k-h+1}$ where k is the sentence length. We then apply max-pooling over time over the feature map f and take the maximum value as the feature corresponding to this particular filter K. We use multiple kernels of various sizes and each kernel multiple times to construct the representation of a sentence. In Figure 1, kernels of size 2 (red) and 4 (blue) are applied three times each. Max-pooling over time yields two feature lists f^{K_2} and $f^{K_4} \in R^3$. The final sentence embeddings have six dimensions.

Document Encoder The document encoder composes a sequence of sentences to obtain a document representation. We use a recurrent neural network with Long Short-Term Memory (LSTM) cells to avoid the vanishing gradient problem when training long sequences (Hochreiter and Schmidhuber, 1997). Given a document D consisting of a sequence of sentences $(s_1, s_2, \ldots, s_n)$, we follow common practice and feed sentences in reverse order (Sutskever et al., 2014; Li et al., 2015; Filippova et al., 2015; Narayan et al., 2017). This way we make sure that the network also considers the top sentences of the document which are particularly important for summarization (Rush et al., 2015; Nallapati et al., 2016).

Sentence Extractor Our sentence extractor sequentially labels each sentence in a document with 1 (relevant for the summary) or 0 (otherwise).

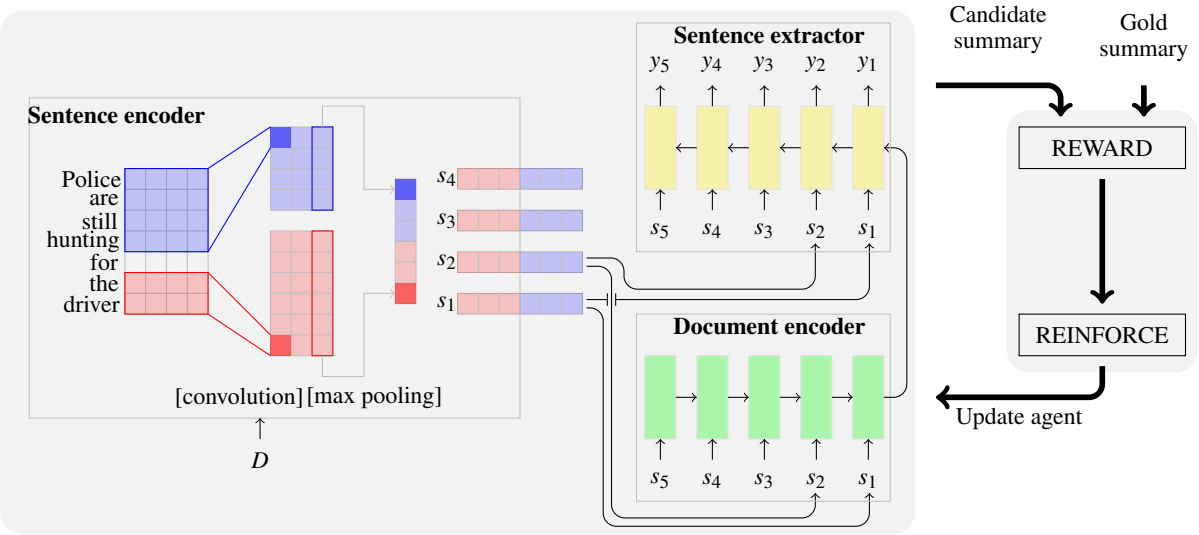


Figure 1: Extractive summarization model with reinforcement learning: a hierarchical encoder-decoder model ranks sentences for their extract-worthiness and a candidate summary is assembled from the top ranked sentences; the REWARD generator compares the candidate against the gold summary to give a reward which is used in the REINFORCE algorithm (Williams, 1992) to update the model.

It is implemented with another RNN with LSTM cells and a softmax layer. At time t_i, it reads sentence s_i and makes a binary prediction, conditioned on the document representation (obtained from the document encoder) and the previously labeled sentences. This way, the sentence extractor is able to identify locally and globally important sentences within the document. We rank the sentences in a document D by $p(y_i = 1|s_i, D, \theta)$, the confidence scores assigned by the softmax layer of the sentence extractor.

We learn to rank sentences by training our network in a reinforcement learning framework, directly optimizing the final evaluation metric, namely ROUGE (Lin and Hovy, 2003). Before we describe our training algorithm, we elaborate on why the maximum-likelihood cross-entropy objective could be deficient for ranking sentences for summarization (Section 3). Then, we define our reinforcement learning objective in Section 4 and show that our new way of training allows the model to better discriminate amongst sentences, i.e., a sentence is ranked higher for selection if it often occurs in high scoring summaries.

3 The Pitfalls of Cross-Entropy Loss

Previous work optimizes summarization models by maximizing $p(y|D,\theta) = \prod_{i=1}^{n} p(y_i|s_i, D, \theta)$, the likelihood of the ground-truth labels $y = (y_1, y_2, ..., y_n)$ for sentences $(s_1, s_2, \ldots, s_n)$, given document D and model parameters θ. This objective can be achieved by minimizing the cross-entropy loss at each decoding step:

$$L(\theta) = -\sum_{i=1}^{n} \log p(y_i|s_i, D, \theta). \quad (1)$$

Cross-entropy training leads to two kinds of discrepancies in the model. The first discrepancy comes from the disconnect between the task definition and the training objective. While MLE in Equation (1) aims to maximize the likelihood of the ground-truth labels, the model is (a) expected to rank sentences to generate a summary and (b) evaluated using ROUGE at test time. The second discrepancy comes from the reliance on ground-truth labels. Document collections for training summarization systems do not naturally contain labels indicating which sentences should be extracted. Instead, they are typically accompanied by abstractive summaries from which sentence-level labels are extrapolated. Cheng and Lapata (2016) follow Woodsend and Lapata (2010) in adopting a rule-based method which assigns labels to each sentence in the document *individually* based on their semantic correspondence with the gold summary (see the fourth column in Table 1). An alternative method (Svore et al., 2007; Cao et al., 2016; Nallapati et al., 2017) identifies the set of sentences which *collectively* gives the highest ROUGE with respect to the gold summary. Sentences in this set are labeled with 1 and 0 otherwise (see the column 5 in Table 1).

Labeling sentences individually often generates too many positive labels causing the model to

sent. pos.	CNN article Sentences	Sent-level ROUGE	Individual Oracle	Collective Oracle	Multiple Collective Oracle
0	A debilitating, mosquito-borne virus called Chikungunya has made its way to North Carolina, health officials say.	21.2	1	1	(0,11,13) : 59.3 (0,13) : 57.5
1	It's the state's first reported case of the virus.	18.1	1	0	(11,13) : 57.2
2	The patient was likely infected in the Caribbean, according to the Forsyth County Department of Public Health.	11.2	1	0	(0,1,13) : 57.1 (1,13) : 56.6
3	Chikungunya is primarily found in Africa, East Asia and the Caribbean islands, but the Centers for Disease Control and Prevention has been watching the virus,+ for fear that it could take hold in the United States – much like West Nile did more than a decade ago.	35.6	1	0	(3,11,13) : 55.0 (13) : 54.5 (0,3,13) : 54.2
4	The virus, which can cause joint pain and arthritis-like symptoms, has been on the U.S. public health radar for some time.	16.7	1	0	(3,13) : 53.4 (1,3,13) : 52.9
5	About 25 to 28 infected travelers bring it to the United States each year, said Roger Nasci, chief of the CDC's Arboviral Disease Branch in the Division of Vector-Borne Diseases.	9.7	0	0	(1,11,13) : 52.0 (0,9,13) : 51.3 (0,7,13) : 51.3
6	"We haven't had any locally transmitted cases in the U.S. thus far," Nasci said.	7.4	0	0	(0,12,13) : 51.0
7	But a major outbreak in the Caribbean this year – with more than 100,000 cases reported – has health officials concerned.	16.4	1	0	(9,11,13) : 50.4 (1,9,13) : 50.1
8	Experts say American tourists are bringing Chikungunya back home, and it's just a matter of time before it starts to spread within the United States.	10.6	0	0	(12,13) : 49.3 (7,11,13) : 47.8
9	After all, the Caribbean is a popular one with American tourists, and summer is fast approaching.	13.9	1	0	(0,10,13) : 47.8 (11,12,13):47.7
10	"So far this year we've recorded eight travel-associated cases, and seven of them have come from countries in the Caribbean where we know the virus is being transmitted," Nasci said.	18.4	1	0	(7,13) : 47.6 (9,13) : 47.5 (1,7,13) : 46.9
11	Other states have also reported cases of Chikungunya.	13.4	0	1	(3,7,13) : 46.0
12	The Tennessee Department of Health said the state has had multiple cases of the virus in people who have traveled to the Caribbean.	15.6	1	0	(3,12,13) : 46.0 (3,9,13) : 45.9
13	The virus is not deadly, but it can be painful, with symptoms lasting for weeks.	54.5	1	1	(10,13) : 45.5
14	Those with weak immune systems, such as the elderly, are more likely to suffer from the virus' side effects than those who are healthier.	5.5	0	0	(4,11,13) : 45.3 ...
	Story Highlights • North Carolina reports first case of mosquito-borne virus called Chikungunya • Chikungunya is primarily found in Africa, East Asia and the Caribbean islands • Virus is not deadly, but it can be painful, with symptoms lasting for weeks				

Table 1: An abridged CNN article (only first 15 out of 31 sentences are shown) and its "story highlights". The latter are typically written by journalists to allow readers to quickly gather information on stories. Highlights are often used as gold standard abstractive summaries in the summarization literature.

overfit the data. For example, the document in Table 1 has 12 positively labeled sentences out of 31 in total (only first 10 are shown). Collective labels present a better alternative since they only pertain to the few sentences deemed most suitable to form the summary. However, a model trained with cross-entropy loss on collective labels will underfit the data as it will only maximize probabilities $p(1|s_i, D, \theta)$ for sentences in this set (e.g., sentences $\{0, 11, 13\}$ in Table 1) and ignore all other sentences. We found that there are many candidate summaries with high ROUGE scores which could be considered during training.

Table 1 (last column) shows candidate summaries ranked according to the mean of ROUGE-1, ROUGE-2, and ROUGE-L F_1 scores. Interestingly, multiple top ranked summaries have reasonably high ROUGE scores. For example, the average ROUGE for the summaries ranked second (0,13), third (11,13), and fourth (0,1,13) is 57.5%, 57.2%, and 57.1%, and all top 16 summaries have ROUGE scores more or equal to 50%. A few sentences are indicative of important content and appear frequently in the summaries: sentence 13 occurs in all summaries except one, while sentence 0 appears in several summaries too. Also note that summaries (11,13) and (1,13) yield better ROUGE scores compared to longer summaries, and may be as informative, yet more concise, alternatives.

These discrepancies render the model less efficient at ranking sentences for the summarization task. Instead of maximizing the likelihood of the ground-truth labels, we could train the model to predict the individual ROUGE score for each sentence in the document and then select the top m sentences with highest scores. But sentences with individual ROUGE scores do not necessarily lead to a high scoring summary, e.g., they may convey overlapping content and form verbose and redundant summaries. For example, sentence 3, despite having a high individual ROUGE score (35.6%), does not occur in any of the top 5 summaries. We next explain how we address these issues using reinforcement learning.

4 Sentence Ranking with Reinforcement Learning

Reinforcement learning (Sutton and Barto, 1998) has been proposed as a way of training sequence-

to-sequence generation models in order to directly optimize the metric used at test time, e.g., BLEU or ROUGE (Ranzato et al., 2015). We adapt reinforcement learning to our formulation of extractive summarization to rank sentences for summary generation. We propose an objective function that combines the maximum-likelihood cross-entropy loss with rewards from policy gradient reinforcement learning to globally optimize ROUGE. Our training algorithm allows to explore the space of possible summaries, making our model more robust to unseen data. As a result, reinforcement learning helps extractive summarization in two ways: (a) it directly optimizes the evaluation metric instead of maximizing the likelihood of the ground-truth labels and (b) it makes our model better at discriminating among sentences; a sentence is ranked high for selection if it often occurs in high scoring summaries.

4.1 Policy Learning

We cast the neural summarization model introduced in Figure 1 in the Reinforcement Learning paradigm (Sutton and Barto, 1998). Accordingly, the model can be viewed as an "agent" which interacts with an "environment" consisting of documents. At first, the agent is initialized randomly, it reads document D and predicts a relevance score for each sentence $s_i \in D$ using "policy" $p(y_i|s_i, D, \theta)$, where θ are model parameters. Once the agent is done reading the document, a summary with labels $\hat{y}$ is sampled out of the ranked sentences. The agent is then given a "reward" r commensurate with how well the extract resembles the gold-standard summary. Specifically, as reward function we use mean F_1 of ROUGE-1, ROUGE-2, and ROUGE-L. Unigram and bigram overlap (ROUGE-1 and ROUGE-2) are meant to assess informativeness, whereas the longest common subsequence (ROUGE-L) is meant to assess fluency. We update the agent using the REINFORCE algorithm (Williams, 1992) which aims to minimize the negative expected reward:

$$L(\theta) = -\mathbb{E}_{\hat{y}\sim\, p_\theta}[r(\hat{y})] \tag{2}$$

where, p_θ stands for $p(y|D, \theta)$. REINFORCE is based on the observation that the expected gradient of a non-differentiable reward function (ROUGE, in our case) can be computed as follows:

$$\nabla L(\theta) = -\mathbb{E}_{\hat{y}\sim\, p_\theta}[r(\hat{y})\nabla \log p(\hat{y}|D, \theta)] \tag{3}$$

While MLE in Equation (1) aims to maximize the likelihood of the training data, the objective in Equation (2) learns to discriminate among sentences with respect to how often they occur in high scoring summaries.

4.2 Training with High Probability Samples

Computing the expectation term in Equation (3) is prohibitive, since there is a large number of possible extracts. In practice, we approximate the expected gradient using a single sample $\hat{y}$ from p_θ for each training example in a batch:

$$\begin{aligned}\nabla L(\theta) &\approx -r(\hat{y})\nabla \log p(\hat{y}|D, \theta) \\ &\approx -r(\hat{y}) \sum_{i=1}^{n} \nabla \log p(\hat{y}_i|s_i, D, \theta)\end{aligned} \tag{4}$$

Presented in its original form, the REINFORCE algorithm starts learning with a random policy which can make model training challenging for complex tasks like ours where a single document can give rise to a very large number of candidate summaries. We therefore limit the search space of $\hat{y}$ in Equation (4) to the set of largest probability samples $\hat{\mathbb{Y}}$. We approximate $\hat{\mathbb{Y}}$ by the k extracts which receive highest ROUGE scores. More concretely, we assemble candidate summaries efficiently by first selecting p sentences from the document which on their own have high ROUGE scores. We then generate all possible combinations of p sentences subject to maximum length m and evaluate them against the gold summary. Summaries are ranked according to F_1 by taking the mean of ROUGE-1, ROUGE-2, and ROUGE-L. $\hat{\mathbb{Y}}$ contains these top k candidate summaries. During training, we sample $\hat{y}$ from $\hat{\mathbb{Y}}$ instead of $p(\hat{y}|D, \theta)$.

Ranzato et al. (2015) proposed an alternative to REINFORCE called MIXER (Mixed Incremental Cross-Entropy Reinforce) which first pretrains the model with the cross-entropy loss using ground truth labels and then follows a curriculum learning strategy (Bengio et al., 2015) to gradually teach the model to produce stable predictions on its own. In our experiments MIXER performed worse than the model of Nallapati et al. (2017) just trained on collective labels. We conjecture that this is due to the unbounded nature of our ranking problem. Recall that our model assigns relevance scores to sentences rather than words. The space of sentential representations is vast and fairly unconstrained compared to other prediction tasks operating with fixed vocabularies (Li et al., 2016; Paulus et al., 2017; Zhang and Lapata, 2017). Moreover, our approximation of the gradient allows the model to

converge much faster to an optimal policy. Advantageously, we do not require an online reward estimator, we pre-compute $\hat{\mathbb{Y}}$, which leads to a significant speedup during training compared to MIXER (Ranzato et al., 2015) and related training schemes (Shen et al., 2016).

5 Experimental Setup

In this section we present our experimental setup for assessing the performance of our model which we call REFRESH as a shorthand for **RE**in**F**o**R**cement Learning-based **E**xtractive **S**ummarization. We describe our datasets, discuss implementation details, our evaluation protocol, and the systems used for comparison.

Summarization Datasets We evaluated our models on the CNN and DailyMail news highlights datasets (Hermann et al., 2015). We used the standard splits of Hermann et al. (2015) for training, validation, and testing (90,266/1,220/1,093 documents for CNN and 196,961/12,148/10,397 for DailyMail). We did not anonymize entities or lower case tokens. We followed previous studies (Cheng and Lapata, 2016; Nallapati et al., 2016, 2017; See et al., 2017; Tan and Wan, 2017) in assuming that the "story highlights" associated with each article are gold-standard abstractive summaries. During training we use these to generate high scoring extracts and to estimate rewards for them, but during testing, they are used as reference summaries to evaluate our models.

Implementation Details We generated extracts by selecting three sentences ($m = 3$) for CNN articles and four sentences ($m = 4$) for DailyMail articles. These decisions were informed by the fact that gold highlights in the CNN/DailyMail validation sets are 2.6/4.2 sentences long. For both datasets, we estimated high-scoring extracts using 10 document sentences ($p = 10$) with highest ROUGE scores. We tuned the initialization parameter k for $\hat{\mathbb{Y}}$ on the validation set: we found that our model performs best with $k = 5$ for the CNN dataset and $k = 15$ for the DailyMail dataset.

We used the One Billion Word Benchmark corpus (Chelba et al., 2013) to train word embeddings with the skip-gram model (Mikolov et al., 2013) using context window size 6, negative sampling size 10, and hierarchical softmax 1. Known words were initialized with pre-trained embeddings of size 200. Embeddings for unknown words were initialized to zero, but estimated during training.

LEAD	• A SkyWest Airlines flight made an emergency landing in Buffalo, New York, on Wednesday after a passenger lost consciousness, officials said. • The passenger received medical attention before being released, according to Marissa Snow, spokeswoman for SkyWest. • She said the airliner expects to accommodate the 75 passengers on another aircraft to their original destination – Hartford, Connecticut – later Wednesday afternoon.
See et al.	• Skywest Airlines flight made an emergency landing in Buffalo, New York, on Wednesday after a passenger lost consciousness. • She said the airliner expects to accommodate the 75 passengers on another aircraft to their original destination – Hartford, Connecticut.
REFRESH	• A SkyWest Airlines flight made an emergency landing in Buffalo, New York, on Wednesday after a passenger lost consciousness, officials said. • The passenger received medical attention before being released, according to Marissa Snow, spokeswoman for SkyWest. • The Federal Aviation Administration initially reported a pressurization problem and said it would investigate.
GOLD	• FAA backtracks on saying crew reported a pressurization problem • One passenger lost consciousness • The plane descended 28,000 feet in three minutes
Q_1	Who backtracked on saying crew reported a pressurization problem? (*FAA*)
Q_2	How many passengers lost consciousness in the incident? (*One*)
Q_3	How far did the plane descend in three minutes? (*28,000 feet*)

Figure 2: Summaries produced by the LEAD baseline, the abstractive system of See et al. (2017) and REFRESH for a CNN (test) article. GOLD presents the human-authored summary; the bottom block shows manually written questions using the gold summary and their answers in parentheses.

Sentences were padded with zeros to a length of 100. For the sentence encoder, we used a list of kernels of widths 1 to 7, each with output channel size of 50 (Kim et al., 2016). The sentence embedding size in our model was 350.

For the recurrent neural network component in the document encoder and sentence extractor, we used a single-layered LSTM network with size 600. All input documents were padded with zeros to a maximum document length of 120. We performed minibatch cross-entropy training with a batch size of 20 documents for 20 training epochs. It took around 12 hrs on a single GPU to train. After each epoch, we evaluated our model on the validation set and chose the best performing model for the test set. During training we used the Adam optimizer (Kingma and Ba, 2015) with initial learning rate 0.001. Our system is implemented in TensorFlow (Abadi et al., 2015).

Evaluation We evaluated summarization quality using F_1 ROUGE (Lin and Hovy, 2003). We report unigram and bigram overlap (ROUGE-1 and ROUGE-2) as a means of assessing informativeness and the longest common subsequence

(ROUGE-L) as a means of assessing fluency.[2] We compared REFRESH against a baseline which simply selects the first m leading sentences from each document (LEAD) and two neural models similar to ours (see left block in Figure 1), both trained with cross-entropy loss. Cheng and Lapata (2016) train on individual labels, while Nallapati et al. (2017) use collective labels. We also compared our model against the abstractive systems of Chen et al. (2016), Nallapati et al. (2016), See et al. (2017), and Tan and Wan (2017).[3]

In addition to ROUGE which can be misleading when used as the only means to assess the informativeness of summaries (Schluter, 2017), we also evaluated system output by eliciting human judgments in two ways. In our first experiment, participants were presented with a news article and summaries generated by three systems: the LEAD baseline, abstracts from See et al. (2017), and extracts from REFRESH. We also included the human-authored highlights.[4] Participants read the articles and were asked to rank the summaries from best (1) to worst (4) in order of informativeness (does the summary capture important information in the article?) and fluency (is the summary written in well-formed English?). We did not allow any ties. We randomly selected 10 articles from the CNN test set and 10 from the DailyMail test set. The study was completed by five participants, all native or proficient English speakers. Each participant was presented with the 20 articles. The order of summaries to rank was randomized per article and the order of articles per participant. Examples of summaries our subjects ranked are shown in Figure 2.

Our second experiment assessed the degree to which our model retains key information from the document following a question-answering (QA) paradigm which has been previously used to evaluate summary quality and text compression (Morris et al., 1992; Mani et al., 2002; Clarke and Lapata, 2010). We created a set of questions based on the gold summary under the assumption that it highlights the most important document content. We then examined whether participants were able to answer these questions by reading system summaries alone without access to the article. The more questions a system can answer, the better it is at summarizing the document as a whole.

We worked on the same 20 documents used in our first elicitation study. We wrote multiple fact-based question-answer pairs for each gold summary without looking at the document. Questions were formulated so as to not reveal answers to subsequent questions. We created 71 questions in total varying from two to six questions per gold summary. Example questions are given in Figure 2. Participants read the summary and answered all associated questions as best they could without access to the original document or the gold summary. Subjects were shown summaries from three systems: the LEAD baseline, the abstractive system of See et al. (2017), and REFRESH. Five participants answered questions for each summary. We used the same scoring mechanism from Clarke and Lapata (2010), i.e., a correct answer was marked with a score of one, partially correct answers with a score of 0.5, and zero otherwise. The final score for a system is the average of all its question scores. Answers were elicited using Amazon's Mechanical Turk crowdsourcing platform. We uploaded data in batches (one system at a time) on Mechanical Turk to ensure that same participant does not evaluate summaries from different systems on the same set of questions.

6 Results

We report results using automatic metrics in Table 2. The top part of the table compares REFRESH against related extractive systems. The bottom part reports the performance of abstractive systems. We present three variants of LEAD, one is computed by ourselves and the other two are reported in Nallapati et al. (2017) and See et al. (2017). Note that they vary slightly due to differences in the preprocessing of the data. We report results on the CNN and DailyMail datasets and their combination (CNN+DailyMail).

Cross-Entropy vs Reinforcement Learning The results in Table 2 show that REFRESH is superior to our LEAD baseline and extractive systems across datasets and metrics. It outperforms

[2] We used pyrouge, a Python package, to compute all ROUGE scores with parameters "-a -c 95 -m -n 4 -w 1.2."

[3] Cheng and Lapata (2016) report ROUGE recall scores on the DailyMail dataset only. We used their code (https://github.com/cheng6076/NeuralSum) to produce ROUGE F_1 scores on both CNN and DailyMail datasets. For other systems, all results are taken from their papers.

[4] We are grateful to Abigail See for providing us with the output of her system. We did not include output from Nallapati et al. (2017), Chen et al. (2016), Nallapati et al. (2016), or Tan and Wan (2017) in our human evaluation study, as these models are trained on a named-entity anonymized version of the CNN and DailyMail datasets, and as result produce summaries which are not comparable to ours. We did not include extracts from Cheng and Lapata (2016) either as they were significantly inferior to LEAD (see Table 2).

Models	CNN			DailyMail			CNN+DailyMail		
	R1	R2	RL	R1	R2	RL	R1	R2	RL
LEAD (ours)	29.1	11.1	25.9	40.7	18.3	37.2	39.6	17.7	36.2
LEAD* (Nallapati et al., 2017)	—	—	—	—	—	—	39.2	15.7	35.5
LEAD (See et al., 2017)	—	—	—	—	—	—	**40.3**	17.7	**36.6**
Cheng and Lapata (2016)	28.4	10.0	25.0	36.2	15.2	32.9	35.5	14.7	32.2
Nallapati et al. (2017)*	—	—	—	—	—	—	39.6	16.2	35.3
REFRESH	**30.4**	**11.7**	**26.9**	**41.0**	**18.8**	**37.7**	40.0	**18.2**	**36.6**
Chen et al. (2016)*	27.1	8.2	18.7	—	—	—	—	—	—
Nallapati et al. (2016)*	—	—	—	—	—	—	35.4	13.3	32.6
See et al. (2017)	—	—	—	—	—	—	39.5	17.3	36.4
Tan and Wan (2017)*	30.3	9.8	20.0	—	—	—	38.1	13.9	34.0

Table 2: Results on the CNN and DailyMail test sets. We report ROUGE-1 (R1), ROUGE-2 (R2), and ROUGE-L (RL) F_1 scores. Extractive systems are in the first block and abstractive systems in the second. Table cells are filled with — whenever results are not available. Models marked with * are not directly comparable to ours as they are based on an anonymized version of the dataset.

the extractive system of Cheng and Lapata (2016) which is trained on individual labels. REFRESH is not directly comparable with Nallapati et al. (2017) as they generate anonymized summaries. Their system lags behind their LEAD baseline on ROUGE-L on the CNN+DailyMail dataset (35.5% vs 35.3%). Also note that their model is trained on collective labels and has a significant lead over Cheng and Lapata (2016). As discussed in Section 3 cross-entropy training on individual labels tends to overgenerate positive labels leading to less informative and verbose summaries.

Extractive vs Abstractive Systems Our automatic evaluation results further demonstrate that REFRESH is superior to abstractive systems (Chen et al., 2016; Nallapati et al., 2016; See et al., 2017; Tan and Wan, 2017) which are all variants of an encoder-decoder architecture (Sutskever et al., 2014). Despite being more faithful to the actual summarization task (hand-written summaries combine several pieces of information from the original document), abstractive systems lag behind the LEAD baseline. Tan and Wan (2017) present a graph-based neural model, which manages to outperform LEAD on ROUGE-1 but falters when higher order ROUGE scores are used. Amongst abstractive systems See et al. (2017) perform best. Interestingly, their system is mostly extractive, exhibiting a small degree of rewriting; it copies more than 35% of the sentences in the source document, 85% of 4-grams, 90% of 3-grams, 95% of bigrams, and 99% of unigrams.

Human Evaluation: System Ranking Table 3 shows, proportionally, how often participants ranked each system, 1st, 2nd, and so on. Perhaps unsurprisingly human-authored summaries are considered best (and ranked 1st 39% of the time). REFRESH is ranked 2nd best followed by LEAD and See et al. (2017) which are mostly ranked in 3rd and 4th places. We carried out pairwise comparisons between all models in Table 3 to assess whether system differences are statistically significant. There is no significant difference between LEAD and See et al. (2017), and REFRESH and GOLD (using a one-way ANOVA with post-hoc Tukey HSD tests; $p < 0.01$). All other differences are statistically significant.

Models	1st	2nd	3rd	4th	QA
LEAD	0.11	0.21	**0.34**	0.33	36.33
See et al. (2017)	0.14	0.18	0.31	**0.36**	28.73
REFRESH	0.35	**0.42**	0.16	0.07	**66.34**
GOLD	**0.39**	0.19	0.18	0.24	—

Table 3: System ranking and QA-based evaluations. Rankings (1st, 2nd, 3rd and 4th) are shown as proportions. Rank 1 is the best and Rank 4, the worst. The column QA shows the percentage of questions that participants answered correctly by reading system summaries.

Human Evaluation: Question Answering The results of our QA evaluation are shown in the last column of Table 3. Based on summaries generated by REFRESH, participants can answer 66.34% of questions correctly. Summaries produced by LEAD and the abstractive system of See et al. (2017) provide answers for 36.33% and 28.73% of the questions, respectively. Differences between systems are all statistically significant ($p < 0.01$) with the exception of LEAD and See et al. (2017).

Although the QA results in Table 3 follow the same pattern as ROUGE in Table 2, differences among systems are now greatly amplified. QA-based evaluation is more focused and a closer reflection of users' information need (i.e., to find out what the article is about), whereas ROUGE simply captures surface similarity (i.e., n-gram overlap)

between output summaries and their references. Interestingly, LEAD is considered better than See et al. (2017) in the QA evaluation, whereas we find the opposite when participants are asked to rank systems. We hypothesize that LEAD is indeed more informative than See et al. (2017) but humans prefer shorter summaries. The average length of LEAD summaries is 105.7 words compared to 61.6 for See et al. (2017).

7 Related Work

Traditional summarization methods manually define features to rank sentences for their salience in order to identify the most important sentences in a document or set of documents (Kupiec et al., 1995; Mani, 2001; Radev et al., 2004; Filatova and Hatzivassiloglou, 2004; Nenkova et al., 2006; Spärck Jones, 2007). A vast majority of these methods learn to score each sentence independently (Barzilay and Elhadad, 1997; Teufel and Moens, 1997; Erkan and Radev, 2004; Mihalcea and Tarau, 2004; Shen et al., 2007; Schilder and Kondadadi, 2008; Wan, 2010) and a summary is generated by selecting top-scored sentences in a way that is not incorporated into the learning process. Summary quality can be improved heuristically, (Yih et al., 2007), via max-margin methods (Carbonell and Goldstein, 1998; Li et al., 2009), or integer-linear programming (Woodsend and Lapata, 2010; Berg-Kirkpatrick et al., 2011; Woodsend and Lapata, 2012; Almeida and Martins, 2013; Parveen et al., 2015).

Recent deep learning methods (Kågebäck et al., 2014; Yin and Pei, 2015; Cheng and Lapata, 2016; Nallapati et al., 2017) learn continuous features without any linguistic preprocessing (e.g., named entities). Like traditional methods, these approaches also suffer from the mismatch between the learning objective and the evaluation criterion (e.g., ROUGE) used at the test time. In comparison, our neural model globally optimizes the ROUGE evaluation metric through a reinforcement learning objective: sentences are highly ranked if they occur in highly scoring summaries.

Reinforcement learning has been previously used in the context of traditional multi-document summarization as a means of selecting a sentence or a subset of sentences from a document cluster. Ryang and Abekawa (2012) cast the sentence selection task as a search problem. Their agent observes a state (e.g., a candidate summary), executes an action (a transition operation that produces a new state selecting a not-yet-selected sentence), and then receives a delayed reward based on tf $*$ idf. Follow-on work (Rioux et al., 2014) extends this approach by employing ROUGE as part of the reward function, while Henß et al. (2015) further experiment with Q-learning. Mollá-Aliod (2017) has adapt this approach to query-focused summarization. Our model differs from these approaches both in application and formulation. We focus solely on extractive summarization, in our case states are documents (not summaries) and actions are relevance scores which lead to sentence ranking (not sentence-to-sentence transitions). Rather than employing reinforcement learning for sentence selection, our algorithm performs sentence ranking using ROUGE as the reward function.

The REINFORCE algorithm (Williams, 1992) has been shown to improve encoder-decoder text-rewriting systems by allowing to directly optimize a non-differentiable objective (Ranzato et al., 2015; Li et al., 2016; Paulus et al., 2017) or to inject task-specific constraints (Zhang and Lapata, 2017; Nogueira and Cho, 2017). However, we are not aware of any attempts to use reinforcement learning for training a sentence ranker in the context of extractive summarization.

8 Conclusions

In this work we developed an extractive summarization model which is globally trained by optimizing the ROUGE evaluation metric. Our training algorithm explores the space of candidate summaries while learning to optimize a reward function which is relevant for the task at hand. Experimental results show that reinforcement learning offers a great means to steer our model towards generating informative, fluent, and concise summaries outperforming state-of-the-art extractive and abstractive systems on the CNN and DailyMail datasets. In the future we would like to focus on smaller discourse units (Mann and Thompson, 1988) rather than individual sentences, modeling compression and extraction jointly.

Acknowledgments We gratefully acknowledge the support of the European Research Council (Lapata; award number 681760), the European Union under the Horizon 2020 SUMMA project (Narayan, Cohen; grant agreement 688139), and Huawei Technologies (Cohen).

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Miguel B. Almeida and André F. T. Martins. 2013. Fast and robust compressive summarization with dual decomposition and multi-task learning. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Sofia, Bulgaria, pages 196–206.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*. San Diego, California, USA.

Regina Barzilay and Michael Elhadad. 1997. Using lexical chains for text summarization. In *Proceedings of the ACL Workshop on Intelligent Scalable Text Summarization*. Madrid, Spain, pages 10–17.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems 28*. pages 1171–1179.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA, pages 481–490.

Ziqiang Cao, Chengyao Chen, Wenjie Li, Sujian Li, Furu Wei, and Ming Zhou. 2016. TGSum: Build tweet guided multi-document summarization dataset. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*. Phoenix, Arizona USA, pages 2906–2912.

Jaime Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Melbourne, Australia, pages 335–336.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. Technical report, Google.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for modeling documents. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*. New York, USA, pages 2754–2760.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Berlin, Germany, pages 484–494.

James Clarke and Mirella Lapata. 2010. Discourse constraints for document compression. *Computational Linguistics* 36(3):411–441.

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

Günes Erkan and Dragomir R. Radev. 2004. LexRank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research* 22(1):457–479.

Elena Filatova and Vasileios Hatzivassiloglou. 2004. Event-based extractive summarization. In *Proceedings of ACL Workshop on Text Summarization Branches Out*. Barcelona, Spain, pages 104–111.

Katja Filippova, Enrique Alfonseca, Carlos A. Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with LSTMs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal, pages 360–368.

Sebastian Henß, Margot Mieskes, and Iryna Gurevych. 2015. A reinforcement learning approach for adaptive single- and multi-document summarization. In *Proceedings of International Conference of the German Society for Computational Linguistics and Language Technology*. Duisburg-Essen, Germany, pages 3–12.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28*. pages 1693–1701.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9(8):1735–1780.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Baltimore, Maryland, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Doha, Qatar, pages 1746–1751.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*. Phoenix, Arizona USA, pages 2741–2749.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*. San Diego, California, USA.

Mikael Kågebäck, Olof Mogren, Nina Tahmasebi, and Devdatt Dubhashi. 2014. Extractive summarization using continuous vector space models. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*. Gothenburg, Sweden, pages 31–39.

Julian Kupiec, Jan Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Seattle, Washington, USA, pages 406–407.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2015. Molding CNNs for text: Non-linear, non-consecutive convolutions. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal, pages 1565–1575.

Jiwei Li, Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. Beijing, China, pages 1106–1115.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas, pages 1192–1202.

Liangda Li, Ke Zhou, Gui-Rong Xue, Hongyuan Zha, and Yong Yu. 2009. Enhancing diversity, coverage and balance for summarization through structure learning. In *Proceedings of the 18th international Conference on World Wide Web*. Madrid, Spain, pages 71–80.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*. Edmonton, Canada, pages 71–78.

Inderjeet Mani. 2001. *Automatic Summarization*. Natural language processing. John Benjamins Publishing Company.

Inderjeet Mani, Gary Klein, David House, Lynette Hirschman, Therese Firmin, and Beth Sundheim. 2002. SUMMAC: A text summarization evaluation. *Natural Language Engineering* 8(1):4368.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical Structure Theory: Toward a functional theory of text organization. *Text* 8(3):243–281.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into texts. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. Barcelona, Spain, pages 404–411.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*. pages 3111–3119.

Diego Mollá-Aliod. 2017. Towards the use of deep reinforcement learning with global policy for query-based extractive summarisation. In *Proceedings of the Australasian Language Technology Association Workshop 2017*. Brisbane, Australia, pages 103–107.

Andrew H. Morris, George M. Kasper, and Dennis A. Adams. 1992. The effects and limitations of automated text condensing on reading comprehension performance. *Information Systems Research* 3(1):17–35.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*. San Francisco, California USA, pages 3075–3081.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*. Berlin, Germany, pages 280–290.

Shashi Narayan, Nikos Papasarantopoulos, Shay B. Cohen, and Mirella Lapata. 2017. Neural extractive summarization with side information. *CoRR* abs/1704.04530.

Ani Nenkova and Kathleen McKeown. 2011. Automatic summarization. *Foundations and Trends in Information Retrieval* 5(2–3):103–233.

Ani Nenkova, Lucy Vanderwende, and Kathleen McKeown. 2006. A compositional context sensitive multi-document summarizer: Exploring the factors that influence summarization. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. pages 573–580.

Rodrigo Nogueira and Kyunghyun Cho. 2017. Task-oriented query reformulation with reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark, pages 585–594.

Daraksha Parveen, Hans-Martin Ramsl, and Michael Strube. 2015. Topical coherence for graph-based extractive summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal, pages 1949–1954.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *CoRR* abs/1705.04304.

Dragomir Radev, Timothy Allison, Sasha Blair-Goldensohn, John Blitzer, Arda Çelebi, Stanko Dimitrov, Elliott Drabek, Ali Hakim, Wai Lam, Danyu Liu, Jahna Otterbacher, Hong Qi, Horacio Saggion, Simone Teufel, Michael Topper, Adam Winkel, and Zhu Zhang. 2004. MEAD — A platform for multidocument multilingual text summarization. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*. Lisbon, Portugal, pages 699–702.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *CoRR* abs/1511.06732.

Cody Rioux, Sadid A. Hasan, and Yllias Chali. 2014. Fear the REAPER: A system for automatic multi-document summarization with reinforcement learning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Doha, Qatar, pages 681–690.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal, pages 379–389.

Seonggi Ryang and Takeshi Abekawa. 2012. Framework of automatic text summarization using reinforcement learning. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Jeju Island, Korea, pages 256–265.

Frank Schilder and Ravikumar Kondadadi. 2008. FastSum: Fast and accurate query-based multi-document summarization. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics and HLT: Short Papers*. Columbus, Ohio, USA, pages 205–208.

Natalie Schluter. 2017. The limits of automatic summarisation according to rouge. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Short Papers*. Valencia, Spain, pages 41–45.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Vancouver, Canada, pages 1073–1083.

Dou Shen, Jian-Tao Sun, Hua Li, Qiang Yang, and Zheng Chen. 2007. Document summarization using conditional random fields. In *Proceedings of the 20th International Joint Conference on Artifical intelligence*. Hyderabad, India, pages 2862–2867.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Berlin, Germany, pages 1683–1692.

Karen Spärck Jones. 2007. Automatic summarising: The state of the art. *Information Processing and Management* 43(6):1449–1481.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*. pages 3104–3112.

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

Krysta Marie Svore, Lucy Vanderwende, and Christopher J. C. Burges. 2007. Enhancing single-document summarization by combining ranknet and third-party sources. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Prague, Czech Republic, pages 448–457.

Jiwei Tan and Xiaojun Wan. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Vancouver, Canada, pages 1171–1181.

Simone Teufel and Marc Moens. 1997. Sentence extraction as a classification task. In *Proceedings of ACL Workshop on Intelligent and Scalable Text Summarization*. Madrid, Spain, pages 58–65.

Xiaojun Wan. 2010. Towards a unified approach to simultaneous single-document and multi-document summarizations. In *Proceedings of the 23rd International Conference on Computational Linguistics*. Beijing, China, pages 1137–1145.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8(3-4):229–256.

Kristian Woodsend and Mirella Lapata. 2010. Automatic generation of story highlights. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Uppsala, Sweden, pages 565–574.

Kristian Woodsend and Mirella Lapata. 2012. Multiple aspect summarization using integer linear programming. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Jeju Island, Korea, pages 233–243.

Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning*. Lille, France, pages 2048–2057.

Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir Radev. 2017. Graph-based neural multi-document summarization. In *Proceedings of the 21st Conference on Computational Natural Language Learning*. Vancouver, Canada, pages 452–462.

Wen-tau Yih, Joshua Goodman, Lucy Vanderwende, and Hisami Suzuki. 2007. Multi-document summarization by maximizing informative content-words. In *Proceedings of the 20th International Joint Conference on Artifical intelligence*. Hyderabad, India, pages 1776–1782.

Wenpeng Yin and Yulong Pei. 2015. Optimizing sentence modeling and selection for document summarization. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*. Buenos Aires, Argentina, pages 1383–1389.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28*. pages 649–657.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark, pages 595–605.

Relational Summarization for Corpus Analysis

Abram Handler and **Brendan O'Connor**
College of Information and Computer Sciences
University of Massachusetts Amherst
ahandler@cs.umass.edu, brenocon@cs.umass.edu

Abstract

This work introduces a new problem, relational summarization, in which the goal is to generate a natural language summary of the relationship between two lexical items in a corpus, without reference to a knowledge base. Motivated by the needs of novel user interfaces, we define the task and give examples of its application. We also present a new query-focused method for finding natural language sentences which express relationships. Our method allows for summarization of more than two times more query pairs than baseline relation extractors, while returning measurably more readable output. Finally, to help guide future work, we analyze the challenges of relational summarization using both a news and a social media corpus.

1 Introduction

Research on automatic summarization (Nenkova et al., 2011; Das and Martins, 2007) aims to help users understand large document sets. However, the details of how textual summaries might actually be presented to users are often ignored. We propose that user interfaces which display noteworthy terms or concepts present the need for **relational summaries**: descriptions of the relationship between two entities or noun phrases from a corpus.

Examples of such interfaces include: commandline software for examining noteworthy terms or phrases (Squirrell, 2017; Robinson, 2016; Monroe et al., 2008), point-and-click browsers which display named entities and their interconnections on a network diagram (Wright et al., 2009; Görg et al., 2014; Tannier, 2016), concept map browsers (Falke and Gurevych, 2017b) and document search engines which suggest terms relevant to a query, such as the related searches displayed on Wikipedia info boxes from Google. In all such settings a natural question arises: what is the nature of the relationship between the entities or concepts shown in the interface? One particular interface which presents the need for a relational summary is shown in figure 1.

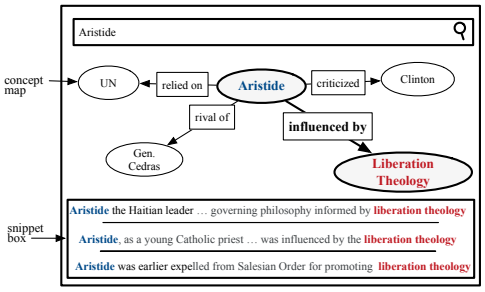


Figure 1: An example interface which requires relational summarization. The user has queried for the entity *Aristide*. The interface shows a *concept map* (top), displaying short summaries of Aristide's important relationships. The user has drilled down to see a more detailed summary of Aristide's relationship with liberation theology, displayed in a *snippet box* (bottom).

Relational questions are ubiquitous and varied. Examples include the following. What is the relationship between the "City of London" and "goal-delivery of Newgate" in 18th century court records (Hitchcock et al., 2012)? What is the relationship between "Advanced Integrated Systems" and "United Arab Emirates" in the Paradise Papers?[1] What does "dad" have to do with "mom" on the subreddit discussion forum *Relationship_Advice*?

This study seeks to answer such questions by examining the problem of **relational summarization**, which lies at the intersection of prior work on summarization and relation extraction. Unlike previous efforts at summarizing relationships (Falke and Gurevych, 2017a), our approach focuses on answering user queries about the connections between two particular terms, without ref-

[1] https://www.icij.org/investigations/paradise-papers/

Proceedings of NAACL-HLT 2018, pages 1760–1769
New Orleans, Louisiana, June 1 - 6, 2018.

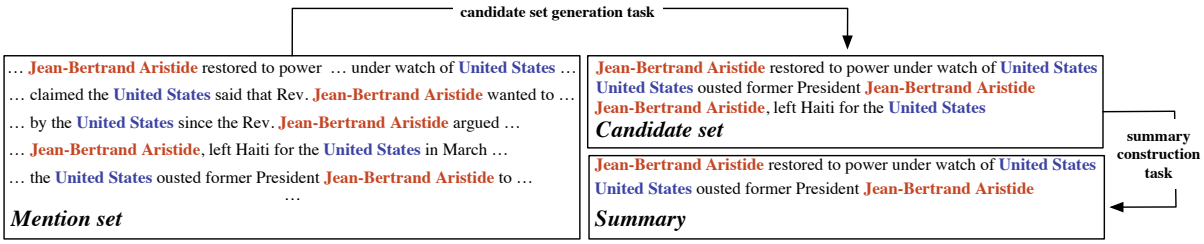


Figure 2: A relational summary is a synopsis of all sentences which mention two terms, denoted (t_1) and (t_2). We refer to such sentences as a *mention set*. In the figure above (t_1) is **Jean-Bertrand Aristide** and (t_2) is **United States**. To create a summary first requires identifying all statements in the mention set which coherently describe some relationship between (t_1) and (t_2). This **candidate set generation task** is a prerequisite for the subsequent **summary construction task**: selecting the top K candidates to create a summary. In this work, we offer a method for the first task and show how the second task will likely require a diversity of summarization techniques (§6).

erencing a knowledge graph (Voskarides et al., 2015).[2] In order to answer such queries we:

- Formally define the problem (§2), which we divide into two subtasks: **candidate set generation** and **summary construction**.
- Provide a new method for the candidate set generation task (§4), which we show outperforms baseline relation extraction techniques (§5) in terms of readability and yield.
- Analyze the summary construction task for future work (§6), demonstrating that different summarization techniques are likely most appropriate for different mention sets.

2 Formal definition and method

We refer to all sentences within a collection of documents which contain two terms, (t_1) and (t_2) as the *mention set*. (t_1) and (t_2) are noun phrases, a syntactic category which encompasses both traditional named entities like people and places, as well as less concrete, but important, entities and concepts like "liberation theology" (Handler et al., 2016).

A relational summary is a synopsis of the mention set. A summary consists of K relation statements, each displayed on its own line. Relation statements are natural language expressions which begin with (t_1) and end with (t_2). We refer to the span of tokens in between (t_1) and (t_2) as a *relation phrase*. We use the notation (t_1) r (t_2) to denote a relation statement, indicating two terms and a relation phrase. In the relation statement, "**Aristide** fled **Haiti**", r is the token "fled", (t_1) is the token **Aristide**, and (t_2) is the token **Haiti**.

Relation statements, which are strings intended for human readers, are similar to the 3-tuples, "*relations*", from prior work on information extraction (Banko et al., 2007). However, in this work, we show that the assumptions underlying the extraction of 3-tuples for machines (§3) leads to poor performance in summarizing mention sets for people (§5).

In this study, we present a strictly extractive method for generating relation statements: each relation statement must be constructed by deleting tokens from some sentence in the mention set.[3] Some relation statements constructed by deleting tokens from a sentence make sense; others do not. We refer to any (t_1) r (t_2) which makes makes sense to a human reader as **acceptable**.[4] Table 1 shows examples of acceptable and unacceptable relation statements, constructed by deletion.

s_1	Aristide fled Haiti ~~in 2004~~. (t_1) r (t_2)
s_2	~~For instance~~ Bush told Aristide ~~to leave~~. (t_1) r (t_2)

Table 1: Two relation statements constructed by deleting tokens from source sentences, s_1 and s_2. The relation statement extracted from s_1 is acceptable; the statement extracted from s_2 is not.

[2] Relational summaries are intended for general-purpose corpus analysis. Existing knowledge bases do not cover topics discussed in many corpora, such as historical court records (Hitchcock et al., 2012). Therefore, our approach does not employ a knowledge base.

[3] In subsequent studies of relation extractors (§5), we allow extractors to lightly introduce new tokens, such as adding the word "is" in relations expressed as noun phrases.

[4] Linguists sometimes use the term "acceptability" to refer to human judgements of the well-formedness of utterance. See Sprouse and Schütze (2014) for an overview.

Only acceptable relation statements are permitted in a summary. The set of all possible acceptable relation statements is called the **candidate set**, denoted $\mathcal{C}$. We refer to the task of identifying all acceptable relation statements as the **candidate set generation task**. Identifying a candidate set presents a subsequent problem of choosing the best collection of K relation statements from $\mathcal{C}$ to create a summary. We refer to this second step as the **summary construction task**.

As in traditional summarization (Das and Martins, 2007; Nenkova et al., 2011), a good relational summary should (i) be readable, (ii) include the most important aspects of the relationship between $(\boldsymbol{t_1})$ and $(\boldsymbol{t_2})$, (iii) avoid redundancy, and (iv) cover the full diversity of topics in the mention set.

Relational summaries might be presented with different kinds of user interfaces. In cases where a user seeks to browse many relationships, a summary might be displayed as a *concept map* (Falke and Gurevych, 2017a), where the two terms are vertexes in a directed graph and their relationship is printed along the edge label between them. In cases where user wants to investigate a specific relationship, a relational summary might be displayed as a *snippet box*: a short list of sentences which begin and end with the two terms. Figure 1 shows a snippet box and concept map. In a snippet box, both the number of lines in the summary and the length of the lines in the summary is longer than in a concept map.

3 Related work

Relational summarization intersects with a diversity of prior work from natural language processing, including work on **relation extraction**, **summarization** and **sentence compression**.

Traditionally, the goal of **relation extraction** is to cull structured facts for knowledge databases from unstructured text. Often, such facts take the form of a 3-tuple which defines a relationship between two arguments, such as (arg1=Angela Merkel, rel=met with, arg2=Theresa May). If extractors do not make use of a predefined schema, the task of finding relations is called Open Information Extraction (OpenIE). OpenIE systems[5] offer an off-the-shelf method for generating a candidate set for a relational summary. Their output can easily be linearized to $(\boldsymbol{t_1})\ r\ (\boldsymbol{t_2})$ statements by simply concatenating the three arguments of the triple to form a string.

However, we find that the recall of relation extractors is often too low to summarize many mention sets. We measure this disadvantage extensively in section §5.1. One reason for their poor performance might be that extractors have goals and assumptions which are poorly suited to the relation summarization task. In relation extraction, the aim is to find relation strings that recur for many different entity pairs, which allows such systems to build knowledge databases. For instance, relation extraction might be used to build tables of world leaders who rel="met with" other world leaders in order to analyze international politics. From this perspective, long, sparse, heterogenous and detailed relation strings which might apply only to a pair of specific arguments are undesirable, as they make it difficult to find general patterns across many different entity pairs. For example, the influential ReVerb OpenIE system (Fader et al., 2011) excludes "overly-specific relation phrases" which apply only to two entities. One way to help ensure that relations generalize across entity pairs is to strive for arguments which are as short as possible, a common goal in OpenIE (Stanovsky and Dagan, 2016).[6]

Our method for generating a candidate set is closer to approaches from **sentence compression** (Knight and Marcu, 2002; Clarke and Lapata, 2008; Filippova and Altun, 2013; Filippova et al., 2015), an NLP task which seeks to make a source sentence shorter while preserving the most important information and producing readable output. We show that our sentence compression approach allows us to achieve higher readability than off-the-shelf relation extractors (§5).

Sentence compression is often used in traditional extractive **summarization** to make more efficient use of a budgeted summary length. We discuss summarization further in §6, where we consider how existing work might be applied to the problem of selecting K statements from the candidate set.

[5]There are many available OpenIE systems. See Stanovsky and Dagan (2016) for an inventory of major work.

[6]Methods from the relation extraction literature which seek to deduce facts from extracted relations, such as Riedel et al. (2013), might also help identify useful summaries in future work. Relations which imply that other relations are true might make good summaries.

Sampled unacceptable compression	**Auburn police** are investigating the death of a **Tuskegee** ~~woman who died ...~~
Known acceptable compression	~~Drug firm~~ **Glenmark** has opened its new facility in **Argentina** ~~which would ...~~

Table 2: Examples of known acceptable and presumed unacceptable training examples, with entities shown in bold. We refer to crossed out spans as *outside of the compression*. Our model uses grammatical information from inside and outside of the compression to predict the acceptability of a compression.

4 Query-focused candidate set generation

Traditionally, relation extraction begins with a fixed notion of what constitutes a desirable "relation" between two arguments, defined by a predefined schema, a syntactic template (Fader et al., 2011), or a collection of seed examples (Angeli et al., 2015). The relation extraction task is then to correctly identify spans in which arguments are joined by a relation.

The relational summarization problem is somewhat different: we begin with a pair of query terms, $(\boldsymbol{t_1})$ and $(\boldsymbol{t_2})$, and we wish to learn the nature of their relationship. Therefore, any statement which coherently describes any relationship between the two query terms is potentially of interest, even if it does not match prior expectations of what constitutes a relation.

We thus approach the candidate set generation task as a specialized form of sentence compression: we attempt to predict if a sentence from the text can be coherently compressed to the form $(\boldsymbol{t_1})\ r\ (\boldsymbol{t_2})$. Table 2 shows examples of sentences which can and cannot be shortened to this form.

We use gold standard sentence–compression pairs from the Filippova and Altun (2013) dataset to supervise this prediction. In sentence compression corpora, gold standard compressions must be acceptable sentences. Therefore, compressions from the dataset which happen to begin and end with a named entity,[7] once extracted from source sentences, can serve as positive examples of acceptable relation statements. On the other hand, randomly chosen spans of the form $(\boldsymbol{t_1})\ r\ (\boldsymbol{t_2})$, which happen to arise in source sentences, are very often not acceptable as standalone sentences. These randomly sampled spans can serve as examples of unacceptable relation statements. We then predict acceptability with supervision from known gold acceptable and sampled, presumed incoherent examples.[8]

Filtering the original dataset in this manner[9] yields 17,529 positive and 30,266 negative sentences. We then downsample negative training examples to create two balanced classes of equal size, and use 81% of data for training, 9% for validation and the remaining 10% for testing.

Let $p(c = 1 \mid s, (\boldsymbol{t_1})\ r\ (\boldsymbol{t_2}))$ indicate the probability that a span of form $(\boldsymbol{t_1})\ r\ (\boldsymbol{t_2})$ extracted from sentence s is coherent. We model $p(c = 1|s, (\boldsymbol{t_1})\ r\ (\boldsymbol{t_2}))$ using logistic regression, with features based on the position of part-of-speech tags and dependency edges in s. Specifically, each sentence in the filtered dataset contains a span of the form $(\boldsymbol{t_1})\ r\ (\boldsymbol{t_2})$. We refer to the tokens in this span as *in the compression* because a user would see these tokens in a relation statement compressed from s. Each sentence also contains spans of tokens which are *outside of the compression* because they are deleted from the original source sentence to create a relation statement. Table 2 displays examples.

Our feature vector records the counts of how many times each part-of-speech tag in the tagset occurs in the compression and also independently records the counts of how many times each part-of-speech tag occurs out of the compression. We refer to the count of each part-of-speech tag in the compression and the count of each part-of-speech tag out of the compression as $\boldsymbol{\Phi}$. We also count the occurrence of each possible dependency edge label in the compression, and the count of each possible dependency edge label out of the compression. If a label's dependent lies in the compres-

[7] https://github.com/google-research-datasets/sentence-compression

[8] We manually inspect 100 negative examples, selected at random, and find that roughly 80% are in fact incoherent.

[9] We also exclude randomly chosen spans which happen to encompass the entire source sentence and exclude randomly chosen spans where $(\boldsymbol{t_1})$ and $(\boldsymbol{t_2})$ are joined by only edges of type compound in the dependency graph of the compression (e.g. "Coup leader Cedras ..."). We use CoreNLP version 3.8 to extract *enhanced++* Universal Dependencies (Manning et al., 2014; Schuster and Manning, 2016; Nivre et al., 2016). We also filter positive and negative examples where the span between $(\boldsymbol{t_1})$ and $(\boldsymbol{t_2})$ is longer than J=75 characters, to simulate a space constraint in a user interface. Finally, we remove all punctuation from the end of the sentence for both positive and negative examples because all gold positive compressions end in punctuation marks. For positive examples, if the compressed version of a sentence deletes tokens between t_1 and t_2, we replace the span between t_1 and t_2 in the source sentence with the compression.

$p(c = 1\|s, (\boldsymbol{t_1})\ r\ (\boldsymbol{t_2}))$	$(\boldsymbol{t_1})\ r\ (\boldsymbol{t_2})$
.005	**Jean-Bertrand Aristide** that the **United States**
.010	**United States** since the Rev. **Jean-Bertrand Aristide**
...	...
.894	**United States** ousted former President **Jean-Bertrand Aristide**
.976	**Jean-Bertrand Aristide**, left Haiti for the **United States**

Table 3: Highest and lowest coherence predictions from the set **United States – Jean-Bertrand Aristide**

sion, we consider the label in the compression.[10] We refer to these dependency edge counts as $\boldsymbol{\Psi}$. Our final feature vector, $\boldsymbol{\Omega}$, is defined as the concatenation of $\boldsymbol{\Psi}$ and $\boldsymbol{\Phi}$.

Features	Test accuracy
$\boldsymbol{\Phi}$ (pos)	.858
$\boldsymbol{\Psi}$ (deps)	.892
$\boldsymbol{\Omega}$ (deps & pos)	.896

Table 4: Test accuracies.

We implement our model with *scikit-learn* (Pedregosa et al., 2011) and manually tune the inverse regularization constant to the setting, $c = 1$, which achieves the highest accuracy on the validation set. For evaluation, a sentence is presumed coherent if $p(c = 1|s, (\boldsymbol{t_1})\ r\ (\boldsymbol{t_2})) > .5$. Using the feature vector $\boldsymbol{\Omega}$ we achieve an accuracy of .896 on the test set. We also present results using only the $\boldsymbol{\Psi}$ and $\boldsymbol{\Phi}$ features (table 4) because reliable dependency parses are not available in some settings (Blodgett et al., 2016; Bamman, 2017).

Two limitations of this approach suggest areas for future work. First, in some cases, the relationship between $(\boldsymbol{t_1})$ and $(\boldsymbol{t_2})$ might not be expressed in the form, $(\boldsymbol{t_1})\ r\ (\boldsymbol{t_2})$, as in "**Russia** and **France** signed an agreement". In order to generate candidate relation statements it would be helpful to lightly rewrite the sentence, as in "**Russia** signed an agreement with **France**". Additionally, a sentence might express a relationship between two terms but be too long to display on a concept map or a snippet box. In these cases, it would be helpful to compress the sentence to create a more concise relation statement.

5 Experiments

Any relational summarization system should deliver a high-quality summary when a user queries for two terms. Therefore, ideally, a system should generate the largest possible candidate set, without returning incoherent relation statements. We thus evaluate our query-focused generation method in terms of both readability and yield (total relation statements recalled). Our method generates three times more relation statements than OpenIE systems, allowing for summarization of two times more query pairs. We also achieve higher scores in a test of human coherence judgements (table 5).

More concretely, we evaluate our compression-based method for generating candidate sets against two relation extractor baselines on two very different corpora: (1) all comments from the large "relationships"[11] subreddit from June, 2015 – September, 2017[12] and (2) a collection of *New York Times* articles from 1987 to 2007 which mention the country "Haiti" (Sandhaus, 2008). For each corpus, we first find a collection of multi-word phrases using the `phrasemachine` package (Handler et al., 2016) which extracts all multi-word, noun phrase terms from the corpus.

After extracting all terms, we determine the top 100 terms, by count. We then examine all non-empty mention sets for all possible combinations of two top terms. A mention set is a set of sentences which mention two terms (§2). We examine all mention sets because an investigator should be able to investigate any entity she chooses while analyzing a corpus.

In subsequent experiments, we require all relation statements be less than or equal to $J = 75$ characters, which excludes overly verbose relation statements which are unsuitable for many user interfaces.

5.1 Yield experiments

Off-the-shelf relation extractors generate 3-tuples from each mention set. Some of those 3-tuples might have one argument which is equal to $(\boldsymbol{t_1})$ and another argument which is equal to $(\boldsymbol{t_2})$. Each such 3-tuple can be linearized into a string of the form $(\boldsymbol{t_1})\ r\ (\boldsymbol{t_2})$ to generate a candidate set. However, we find that using extractors in this

[10] Enhanced dependencies allow for a token to have more than one incoming edge (i.e., multiple parents). If there is more than one incoming edge, we pick an edge at random.

[11] "relationships" refers to interpersonal relationships

[12] https://medium.com/@jason_82699/pushshift-reddit-api-md-c2d70745c270

manner achieves a low yield (total number of extracted relations). A low yield is undesirable both because it limits the number of mention sets which may be summarized and generates fewer relation statements from which to select an optimal relational summary.

More precisely, we identify the 3-tuples which an OpenIE system extracts from a mention set such that exactly one argument from the triple is equal[13] to (t_1) and exactly one argument from the triple is equal to (t_2). We refer to these 3-tuples as "matching". We then count (1) the total number of mention sets which contain at least one matching 3-tuple and (2) the total number matching 3-tuples across all mention sets. We refer to such counts as the *yield* of a candidate generation system.

We measure the yield of Stanford OpenIE (Angeli et al., 2015) and ClausIE (Del Corro and Gemulla, 2013) on the *New York Times* and *Reddit* corpora, and compare each system to our compression-based approach (§4).[14] We measure these two relation extractors because Stanford OpenIE is included with the popular CoreNLP software and ClausIE achieves the highest recall in two systematic studies of relation extractors (Stanovsky and Dagan, 2016; Zhang et al., 2017).

We find that, for the great majority of sentences, relation extractors do not extract any relations between (t_1) and (t_2). Moreover, for many mention sets, the number of relations extracted with off-the-shelf systems is often zero. We show these results in table 5.

This suggests that although relation summarization is superficially similar to relation extraction, off-the-shelf extractors are poor tools for creating textual units to summarize mention sets. Very often, two terms are related to each other in ways which are simply not captured by relation extractors.

[13]Note that OpenIE systems might not extract the literal string (t_1) or (t_2) as arguments. For instance, if (t_1) is "Merkel" the OpenIE system might extract the argument "Angela Merkel". If some term and some argument from a relational triple share the same head token in the dependency parse of the sentence we say that they are equal. Falke and Gurevych (2017c) employ a similar equality criterion. We tokenize with CoreNLP. In extremely rare cases, tokenization mismatches between CoreNLP and ClausIE make it impossible to apply this criterion.

[14]For our compression-based approach, we count all cases where $p(c = 1 \mid s, (t_1)\ r\ (t_2)) > .5$ as extracting a relation statement.

5.2 Human acceptability judgments

Our compression-based method achieves higher yield than off-the-shelf relation extractors. However, because all sentences in a mention set include (t_1) and (t_2), it is always possible to generate a very large candidate set by simply extracting all spans between (t_1) and (t_2) from the mention set, regardless if such relation statements are coherent. We examine if gains in yield come at the expense of acceptability by presenting randomly selected relation statements to workers on the platform Figure Eight[15] (formerly Crowdflower) and asking workers to rate the extent to which they agree or disagree as to whether a relation statement is a "coherent English sentence" on a scale from 1 to 5. Each relation statement is shown to three workers in total.[16] Our approach is broadly similar to the readability experiments reported in Filippova and Altun (2013).

We solicit 481 total judgements from workers and calculate the mean acceptability score, by method and corpus (table 5). Our method achieves the highest mean acceptability score for both corpora.

Additionally, aggregating judgments across corpora, we observe a statistically significant (p=8 x 10^{-4}) difference between our method ($\mu = 3.89$, $\sigma = 1.38$) and Stanford OpenIE ($\mu = 3.33$, $\sigma = 1.46$) in a two-tailed t-test. Our method also achieves a higher mean score than ClausIE ($\mu = 3.69$, $\sigma = 1.44$), although the difference is not significant.

6 Future work: summary construction task

After a relational summarization system generates a candidate set, the next task is selecting the top K candidate statements for inclusion in a summary (figure 2). In this work, we do not attempt this summary construction task. However, in this section, we analyze the nature of the relational summarization challenge by describing differences among mention sets, and how these differences might affect future efforts at summarization.

We observe that mention sets are inherently heterogenous. Some describe a single, temporally-

[15]https://www.figure-eight.com/

[16]We use seven test questions to filter out careless or bad faith responses. Workers must answer 70% of test questions correctly to be included in a task's results. We construct test questions blindly, without knowledge of the system which generated the relation statement.

	Yield				Coherence	
	Total non-empty pairs		Total rel. stmts.		Mean judgment	
	Haiti	Reddit	Haiti	Reddit	Haiti	Reddit
ClauseIE	128	1,121	279	3,949	3.67	3.71
StanfordOIE	443	1,488	972	5,605	3.69	2.97
This work	**739**	**3,766**	**2,954**	**21,495**	**3.94**	**3.85**
Upper bound	2,472	4,496	43,051	123,760	Range: 1-5	

Table 5: We compare Stanford OpenIE, ClausIE and our headline-based compression method for the candidate set generation task on two different corpora (Haiti articles from *New York Times*, and the *Reddit* relationships forum) in terms of (1) how many entity pairs have a non-empty candidate set, (2) how many total relation statements are generated, and (3) the average human judgment of acceptability (§5.2). For yield measures, the upper bound on the left shows the total number of non-empty entity pairs (i.e. how many pairs actually cooccur in at least one sentence, out of all $\binom{100}{2} = 4950$ theoretically possible pairs) and the upper bound on the right shows the total number of sentences in the corpus which mention at least two of the terms. Our method summarizes more entity pairs across both corpora, and achieves the highest acceptability scores among all techniques (§5.2).

focused event. Others describe a consistent, unchanging relationship. Still others describe intricate sagas unfolding across time. For instance, within the Haiti corpus, one mention set describes events in 1994 when **General Cedras** fled to the **Dominican Republic**. This mention set is quite different from a set of sentences in the Reddit corpus in which users assert that **video games** are a **deal breaker** in interpersonal relationships. Figure 3 displays hand-crafted summarizes for these mention sets.

In general, the properties which guide how a mention set should be summarized are its **size**, **topical diversity**, **temporal focus** and the degree to which the set expresses **states or events**. In this section, we use the notation (t_1) – (t_2) to refer to a mention set. For instance, *New York – London* would refer to all sentences from a corpus which contain the names of both of these cities.

Size. In general, because many word types in a corpus occur infrequently (Zipf, 1949), the number of sentences which mention (t_1) and (t_2) is often small. For instance, of the 320,670 total sentences in the Haiti corpus, only 160 mention "Jean-Bertrand Aristide" and the "United States," which is nonetheless among the larger mention sets in the corpus. In general, larger sets often describe complex and noteworthy relationships, which are more difficult to summarize (figure 3c). Note that although individual mention sets are often small enough to simply read (unlike in some multi-document summarization settings), summarization of mention sets is still quite useful, as practitioners will often seek to understand many different relationships as they investigate a new topic (e.g. figure 1).

Topical diversity. In general, some mention sets are focused on a single topic, others are more diffuse. For instance, after losing power in a second, 2004 coup Haiti's Jean Bertrand Aristide was forced into exile in South Africa. The mention set for *Jean Bertrand Aristide – South Africa* contains twelve sentences which (mostly, but not exclusively) describe Aristide's removal from power and life in exile in South Africa from 2004 onwards. Detecting and including diverse or complex topics is a classic aim of traditional multi document summarization (e.g. Lin and Hovy (2000)), which might be applied in this new setting.

Temporal focus. In timestamped corpora such as news archives or social media posts, some mention sets are focused within certain time periods; others are spread across the span of the corpus. For instance, in the Haiti corpus, *General Cedras – Dominican Republic* are only mentioned together during a few months of 1994 (figure 3b). A good summary for this mention set should describe a central event from this time period: when General Cedras fled to the Dominican Republic. On the other hand, *Jean-Bertrand Aristide – United States* are mentioned together in 67 months in the corpus, covering a number of important events spread across decades (figure 3c). For this mention set, a narrow summary focusing on a single event would be inappropriate.

Many existing methods specialize in detecting (Chaney et al., 2016), tracking (Allan et al., 1998) and summarizing evolving topics in timestamped documents. Some systems focus specifically on summarizing event "spikes": both in news (e.g.

video games and I don't want that to be a **deal breaker**
video games was a **deal breaker**
video games is a **deal breaker**

(a) A hand-crafted summary for the mention set **video games–deal breaker**. The mention set contains many stative descriptions of the relationships between the two terms, indicating that a summary might focus on presenting fixed relationships rather than evolving events.

General Cedras ... last week fled to the **Dominican Republic**
Dominican Republic ... has indicated it will not permit permanent residence by **General Cedras**

(b) A hand-crafted summary for the mention set **General Cedras–Dominican Republic**. The set has a high number of mentions which all fall within a several month span, hinting at a relationship focused on a particular event at a particular point in time.

Aug. 1994 **United States** supports the restoration of the democratically elected president of Haiti, **Jean-Bertrand Aristide**
Oct. 1995 **Jean-Bertrand Aristide** was restored to power a year ago under the watch of **United States**
Sep. 2002 **United States** and other donors withheld contributions, hoping to spur President **Jean-Bertrand Aristide**
Mar. 2004 **Jean-Bertrand Aristide** asserted that he had been driven from power by the **United States**

(c) A hand-crafted summary for the mention set **Jean-Bertrand Aristide–United States**, one of the largest in the Haiti corpus. The mention set describes a complex, shifting relationship; at different times over several decades, Aristide was a beneficiary, opponent and critic of the United States.

Figure 3: Mention sets are heterogenous, requiring a diversity of summarization techniques. In this work, we analyze the diversity of mention sets towards future attempts that the relational summarization problem.

Alfonseca et al. (2013)) and on social media (e.g. Nichols et al. (2012)). In some cases, the event described in a mention set will even match the loose form of a common narrative template (Chambers and Jurafsky, 2008), such as when the two terms are codefendants at a trial.

Mention sets which are more temporally diffuse are also more challenging. Update summarization refers to summarizing changes introduced by new documents, possibly from a high volume stream (Kedzie et al., 2015). This form of summarization is important in cases when a relationship shifts or changes through time, as in figure 3c.

States or events. Mention sets may be coarsely divided into cases where the set expresses a stable state or property of the world in the eyes of the author (e.g. "England is a close ally of the US" or "video games are a deal breaker") and cases where the relation statement expresses a change or event (e.g. "Gen. Cedras fled to the Dominican Republic" or "dad left mom"). In many interesting cases, the mention set contains a mix of stative and eventive relation statements which express a narrative, such as "**America** is an ally of **South Korea**" and "**America** sent a destroyer to **South Korea**".

Defining (Pustejovsky, 1991), extracting (Aguilar et al., 2014) and determining relationships between events (Hovy et al., 2013) is a challenging research area. But a better understanding of states and events would improve future work. For instance, if a summary includes the event "Jolie divorced Pitt", it does not need to include the stative relation phrase "Jolie was married to Pitt". To our knowledge, there is no prior work which considers how fine-grained relations between states and events might be employed for summarization. MacCartney and Manning (2009) offer a framework which might serve as a useful starting catalog.

Conclusion

This work defines a problem which lies at the intersection of typically unrelated fields in natural language processing, summarization and relation extraction. We present a new method which finds large numbers of natural language expressions which coherently describe relationships. We also analyze the challenges of the relational summarization task, by investigating and describing the inherent heterogeneity of mention sets. Because of this heterogeneity, we argue that future attempts to summarize relationships will likely require a diversity of models and techniques.

Acknowledgments

Thanks to Emma Strubell, Patrick Verga, Haw-Shiuan Chang, Su Lin Blodgett, Katherine Keith and the UMass NLP reading group for helpful discussions and comments. Thanks to Brian Dillon for helping us better understand how to collect and interpret human judgements of linguistic acceptability.

References

Jacqueline Aguilar, Charley Beller, Paul McNamee, Benjamin Van Durme, Stephanie Strassel, Zhiyi Song, and Joe Ellis. 2014. A comparison of the events and relations across ACE, ERE, TAC-KBP, and FrameNet annotation standards. In *Proceedings of the Second Workshop on EVENTS: Definition, Detection, Coreference, and Representation*. Association for Computational Linguistics.

Enrique Alfonseca, Daniele Pighin, and Guillermo Garrido. 2013. Heady: News headline abstraction through event pattern clustering. In *ACL*.

James Allan, Jaime Carbonell, George Doddington, Jonathan Yamron, and Yiming Yang. 1998. Topic detection and tracking pilot study: Final report.

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *ACL*.

David Bamman. 2017. Natural language processing for the long tail. *Digital Humanities* .

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *IJCAI*.

Su Lin Blodgett, Lisa Green, and Brendan T. O'Connor. 2016. Demographic dialectal variation in social media: A case study of african-american english. In *EMNLP*.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of ACL-08: HLT*.

Allison Chaney, Hanna Wallach, Matthew Connelly, and David Blei. 2016. Detecting and characterizing events. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research* 31:399–429.

Dipanjan Das and André F. T. Martins. 2007. A survey on automatic text summarization. Technical report, Literature Survey for the Language and Statistics II course at Carnegie Mellon University.

Luciano Del Corro and Rainer Gemulla. 2013. Clausie: clause-based open information extraction. In *Proceedings of the 22nd international conference on World Wide Web*. ACM.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *EMNLP*. Edinburgh, Scotland, UK.

Tobias Falke and Iryna Gurevych. 2017a. Bringing structure into summaries: Crowdsourcing a benchmark corpus of concept maps. In *EMNLP*.

Tobias Falke and Iryna Gurevych. 2017b. Graphdocexplore: A framework for the experimental comparison of graph-based document exploration techniques. In *EMNLP: System Demonstrations*.

Tobias Falke and Iryna Gurevych. 2017c. Utilizing automatic predicate-argument analysis for concept map mining. In *Proceedings of the 12th International Conference on Computational Semantics (IWCS)*.

Katja Filippova, Enrique Alfonseca, Carlos A Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with LSTMs. In *EMNLP*.

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *EMNLP*.

Carsten Görg, Zhicheng Liu, and John Stasko. 2014. Reflections on the evolution of the jigsaw visual analytics system. *Information Visualization* 13(4):336–345.

Abram Handler, Matthew J Denny, Hanna Wallach, and Brendan OConnor. 2016. Bag of what? Simple noun phrase extraction for text analysis. *Workshop on NLP + CSS, EMNLP* .

Tim Hitchcock, Robert Shoemaker, Clive Emsley, Sharon Howard, and Jamie McLaughlin. 2012. The old bailey proceedings online, 1674-1913. www.oldbaileyonline.org.

Eduard Hovy, Teruko Mitamura, Felisa Verdejo, Jun Araki, and Andrew Philpot. 2013. Events are not simple: Identity, non-identity, and quasi-identity. In *NAACL*.

Chris Kedzie, Kathleen McKeown, and Fernando Diaz. 2015. Predicting salient updates for disaster summarization. In *ACL*.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence* 139(1):91–107.

Chin-Yew Lin and Eduard Hovy. 2000. The automated acquisition of topic signatures for text summarization. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*. Association for Computational Linguistics.

Bill MacCartney and Christopher D Manning. 2009. An extended model of natural logic. In *Proceedings of the eighth international conference on computational semantics*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, Baltimore, Maryland, pages 55–60. http://www.aclweb.org/anthology/P14-5010.

Burt L Monroe, Michael P Colaresi, and Kevin M Quinn. 2008. Fightin' words: Lexical feature selection and evaluation for identifying the content of political conflict. *Political Analysis* 16(4):372–403.

Ani Nenkova, Kathleen McKeown, et al. 2011. Automatic summarization. *Foundations and Trends in Information Retrieval* 5(2–3):103–233.

Jeffrey Nichols, Jalal Mahmud, and Clemens Drews. 2012. Summarizing sporting events using twitter. In *IUI*.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association (ELRA), Paris, France.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.* 12:2825–2830.

James Pustejovsky. 1991. The syntax of event structure. *Cognition* 41(1):47–81.

Sebastian Riedel, Limin Yao, Benjamin M. Marlin, and Andrew McCallum. 2013. Relation extraction with matrix factorization and universal schemas. In *NAACL*.

David Robinson. 2016. Text analysis of trump's tweets confirms he writes only the (angrier) android half. http://varianceexplained.org/r/trump-tweets/.

Evan Sandhaus. 2008. The New York Times Annotated Corpus. *Linguistic Data Consortium* LDC2008T19.

Sebastian Schuster and Christopher D. Manning. 2016. Enhanced english universal dependencies: An improved representation for natural language understanding tasks. In *LREC*.

John Sprouse and Carson Schütze. 2014. *Research Methods in Linguistics*, Cambridge University Press, Cambridge, UK, chapter Judgment Data.

Tim Squirrell. 2017. Linguistic data analysis of 3 billion Reddit comments shows the alt-right is getting stronger. https://qz.com/1056319/what-is-the-alt-right.

Gabriel Stanovsky and Ido Dagan. 2016. Creating a large benchmark for open information extraction. In *EMNLP*. Austin, Texas.

Xavier Tannier. 2016. NLP-driven data journalism: Time-aware mining and visualization of international alliances. In *Proceedings of the 2016 IJCAI Workshop on Natural Language Processing meets Journalism*.

Nikos Voskarides, Edgar Meij, Manos Tsagkias, Maarten De Rijke, and Wouter Weerkamp. 2015. Learning to explain entity relationships in knowledge graphs. In *ACL*.

Brandon Wright, Jason Payne, Matthew Steckman, and Scott Stevson. 2009. Palantir: A visualization platform for real-world analysis. In *Visual Analytics Science and Technology, 2009. VAST 2009. IEEE Symposium on*. IEEE, pages 249–250.

Sheng Zhang, Rachel Rudinger, and Benjamin Van Durme. 2017. An Evaluation of PredPatt and Open IE via Stage 1 Semantic Role Labeling. In *The Proceedings of the 12th International Conference on Computational Semantics (IWCS)*.

George K. Zipf. 1949. *Human Behavior and the Principle of Least Effort*. Addison-Wesley.

What's this Movie about?
A Joint Neural Network Architecture for Movie Content Analysis

Philip John Gorinski and **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh, EH8 9AB
P.J.Gorinski@sms.ed.ac.uk, mlap@inf.ed.ac.uk

Abstract

This work takes a first step toward movie content analysis by tackling the novel task of movie overview generation. Overviews are natural language texts that give a first impression of a movie, describing aspects such as its genre, plot, mood, or artistic style. We create a dataset that consists of movie scripts, attribute-value pairs for the movies' aspects, as well as overviews, which we extract from an online database. We present a novel end-to-end model for overview generation, consisting of a multi-label encoder for identifying screenplay attributes, and an LSTM decoder to generate natural language sentences conditioned on the identified attributes. Automatic and human evaluation show that the encoder is able to reliably assign good labels for the movie's attributes, and the overviews provide descriptions of the movie's content which are informative and faithful.

1 Introduction

Movie summarization is the task of automatically summarizing a screenplay in order to gain a general impression of its content. This may include describing the movie's main characters and plot, its genre, artistic style, and so on. As more and more movies are being produced every year[1], there is an ever growing need to facilitate this task. Potential applications include producing shorter versions of scripts to help with the decision making process in a production company, enhancing movie search by generating descriptions of what the movie is about, and notably, supporting movie recommendation engines by abstracting over specific keywords to more general concepts.

Figure 1 gives an example of the type of movie content analysis we would like to obtain automatically. The information is taken from Jinni, a large database (and movie recommendation engine) which indexes movies based on attributes and their values[2] (see the top half of Figure 1) and further aggregates these into a comprehensive overview (see the second half of Figure 1). Jinni's movie attributes were created by film professionals based on analysis of user reviews and metadata. There are hundreds, and they aim to describe aspects such as mood, style, plot, and setting for any released movie or TV show. Although some of these attributes could not be possibly ascribed without information from external sources (e.g., *Praise*, or *Based on*), others could be inferred by watching the movie or reading the

Mood:	Suspenseful, Captivating, Tense, Scary
Plot:	Serial Killer, Special Agents, Investigation, Mind Game, Psychopath, Crimes, Deadly, Law Enforcement, Mind and Soul, Rivalry
Genre:	Crime, Thriller
Style:	Strong Female Presence
Attitude:	Serious, Realistic
Place:	Maryland, USA, Virginia
Period:	20th Century, 90s
Based on:	Based on Book
Praise:	Award Winner, Blockbuster, Critically Acclaimed, Oscar Winner, Modern Classic, Prestigious Awards
Flag:	Brief Nudity, Sexual Content, Strong Violent Content

The Silence of the Lambs can be described as tense, captivating, and suspenseful. The plot revolves around special agents, mind games, and a psychopath. The main genres are thriller and crime. In terms of style, The Silence of the Lambs stars a strong female character. In approach, it is serious and realistic. It is located in Maryland and Virginia. The Silence of the Lambs takes place in the 1990s. It is based on a book. The movie has received attention for being a modern classic, an Oscar winner, and a blockbuster. Note that The Silence of the Lambs involves brief nudity and sexual content.

Figure 1: Jinni attributes, their values, and overview for "The Silence of the Lambs". Underlined attribute values appear in the overview.

[1] According to http://www.boxofficemojo.com/ during 2009–2016, movie releases went up from 536 to 729.

[2] Throughout this paper *attributes* are in italic font and their values in sans serif.

Proceedings of NAACL-HLT 2018, pages 1770–1781
New Orleans, Louisiana, June 1 - 6, 2018.

screenplay (e.g., *Genre, Plot, Flag, Mood, Place*).

This work takes a step toward automatic script summarization by jointly modeling the tasks of movie attribute identification and overview generation. Specifically, we propose a novel neural network architecture which draws insights from encoder-decoder models recently proposed for machine translation (Bahdanau et al., 2015) and related sentence generation tasks (Wen et al., 2015; Mei et al., 2016; Lebret et al., 2016). Our model takes the screenplay as input and generates an overview for it. Rather than representing the script as a sequence, we employ feed-forward neural networks (Zhang and Zhou, 2006; Kurata et al., 2016) to encode the screenplay into various attributes (e.g., *Plot*, *Genre*) and their labels (e.g., thriller, romance), viewing movie content analysis as a multi-label classification problem. Our decoder generates movie overviews using a Long Short-Term Memory network (LSTM; Hochreiter and Schmidhuber, 1997), a type of recurrent neural network with a more complex computational unit which is semantically conditioned (Wen et al., 2015, 2016) on this attribute specific representation. Our model is trained end-to-end using screenplays and movie overviews as the supervision signal.

In both automatic and human-based evaluations our neural network architecture outperforms competitive baselines and generates movie overviews which are well-received by human judges. To the best of our knowledge, this is the first work to automatically analyze and summarize the content of screenplays.

2 Related Work

Recent years have seen increased interest in the computational analysis of movie screenplays. Ye and Baldwin (2008) create animated storyboards using the action descriptions of movie scripts. Danescu-Niculescu-Mizil and Lee (2011) use screenplays to study the coordination of linguistic styles in dialog. Bamman et al. (2013) induce personas of film characters from movie plot summaries. Agarwal et al. (2014a; 2014b; 2015) extract social networks from scripts, create *xkcd* movie narrative charts, and automate the Bechdel test which is designed to assess the presence of women in movies. Gorinski and Lapata (2015) summarize screenplays by selecting important scenes. Our work joins this line of research in an attempt to automatically induce information pertaining to a movie's content such as its genre and plot elements.

There has been a surge of interest recently in repurposing sequence transduction neural network architectures for various generation tasks such as machine translation (Sutskever et al., 2014), sentence compression (Chopra et al., 2016), and simplification (Zhang and Lapata, 2017). Central to these approaches is an encoder-decoder architecture modeled by recurrent neural networks. The encoder reads the source sequence into a list of continuous-space representations from which the decoder generates the target sequence. Previously proposed architectures are not directly applicable to our task for at least two reasons: (a) the correspondence between screenplays and overviews is very loose, and (b) the screenplay is not strictly speaking a sequence (a screenplay is more like a book consisting of thousands of sentences), and cannot be easily compressed into a vector-based representation from which to generate the overview.

Rather than attempting to decode the overview directly from the screenplay, we encode the latter into attribute-value pairs which we then decode into overviews. We conceptualize the generation task as a joint problem of multi-label categorization, where each screenplay is assigned to one or more categories, and content-sensitive natural language generation. Many machine learning techniques have been proposed for building automatic text categorization systems (see Sebastiani, 2002 and Dalal and Zaveri, 2011 for overviews), including neural networks (Belanger and McCallum, 2016; Kurata et al., 2016). Our encoder is a feed-forward neural network, which, however, is able to capture label interactions which are important for our content analysis task. Our decoder employs an enhanced LSTM architecture which directly maximizes the probability of the overview given the screenplay's attribute values. Conditional LSTMs have been applied to various related tasks, including image description generation (Vinyals et al., 2015), the verbalization of database records (Mei et al., 2016; Lebret et al., 2016), and the generation of dialogue acts (Wen et al., 2015, 2016).

3 The Jinni Movie Dataset

Our dataset was built on top of ScriptBase, a collection of 1,276 movie scripts, which Gorinski and

Attribute	Jinni	Frequent	Merged
Mood	29	19	19
Plot	406	101	101
Genre	31	31	31
Attitude	8	8	8
Place	173	53	24
Flag	9	9	6

Table 1: Movie attributes and their values.

Lapata (2015) obtained by automatically crawling web-sites such as imsdb.com. We crawled Jinni[3] in order to obtain attributes and overviews (see Figure 1) for each movie in ScriptBase. As mentioned earlier, attributes have values which are essentially labels/tags describing the movie's content, whereas overviews are short summaries giving a first impression of the movie. The crawl resulted in 917 movies which Jinni and ScriptBase had in common. We further split these into training, development and test sets, with 617, 200, and 100 instances, respectively. We concentrate on the six types of attributes shown in Table 1 whose values we hypothesize can be inferred from analyzing the movie's screenplay.

Table 1 provides an overview of the number of labels used in our experiments. Jinni contains a wealth of attribute values varying from nine for *Flag* to more than 400 for *Plot*. Additionally, value names for some attributes are synonyms or near-synonyms (e.g., Nudity and Brief Nudity for *Flag*). We reduced the set of attribute values to those that occurred most frequently (column Frequent in the table) and merged synonyms into a common label (column Merged).

4 Neural Network Architecture

We could approach the movie overview generation task using an attention-based encoder-decoder model (Bahdanau et al., 2015). The encoder would transform the screenplay into a sequence of hidden states with an LSTM (Hochreiter and Schmidhuber, 1997) or another type of computational unit (Cho et al., 2014). The decoder would use another recurrent neural network to generate the overview one word at time, conditioning on all previously generated words and the representation of the input, while an attention mechanism would revisit the input sequence dynamically highlighting pieces of information relevant for the generation task. As mentioned earlier, viewing screenplays as a sequence of sentences is problematic both computationally and conceptually. Even if we used a hierarchical encoder (Tang et al., 2015; Yang et al., 2016) by first building representations of sentences and then aggregating those into a representation of a screenplay, it is doubtful whether a fixed length vector could encode the content of the movie in its entirety or whether the attention mechanism would effectively isolate the parts of the input relevant for generation.

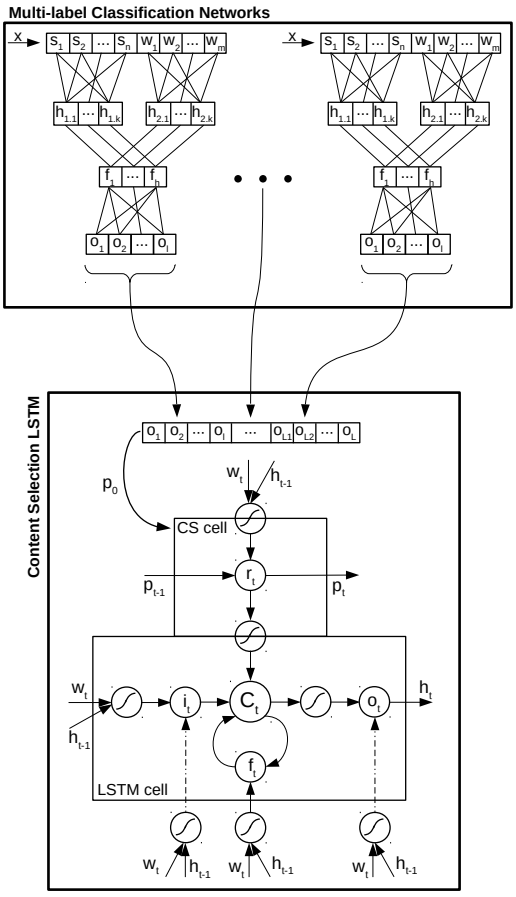


Figure 2: Neural network architecture: given feature vector x representing a screenplay, we employ feed-forward multi-label classification networks to encode the movie into a content vector p_θ representing attribute labels; this encoding is fed into an LSTM with a content selection cell.

We therefore propose an architecture that consists of two stacked neural network models for the tasks of movie attribute identification and overview generation. Figure 2 illustrates our model. We use simple feed-forward neural networks to impose some structure on the input by identifying the labels that most likely apply to the screenplay. We subsequently employ a semantically conditioned LSTM (Wen et al., 2015, 2016)

[3]Dated on 2015/04/18 and available from http://www.jinni.com/.

to select the content for which to generate sentences. This architecture is advantageous for a number of reasons. Firstly, by imposing structure on the screenplays, the generation network is faced with a more compact and informative representation. This allows us to make use of a content selection LSTM similar to Wen et al., (2015; 2016), generating fluent and label-specific outputs. Secondly, it enables us to train the screenplay encoder (aka classification network) and the decoder jointly, in an end-to-end fashion.

4.1 Multi-label Encoder

As shown in Figure 1, the overview highlights various aspects of the movie, essentially devoting a sentence to each attribute. This observation motivates us to encode the screenplay as a set of attributes (with their values) and then decode these into a sentence one by one. We treat attribute encoding as a multi-label classification problem: an attribute (e.g., *Genre* or *Plot*), will typically have multiple values (aka labels) which are suitable for the movie and should occur in the generated sentence. Furthermore, these labels naturally influence each other. For example, a movie whose *Genre* is Crime is also likely to be a Thriller while it is less likely to be a Parody. In traditional multi-label classification such interactions are either ignored (Read et al., 2011; Tsoumakas and Katakis, 2006; Godbole and Sarawagi, 2004; Zhang and Zhou, 2005), or represented by label combinations (Tsoumakas and Vlahavas, 2007; Read et al., 2008). A few approaches assume or impose an existing structure on the label space (Schwing and Urtasun, 2015; Chen et al., 2015; Huang et al., 2015; Jaderberg et al., 2014; Stoyanov et al., 2011; Hershey et al., 2014; Zheng et al., 2015).

We employ a neural network approach with the aim of abstracting the screenplay into a set of meaningful labels whose correlations are discovered automatically, during training. As shown on top of Figure 2, our encoder is a feed-forward neural network where individual neurons represent the labels to be classified. The input to the network is a feature vector x representing the screenplay (we discuss the specific features we use in more detail shortly):

$$h_n = \sigma(W_n x_n) \tag{1}$$

The input is split into k segments by feature type, and the feature segments are fed into k separate fully connected hidden layers. The hidden layer outputs are then combined using simple element-wise addition:

$$f = h_1 \oplus h_2 \oplus \cdots \oplus h_k \tag{2}$$

The combined feature layer is used to compute an l-sized output layer, where l corresponds to the size of the classification label set. The final activation of the output units is obtained by applying the sigmoid function to the output layer:

$$O = \sigma(W_o f) \tag{3}$$

In order to better capture label interactions, we adapt a method of network initialization recently introduced in Kurata et al. (2016). In this approach, instead of initializing the model's output weights W_o from a uniform distribution, the first p rows of the weight matrix are initialized according to λ patterns observed in the data. To this end, we initialize the nth row of W_o with pattern λ_n (equation (4)), which is a vector corresponding to the nth label-assignment observed in the training data:

$$W_o^n = i(\lambda_n) \tag{4}$$

The initialization weight i for unit l of pattern λ_n is set to 0 if the corresponding label is not present in the given instance; or to the upper bound UB of the normalized initialization weights of hidden layer h and output layer o, scaled by the number of times c the pattern occurs in the data:

$$i(\lambda_n^l) = \begin{cases} \sqrt{c} \times UB & \text{if } \lambda_n^l = 1 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$UB = \frac{\sqrt{6}}{\sqrt{|h| + |o|}} \tag{6}$$

We follow Glorot and Bengio (2010) in using $\sqrt{6}$ as normalization factor for UB, and limit the number of patterns λ to the most frequently observed label assignments.

Our model uses three types of features representing the screenplay's lexical make up, its underlying character relations, and interactions.

Lexical Features An obvious feature class is the language of the movie. Comedies will be characterized by a different vocabulary compared to thrillers or historical drama. We thus represent each script as a vector of 7,500 dimensions corresponding to the most frequent words in the training corpus. Vector components were set to the

words' tf-idf values. Words in scripts were further annotated with their sentiment values using the AFINN lexicon (Nielsen, 2011), a list of words scored with sentiment strength within the range $[-5,+5]$. We extracted several features based on these sentiment values such as the sentiment score of the entire movie, the number of scenes with positive/negative sentiment, the ratio of positive to negative scenes, and the minimum and maximum scene sentiment. From scene headings, we were also able to extrapolate the number of internal and external locations per script.

Graph-based Features Our graph-based features are similar to those described in Gorinski and Lapata (2015). Specifically, we view screenplays as weighted, undirected graphs, where vertices correspond to movie characters and edges denote character-to-character interactions (essentially the number of times two characters talk to each other or are involved in a common action). From the graph we extract features corresponding to the number of main and supporting characters, which we identify by measuring their centrality in the movie network (e.g., the number of edges terminating in a given node). We also estimate character polarity by summing the sentiment of each character's utterances as well as the ratio of positive to negative characters in a given script.

Interaction-based Features We extract features based on how often any two characters interact, i.e., whether they are engaged in a conversation or in the same event (e.g., if a character kills another). We identify interactions as described in Gorinski and Lapata (2015) and measure the number of interactions per scene and movie, the number of positive and negative interactions, and their ratio.

4.2 Movie Overview Decoder

Our decoder generates a movie overview from the multi-label encoding described above. For this, we adapt the LSTM architecture of Wen et al. (2015; 2016) which was originally designed for dialogue act generation (e.g., given input inform(type="hotel", count="182", dogsallowed="dontcare"), the network outputs "*there are 182 hotels if you do not care whether dogs are allowed*"). The network performs content selection, i.e., decides which attribute labels to talk about, while generating the sentences describing them.

As outlined in the lower part of Figure 2, a sigmoid control gate feeds a content vector, p_0, into a traditional LSTM cell to generate a natural language surface form. At each timestep t, the output word w_t is drawn from an output distribution conditioned on the previous hidden layer h_{t-1} *as well as* the previous content vector p_{t-1}. The content selection cell effectively acts as a *sentence planner*, retaining or omitting information from the original vector p_0 at every time step t to guide the sentence generating LSTM cell. Our LSTM architecture is defined by the following equations:

$$i_t = \sigma(W_{wi}w_t + W_{hi}h_{t-1}) \quad (7)$$
$$f_t = \sigma(W_{wf}w_t + W_{hf}h_{t-1}) \quad (8)$$
$$o_t = \sigma(W_{wo}w_t + W_{ho}h_{t-1}) \quad (9)$$
$$\hat{c}_t = \tanh(W_{wc}w_t + W_{hc}h_{t-1}) \quad (10)$$
$$r_t = \sigma(W_{wr}w_t + W_{hr}h_{t-1}) \quad (11)$$
$$p_t = r_t \odot p_{t-1} \quad (12)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t + \tanh(W_{pc}p_t) \quad (13)$$

where σ is the sigmoid function, $i_t, f_t, o_t, r_t \in [0,1]^n$ are input, forget, output, and reading gates respectively, and $\hat{c}_t$ and c_t are proposed cell value and true cell value at time t.

In the original paper, the input p_0 to the LSTM is a 1-hot representation of the information that should be included in the natural language output. In our setup, we relax this constraint such that each element of $p_0 \in [0,1]$, i.e., we directly use the output of the multi-label encoder just described.

4.3 Training

The proposed architecture is trained jointly in an end-to-end fashion, minimizing the objective:

$$F(\theta) = \sum_t y_t^T \log(\hat{y}_t) + ||p_T|| + \sum_{t=0}^{T-1} \eta\xi^{||p_{t+1}-p_t||} \quad (14)$$

where y_t and $\hat{y}_t$ are the observed and predicted word distributions over the training data, p_T is the content vector at the final time index T, p_0 is the initial content vector as given by the encoder network, and η, ξ are training constants. The second term in the objective penalizes the network for generating output without realizing all required labels, while the third term deters the network from utilizing more than one label at any given time step.

The model is trained on pairs of scripts and sentences extracted from Jinni. To give a concrete example, a training instance for the *Plot* sentence from Figure 1 would consist of the features representing the movie's screenplay, and the overview's

Attributes	ZeroR	NB	DS	SVM	Lib	MLE
Mood	43.6	51.2	47.1	45.3	50.7	**58.4**
Plot	31.3	36.7	35.4	31.5	39.6	**43.9**
Genre	37.1	52.4	45.8	40.6	54.9	**55.3**
Attitude	63.0	68.5	67.3	64.0	71.6	**76.5**
Place	51.3	49.7	54.9	51.4	54.2	**58.6**
Flag	51.7	49.3	54.7	51.4	50.9	**57.0**
All	46.3	51.3	50.9	47.4	53.7	**58.3**

Table 2: Attribute identification (average %F1 across 10 folds). Best system in bold.

Attributes	ZeroR	Lib	MLE
Mood	43.4	48.0	**61.6**
Plot	32.8	38.9	**42.9**
Genre	37.8	54.6	**58.5**
Attitude	61.2	69.0	**73.1**
Place	48.2	46.1	**54.4**
Flag	48.8	46.4	**54.0**
All	45.4	50.5	**57.4**

Table 3: Attribute identification (%F1; test set).

Plot sentence "*The plot revolves around special agents, mind games, and a psychopath.*". The *Plot* multi-label network encodes the script into content vector p_0, and the LSTM learns which "labels" represented in p_0 to talk about while its training objective discourages to leave too many labels unmentioned. The observed output error is back-propagated through the LSTM and the embedding network using stochastic gradient descent (Bottou, 1991) with decaying learning rate.

5 Evaluation

In this section we report our evaluation experiments. We begin by assessing how good our encoder is at capturing screenplay content and then proceed to evaluated the generated overviews themselves.

5.1 How Good is the Encoder?

In order to assess encoder's ability to induce structure over screenplays, we focus solely on the top part of the architecture in Figure 2. Specifically, we trained stand-alone models for the six attributes shown in Table 1 on the gold data provided in the Jinni dataset. All networks used the same features introduced earlier and were initialized using the pattern-based method of Kurata et al. (2016). To better capture the fact that we are dealing with multi-label assignments, we used the global error function described in Zhang and Zhou (2006). Given the network output vector $\hat{y}$ for input x, the true bag of label assignments y and its complement $\bar{y}$, the error observed for each instance is computed as:

$$E = \frac{1}{|y||\bar{y}|} \sum_{(k,l)\in y\times\bar{y}} \exp(-(\hat{y}_k - \hat{y}_l)) \qquad (15)$$

The networks were trained with stochastic gradient descent during back propagation, using the same method as for the full model.

We compared our multi-label encoders (MLE) against several baselines. These include assigning the most frequent attribute labels to each movie based on the attributes' mean distribution (ZeroR), Naive Bayes (NB), Decision Stump (DS), LibLinear (Lib; Fan et al., 2008) and Support Vector Machines (SVMs; Chang and Lin, 2011). For each comparison system, we trained a binary classifier per attribute label using features identical to the ones used for the MLE.

Table 2 shows F1 performance on the training data for MLE and comparison systems, averaged over 10 folds. As can be seen, MLE performs best, followed by LibLinear. Table 3 compares MLE, ZeroR, and Lib, the strongest baseline, on the test set using F1 and the best parameters found for each system during cross-validation. As can be seen, MLE outperforms Lib across attributes, and is superior to ZeroR by a large margin. F1 differences between MLE and LibLinear are significant ($p < 0.01$), using approximate randomization testing (Noreen, 1989).

Overall, the results in Tables 2 and 3 indicate that the classification task is hard. This is especially true for *Plot* which has the largest number of labels. Nevertheless, the multi-label encoders introduced here achieve good performance on their own, indicating that they are able to capture the content of the screenplay, albeit approximately.

5.2 How Good is the Decoder?

We next evaluate the performance of the jointly trained system which we call MORGAN as a shorthand for **M**ovie **O**ve**R**view **G**ener**A**tio**N** model. MORGAN is trained on pairs of screenplays and their corresponding verbalizations in the Jinni dataset. Unfortunately, our dataset is relatively small for neural network training; it contains 617 movies only, i.e., there are 617 sentences for each attribute. To alleviate this problem, we augmented the data as follows. We extracted sentence templates from the training set (209 in total), ex-

Mood	T can be described as M_1 and M_2 The mood of T is M_1.
Plot	The plot centers around a P_1, P_2, and P_3 The plot revolves around P_1, P_2, and P_3
Genre	The main genres are G_1, G_2, and G_2 T is M_1 and M_2 movie
Attitude	In approach, T is A_1 The pacing is A_1
Place	The setting is L_1 It is located in L_1
Flag	Note that the movie involves F_1 and F_2 Note that it includes F_1, F_2, and F_3

Table 4: Template sentences extracted from Jinni overviews. Variable T is filled by the movie's title, whereas M, G, P, A, L, F correspond to values for attributes *Mood*, *Genre*, *Plot*, *Attitude*, *Place*, and *Flag*, respectively.

amples of which are shown in Table 5. We replaced the title and attribute values with variables (shown as capital letters in the table). We then used the templates to generate additional data for each movie by substituting attribute variables in template sentences with permutations of the movie's gold-standard labels. We thereby obtained a total of 31,000 training instances.

The model was trained with a learning rate of 0.5, using a decay of 0.01 over 50 epochs, fixing it for subsequent epochs. Constants η and ξ in equation (14) were set to 10^{-4} and 100, respectively. At test time, we used screenplay features as input and generated one sentence per attribute. We arranged these into an overview following the ordering *Mood* $\gg$ *Plot* $\gg$ *Genre* $\gg$ *Attitude* $\gg$ *Place* $\gg$ *Flag* which is fixed and attested in all overviews in our dataset.

We compared MORGAN against several systems: (1) a random baseline, selecting for each movie and attribute type a random sentence from the training set; (2) a nearest-neighbor baseline (NN) which uses the same screenplay features as MORGAN (and cosine similarity) to identify the closest matching script in the training data, and rehashes its overview as output; (3) an attention-based LSTM (Bahdanau et al., 2015) trained on script sentence pairs (31,000 in total); and (4) six attention-based LSTMs, one per attribute type, trained on script sentence pairs (on average 5,200 per LSTM). The attention LSTMs were trained on the same screenplay features as MORGAN, with the attention mechanism at each timestep t focusing on parts of the input. Example overviews generated by MORGAN, the attention LSTMs, and the nearest neighbor system are shown in Table 6.

Models	BLUE	Coherence	Grammaticality
Random	38.0	2.42*	3.83
NN	40.4	3.45	3.93
Attn	23.0	2.93*	3.91
typAttn	37.9	3.20	3.80
MORGAN	**42.0**	**3.72**	**4.08**
Jinni	—	4.27	4.22

Table 5: BLEU scores and mean coherence and grammaticality ratings for movie overviews. * significantly different from MORGAN ($p < 0.05$). Best performing system shown in bold.

We evaluated system output with multi-reference BLEU[4] (Papineni et al., 2002), using sentences from the extended gold-standard as references. Table 5 (first column) summarizes our results. As can be seen, MORGAN outperforms the attention based models, the nearest neighbor system, and the random baseline. The attention-based models cannot succinctly capture the movie's content in order to render it into meaningful sentences. Although the generated sentences are more or less grammatical on their own (see Table 6), the generated overview lacks coherence, and is fairly repetitive. The model does not reliably learn what type of information to focus on for the generation task. For MORGAN this problem is alleviated during the encoding step, which performs content distillation prior to generating overview sentences.

5.3 How are System Overviews Perceived by Humans?

In addition to evaluating system output automatically, we are also interested in how it is perceived by humans. To this end, we ran two judgment elicitation studies on Amazon Mechanical Turk. Both experiments were conducted on 12 movies. In a pre-test we asked 20 workers whether they had seen the movies in our test set and chose the three most popular ones from each of the genres Action, Comedy, Drama, and Romance.

In our first experiment Turkers were presented with an overview taken from the Jinni gold standard, MORGAN or one of the comparison systems and asked to rate its coherence (i.e., whether it was readily comprehensible or difficult to follow) on a scale from 1 (incoherent) to 5 (coherent). Subsequently, they had to rate the grammaticality of

[4]We use NLTK's (http://www.nltk.org/) implementation of BLEU, and report the interpolation of BLEU 1 through 4.

Nearest Neighbor
The mood of Burn after Reading is stylized. The plot revolves around love and romance, motives, buddies, and friends. Its main genres are romance, crime, and comedy. In approach, Burn after Reading is semi serious. It takes place, at least partly, in the USA. Note that Burn after Reading involves violent content, profanity, and sexual content.
attnLSTM
The setting is Europe. It is a sci-fi, adventure, and adventure movie. Note that Burn after Reading includes violent content. Burn after Reading is set in the USA. It is located in the USA. It is set, at least in part, in the USA.
typed attnLSTM
The mood of Burn after Reading is bleak, touching, and feel good. The plot revolves around ambition motives, love and romance, and fall in love. It is an adventure and fantasy, and features thriller themes. In approach, Burn after Reading is serious. Burn after Reading is set, at least in part, in the USA. Note that Burn after Reading involves violent content.
MORGAN
The mood of Burn after Reading is feel good, exciting, and clever. The plot centers around ambition, special agents, and partners. Burn after Reading is a comedy movie. In approach, it is realistic and semi serious. It is set, at least in part, in the USA. Note that it involves profanity.
JINNI
Burn after Reading can be described as cynical, clever, and exciting. The plot revolves around espionage , disorder, and escapades. Burn after Reading's main genres are comedy and crime. In approach, it is not serious and realistic. Burn after reading is set, at least in part, in an urban environment. It is located in Virginia. Note that it involves sexual content, profanity, and violent content.

Nearest Neighbor
Lara Croft: Tomb Raider can be described as touching, offbeat, and humorous. The plot revolves around a psychopath, a chase, and couples motives. Its main genres are drama and comedy. In approach, Lara Croft: Tomb Raider is semi serious. Lara Croft: Tomb Raider is set in the USA.
attnLSTM
Lara Croft: Tomb Raider is set, at least in part, in a dangerous and in a haunted house. The plot centers around buddies, disorder, and criminal heroes motives. The plot centers around escapades, supernatural, and deadly. The setting is a school or college environment. Note that Lara Croft: Tomb raider includes drugs/alcohol and sexual content. The mood of Lara Croft: Tomb Raider is rough, being, and contemplative.
typed attnLSTM
The mood of Lara Croft: Tomb Raider is suspenseful, exciting, and stylized. The plot revolves around a police investigation, cons and scams, and couple relations motives. Lara Croft: Tomb Raider's main genres are fantasy, action, and comedy. In approach, it is serious and fantastic. Lara Croft: Tomb Raider is set, at least in part, in the USA. Note that Lara Croft: Tomb Raider involves profanity and violent content.
MORGAN
Lara Croft: Tomb Raider can be described as suspenseful and rough. The plot centers around danger, a master villain, and deadly. Lara Croft: Tomb raider is an adventure, thriller, and action movie. In approach, it is fantastic. Lara Croft: Tomb Raider takes place, at least partly in Europe. Note that it involves profanity.
JINNI
Lara Croft: Tomb Raider can be described as stylized, exciting, and suspenseful. The plot revolves around time travel, a treasure hunt, and a master villain. The main genres are action, adventure, and fantasy. In approach, it is fantastical and serious. Note that it involves violent content.

Table 6: Overviews generated by MORGAN and comparison systems for "Burn after Reading" (top) and "Lara Croft; Tomb Raider" (bottom).

each overview sentence, again on a scale from 1 (ungrammatical) to 5 (grammatical) and decide whether it appropriately described aspects of the movie's content ("Yes", "No", "Unsure"). We elicited five responses for each overview across six systems (Jinni, typAttn, Attn, Random, NN, and MORGAN) and 12 movies. Finally, participants had to answer a question relating to the movie's content, to make sure that they had actually seen the movie. We discarded responses with wrong answers to the content question. Examples of the overviews participants judged are given in Table 6.

Model	Mood	Plot	Genre	Attitude	Place	Flag	All
Random	37.7	39.6	34.0	43.4	35.8	50.9	19.0*
NN	78.6	67.9	71.4	66.1	58.9	91.1	58.9*
Attn	38.2	38.2	38.2	41.8	51.0	34.5	40.0*
typAttn	60.0	60.0	53.3	57.8	**66.7**	64.4	40.0*
MORGAN	**89.5**	**73.7**	**80.7**	**71.9**	63.2	**89.5**	82.5
Jinni	91.1	89.3	92.9	82.1	67.9	75.0	91.1

Table 7: Proportion of sentences and overviews (All) which describe the movie accurately. * significantly different from MORGAN ($p < 0.05$). Best performing system per attribute is in bold.

Table 5 (columns 2 and 3) summarizes the results of our first judgment elicitation study. All systems perform well with regards to grammaticality. This is not surprising for Random and NN which do not perform any generation. Attn and typAttn also perform well with MORGAN achiev-

ing highest scores for grammaticality amongst automatic systems. Grammaticality differences between the various systems in Table 5 and the Jinni gold standard are not statistically significant (using a one-way ANOVA with post-hoc Tukey HSD tests). Overviews generated by MORGAN are perceived as more coherent in relation to those generated by comparison systems, even though the model does not explicitly take coherence into account. MORGAN overviews are not significantly different in terms of coherence from Jinni, typAttn, and NN, but are significantly better than Random and Attn.

Table 7 shows the percentage of sentences (per attribute and overall) which participants think describe the movie's content felicitously. MORGAN identifies most aspects of the movie successfully, in some cases close to (*Mood, Place*) or even better (*Flag*) than the original Jinni overview. MORGAN is significantly better compared to all other models but not significantly worse than Jinni (using a χ^2 test; see last column in Table 7).

In a second experiment, participants were presented with six overviews for a movie (from Jinni, Attn, typAttn, Random, NN, and MORGAN) and asked to rank them (equal ranks were not allowed) in order of relevance (i.e., whether they express content relevant to the movie). Again, we obtained five responses for each movie. As can be seen in Table 8, while Jinni is ranked first most of the time, MORGAN is ranked second followed by the NN system. We further converted the ranks to ratings on a scale of 1 to 6 (assigning ratings 6...1 to rank placements 1...6) and performed and ANOVA which showed that all systems are significantly ($p < 0.05$) worse than Jinni but MORGAN is significantly better than the comparison systems.

Model	1st	2nd	3rd	4th	5th	6th	AvgRank
Random	1.0	5.8	16.3	22.1	19.2	**35.6**	4.59
NN	5.8	19.2	**24.0**	23.1	15.4	12.5	3.60
Attn	3.8	13.5	20.2	**28.8**	16.3	17.3	3.92
typAttn	1.9	7.7	15.4	10.6	**33.6**	30.8	4.58
MORGAN	8.7	**42.3**	22.1	12.5	12.5	1.9	2.71
Jinni	**78.8**	11.5	1.9	2.9	2.9	1.9	1.45

Table 8: Relevance rankings (shown as proportions) given to overviews by human subjects. Most frequent rank per system and Jinni is in bold.

6 Conclusions

In this work we have presented a novel approach to automatic movie content analysis. We have assembled a new dataset which combines ScriptBase (Gorinski and Lapata, 2015), a corpus of movie scripts, with information gathered from Jinni, a large movie database. We proposed an end-to-end model for movie overview generation via *multi-attribute* encoders and a *semantically conditioned* LSTM decoder. Experimental results show that our encoders are capable of distilling meaningful structures from the screenplay. When applied to the overview generation task, our end-to-end model outperforms a standard attention-based LSTM. Human evaluation also indicates the overviews generated by our model are felicitous, informative, and rated favorably by humans.

In the future, we would like to investigate how attribute-specific features can improve performance compared to our more general feature set which is invariant for each sentence type. It would also be possible to equip the model with a hierarchical decoder which generates a document instead of individual sentences. Although currently our model relies solely on textual information, it would be interesting to incorporate additional modalities such as video (Zhou et al., 2010) or audio (e.g., we expect comedies to be *visually* very different from thrillers, or romantic movies to have a different *score* from superhero movies). Finally, we would like to examine whether the content analysis presented here can extend to different types of fiction such as novels or short stories.

Acknowledgments We thank the NAACL reviewers for their constructive feedback. We gratefully acknowledge the financial support of the European Research Council (award number 681760).

References

Apoorv Agarwal, Sriramkumar Balasubramanian, Jiehan Zheng, and Sarthak Dash. 2014a. Parsing Screenplays for Extracting Social Networks from Movies. In *Proceedings of the 3rd Workshop on Computational Linguistics for Literature*. Gothenburg, Sweden, pages 50–58.

Apoorv Agarwal, Sarthak Dash, Sriramkumar Balasubramanian, and Jiehan Zheng. 2014b. Using Determinantal Point Processes for Clustering with Application to Automatically Generating and Drawing xkcd Movie Narrative Charts. In *Proceedings of the 2nd Academy of Science and Engineering International Conference on Big Data Science and Computing*. Stan-

ford, California.

Apoorv Agarwal, Jiehan Zheng, Shruti Kamath, Sriramkumar Balasubramanian, and Shirin Ann Dey. 2015. Key Female Characters in Film Have More to Talk About Besides Men: Automating the Bechdel Test. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Denver, Colorado, pages 830–840.

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

David Bamman, Brendan O'Connor, and Noah A. Smith. 2013. Learning latent personas of film characters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Sofia, Bulgaria, pages 352–361.

David Belanger and Andrew McCallum. 2016. Structured prediction energy networks. In *Proceedings of the 33rd International Conference on Machine Learning*. New York, pages 983–992.

Léon Bottou. 1991. Stochastic gradient learning in neural networks. In *Proceedings of the 7th Neuro-Nîmes International Conference*. EC2, Nîmes, France.

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

Liang-Chieh Chen, Alexander G Schwing, Alan L Yuille, and Raquel Urtasun. 2015. Learning deep structured models. In *Proceedings of the 32nd International Comference on Machine Learning*. Lille, France, pages 1785–1794.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Doha, Qatar, pages 103–111.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of NAACL: HLT*. San Diego, CA, pages 93–98.

Mita K. Dalal and Mukesh A. Zaveri. 2011. Automatic Text Classification: A Technical Review. *International Journal of Computer Applications* 28(2):37–40.

Cristian Danescu-Niculescu-Mizil and Lillian Lee. 2011. Chameleons in Imagined Conversations: A New Approach to Understanding Coordination of Linguistic Style in Dialogs. In *Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics*. Portland, Oregon, pages 76–87.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A Library for Large Linear Classification. *Journal of Machine Learning Research* 9 (Aug):1871–1974.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*. Chia Laguna Resort, Sardinia, Italy, pages 249–256.

Shantanu Godbole and Sunita Sarawagi. 2004. Discriminative methods for multi-labeled classification. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, pages 22–30.

Philip John Gorinski and Mirella Lapata. 2015. Movie script summarization as graph-based scene extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Denver, Colorado, pages 1066–1076.

John R. Hershey, Jonathan Le Roux, and Felix Weninger. 2014. Deep unfolding: Model-based inspiration of novel deep architectures. *CoRR* abs/1409.2574.

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

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

Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep structured output learning for unconstrained text recognition. *CoRR* abs/1412.5903.

Gakuto Kurata, Bing Xiang, and Bowen Zhou. 2016. Improved neural network-based multi-label classification with better initialization leveraging label co-occurrence. In *Proceedings of the 2016 Conference of the North Amer-*

ican Chapter of the Association for Computational Linguistics: Human Language Technologies. San Diego, California, pages 521–526.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas, pages 1203–1213.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California, pages 720–730.

Finn A. Nielsen. 2011. A new ANEW: Evaluation of a word list for sentiment analysis in microblogs. In *Proceedings of the ESWC2011 Workshop on Making Sense of Microposts: Big things come in small packages*. Heraklion, Crete, pages 93–98.

Eric Noreen. 1989. *Computer Intensive Methods for Testing Hypotheses: An Introduction*. Wiley, Hoboken, New Jesey.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA, pages 311–318.

Jesse Read, Bernhard Pfahringer, and Geoff Holmes. 2008. Multi-label classification using ensembles of pruned sets. In *Proceedings of the 8th International Conference on Data Mining*. Pisa, Italy, pages 995–1000.

Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. 2011. Classifier chains for multi-label classification. *Machine Learning* 85(3):333–359.

Alexander G. Schwing and Raquel Urtasun. 2015. Fully connected deep structured networks. *CoRR* abs/1503.02351.

Fabrizio Sebastiani. 2002. Machine Learning in Automated Text Categorization. *Association for Computing Machinery: Computing Surveys* 34(1):1–47.

Veselin Stoyanov, Alexander Ropson, and Jason Eisner. 2011. Empirical risk minimization of graphical model parameters given approximate inference, decoding, and model structure. In *AISTATS*. pages 725–733.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*. pages 3104–3112.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 1422–1432.

Grigorios Tsoumakas and Ioannis Katakis. 2006. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining* 3(3).

Grigorios Tsoumakas and Ioannis Vlahavas. 2007. Random k-labelsets: An ensemble method for multilabel classification. In *European Conference on Machine Learning*. Springer, pages 406–417.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 3156–3164.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal, pages 1711–1721.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, and Steve Young. 2016. Multi-domain neural network language generation for spoken dialogue systems. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California, pages 120–129.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 1480–1489.

Patrick Ye and Timothy Baldwin. 2008. Towards Automatic Animated Storyboarding. In

Proceedings of the 23rd Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence. Chicago, Illinois, pages 578–583.

Min-Ling Zhang and Zhi-Hua Zhou. 2005. A k-nearest neighbor based algorithm for multi-label classification. In *Proceedings of the IEEE International Conference on Granular Computing*. Beijing, China, volume 2, pages 718–721.

Min-Ling Zhang and Zhi-Hua Zhou. 2006. Multi-label neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering* 18(10):1338–1351.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark, pages 595–605.

Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. 2015. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*. Santiago, Chile, pages 1529–1537.

Howard Zhou, Tucker Hermans, Asmita V. Karandikar, and James M. Rehg. 2010. Movie genre classification via scene categorization. In *Proceedings of the 18th ACM International Conference on Multimedia*. New York, NY, pages 747–750.

Which Scores to Predict in Sentence Regression for Text Summarization?

Markus Zopf, Eneldo Loza Mencía, and Johannes Fürnkranz
Research Training Group AIPHES / Knowledge Engineering Group
Department of Computer Science, Technische Universität Darmstadt
Hochschulstraße 10, 64289 Darmstadt, Germany
{zopf@aiphes,eneldo@ke,juffi@ke}.tu-darmstadt.de

Abstract

The task of automatic text summarization is to generate a short text that summarizes the most important information in a given set of documents. Sentence regression is an emerging branch in automatic text summarizations. Its key idea is to estimate the importance of information via learned utility scores for individual sentences. These scores are then used for selecting sentences from the source documents, typically according to a greedy selection strategy. Recently proposed state-of-the-art models learn to predict ROUGE recall scores of individual sentences, which seems reasonable since the final summaries are evaluated according to ROUGE recall. In this paper, we show in extensive experiments that following this intuition leads to suboptimal results and that learning to predict ROUGE precision scores leads to better results. The crucial difference is to aim not at covering as much information as possible but at wasting as little space as possible in every greedy step.

1 Introduction

More and more data is generated in textual form in newspapers, social media platforms, and microblogging services and it has become impossible for humans to read, comprehend, and filter all the available data. Automatic summarization aims at mitigating these problems by *"taking an information source, extracting content from it, and presenting the most important content to the user in a condensed form and in a manner sensitive to the users or applications needs"* (Mani, 2001).

Very prominent in automatic text summarization is the idea of extractive summarization. In extractive summarization, summaries are not generated from scratch. Instead, sentences in the source documents, which are supposed to be summarized, are extracted and concatenated to form a summary. To be able to select sentences in a meaningful manner, it is crucial for the extractive systems to be able to estimate the utility of individual sentences.

Supervised extractive methods are usually modeled in a regression framework. Hence, this subfield of automatic summarization is called *sentence regression*. The predicted scores are used to generate a ranking of the sentences, and a greedy strategy is often used in combination with additional redundancy avoidance to select sentences which will be added to the iteratively generated summary (Carbonell and Goldstein, 1998). Another method for the selection is solving an integer linear programming (ILP) problem (Gillick et al., 2008; Hong and Nenkova, 2014) which is, however, an NP-hard problem (Filatova and Hatzivassiloglou, 2004). Even though it can be argued that the complexity is not an issue since there are good solvers for ILPs, it remains a problem when large document collections with many sentences have to be summarized or the system should be used on a large scale for many users. The greedy approach is due its simplicity and efficiency very appealing.

Crucial for building sentence regression models is the choice of the regressands which has to be predicted by the models. Most of the recent works try to predict ROUGE recall scores of individual sentences, which seems to be an obvious choice since the final summaries are also evaluated with ROUGE recall metrics (Lin, 2004; Owczarzak et al., 2012). We show in this paper that following this intuition leads to suboptimal results. In extensive experiments, we investigate sentence regression models with perfect and noisy prediction of different regressand candidates with and without redundancy avoidance. In all experiments, we observe the very same result: learning to predict ROUGE precision scores of sentences leads to better results than learning to predict ROUGE recall scores if the scores are selected

Proceedings of NAACL-HLT 2018, pages 1782–1791
New Orleans, Louisiana, June 1 - 6, 2018.

with a greedy algorithm afterwards. Our findings are in particular important for automatic summarization research since the best models currently available are sentence regression models trained to predict ROUGE recall scores. We expect that simply replacing ROUGE recall scores as regressand with ROUGE precision scores can potentially improve these state-of-the-art models further.

We note in passing that the problem is reminiscent of defining heuristics in inductive rule learning: Individual rules are typically evaluated according to their consistency (minimizing the amount of false positives) and completeness (maximizing the amount of true positives), which loosely correspond to precision and recall (Fürnkranz and Flach, 2005). Heuristics such as weighted relative accuracy, which give equal importance to both dimensions, are successfully used for evaluating single rules in subgroup discovery (Lavrač et al., 2004), but tend to over-generalize when being used for selecting rules for inclusion into a predictive rule set. The reason for this is that a lack of completeness can be repaired by adding more rules, whereas a lack of consistency can not, so that consistency or precision of individual rules should receive a higher weight in the selection task. Transferred to summarization, this means that space wasted by recall-oriented selection cannot be used anymore whereas a low recall in a partial summary can be repaired by adding more sentences.

In the following, we will first formalize the problem of extractive summarization and outline the greedy selection strategy (Section 2). Previously extractive summarization systems, in particularly sentence regression models, are summarized in Section 3. We then present an intuition why predicting ROUGE precision scores can potentially give better results in Section 4. In extensive experiments (Section 5), we actually show the previously stated hypothesis which says that selecting sentence according to ROUGE precision instead of ROUGE recall leads to better results if sentence are selected greedily.

2 Extractive Summarization

In this section, we will first formally define the problem of extractive summarization and then describe the greedy sentence selection strategy which is used by many prior works.

2.1 Problem Definition

The task in extractive summarization is to generate a list of sentences S (the summary) from given list of input sentences I (the text to summarize). The size of the generated summary S must not be longer than a predefined length l (usually measured in words or characters).

In order to select sentences, both supervised and unsupervised models are used to predict utility scores of sentences in a first phase. In a second phase, sentences are selected and concatenated to build a summary.

For evaluation, the generated summary is typically compared to human written summaries by automatic means, in many cases by computing so-called ROUGE scores (Lin, 2004).

2.2 Greedy Selection Strategy

A popular strategy to select sentences based on the previously predicted utility scores is the greedy sentence selection strategy which is described in Algorithm 1.

Algorithm 1 Greedy Sentence Selection with Redundancy Avoidance in Extractive Summarization

list of all input sentences $I = s_1, \ldots, s_n$
utility function u
desired summary length l
π = permutation of I s.t. $u(s_{\pi(1)}) \geq \cdots \geq u(s_{\pi(n)})$
$S \leftarrow \emptyset, i \leftarrow 1$
while $|S| < l$ **and** $i < n$ **do**
  if $\text{sim}(s_{\pi(i)}, S) < \theta$ **then**
    $S \leftarrow S + s_{\pi(i)}$
  end if
  $i \leftarrow i + 1$
end while
return S

According to the greedy strategy, the sentence with the highest utility score is selected first. After the best sentence has been selected, it is removed from the input list of available sentences, and the former second best sentence is considered next. Redundancy avoidance strategies are used to ensure that sentences with similar contents are not added multiple times to the summary. A simple strategy computes the similarity of the currently best sentence and all already selected sentences. If the maximum similarity exceeds a predefined threshold θ, the summarizer removes the sentences from the input list without adding it to the summary. The selection process is repeated until the desired summary length is reached. Once a decision is made, it is never revised.

3 Sentence Regression for Extractive Summarization

After the field of automatic summarization has been dominated by unsupervised extractive summarization models for some time (Carbonell and Goldstein, 1998; Erkan and Radev, 2004; Mihalcea and Tarau, 2004; Li et al., 2006), supervised regression models are more commonly used in recent years. The crucial difference is that supervised models learn to predict regressands based on training examples in a training phase whereas unsupervised models do not predict regressands. We focus on supervised extractive regression systems in this paper. Comprehensive overviews of automatic summarization (Nenkova and McKeown, 2011; Gambhir and Gupta, 2016; Yao et al., 2017) also cover unsupervised methods in more detail and include abstractive summarization methods which are out of scope for this paper.

Extractive sentence regression can be described as the task of learning regressands for individual sentences from examples. The general learning problem can be formulated as $y_i = u(x_i) + e_i$ where y_i denotes the regressand (also called dependent variable or target variable) of sentence x_i (the regressor, also called independent variable or features), and e_i denotes the ith residuum (also called error). Sentence regression aims at learning the utility function u from observed sentence-utility pairs in order to minimize the errors for unseen sentences-utility pairs.

Kupiec et al. (1995) proposed one of the first supervised summarization systems, which trains a Bayesian model to predict the probability that a sentence will be included in the summary. They criticized that although a large number of different features had been used in previous unsupervised models, no principled method to select or weight the features had been proposed at this time. Instead of generating summaries, the performance of the model was evaluated based on the classification output of the model for individual sentences. Similarly, Conroy and O'leary (2001) use a Hidden Markov Model to predict the probability that a sentence is included in a reference summary.

The model proposed by Li et al. (2006) already predicts utility scores for individual sentences. The model weights are, however, not learned in a supervised training but assigned by humans. Li et al. (2007) extends this previously proposed unsupervised model and used a support vector regression (SVR) model in the DUC 2007 shared task (Over et al., 2007). Both Li et al. (2006) and Li et al. (2007) use a greedy selection strategy. Instead of learning to predict the probability of appearance of a sentence in a summary (Kupiec et al., 1995; Conroy and O'leary, 2001), Li et al. (2007) use the average and maximum text similarity of candidate sentences and reference summaries as regressands. Ouyang et al. (2011) also applied SVR but used the sum of word probabilities as regressand. Their system therefore also tends to select longer sentences similarly to systems which use ROUGE recall.

PriorSum (Cao et al., 2015b) follows Li et al. (2007) and presents a linear regression framework which uses prior and document dependent features. As regressand, ROUGE-2 recall is used.

Cao et al. (2015a) propose a hierarchical regression process which predicts the importance of sentences based on its constituents. ROUGE-1 recall and ROUGE-2 recall are used as regressand for sentences. For sentence selection, they implement both a greedy selection and a selection based on integer linear programming.

The Redundancy-Aware Sentence Regression (Ren et al., 2016) framework models both importance and redundancy jointly. They train a multi-layer perceptron which then predicts relative importance utilities based on ROUGE-2 recall scores.

REGSUM (Hong and Nenkova, 2014) predicts sentence importance based on word importance and additional features. They use a greedy selection strategy with additional redundancy avoidance which only appends sentences to the summary if the maximum cosine similarity to already selected sentences is lower than a fixed threshold.

We summarize that ROUGE recall is often used in the field of sentence regression in combination with a greedy selection and an additional redundancy avoidance strategy. In the following, we first describe the underlying intuition of using ROUGE recall. Second, we describe why using ROUGE precision instead can be potentially better. Later, we show in the experiments that using ROUGE precision is not only theoretically appealing but also works better in practice than ROUGE recall.

4 ROUGE Recall vs. ROUGE Precision

The ROUGE metric (Lin, 2004) is the method of choice for the evaluation of generated summaries in the field of automatic summarization. Its idea is to compute the similarity between automatically generated summaries and references summaries, which are typically provided by humans.

ROUGE can be viewed as an evaluation measure for an information retrieval task in which precision and recall can be measured. Let E be a set of elements, $R \subset E$ the multiset of desired elements in the reference output, $G \subset E$ is the generated output multiset, and $|.|$ the size of a multiset. Then, the recall is defined as

$$r(G, R) = \frac{|G \cap R|}{|R|} \quad (1)$$

and measures how much of the desired content was returned by the system. On the other hand, precision is defined as

$$p(G, R) = \frac{|G \cap R|}{|G|}, \quad (2)$$

and measures how much of the returned content was actually desirable. We define the intersection $\cap$ of two multisets as the smallest multiset S with $\sigma_S(e) = \min(\sigma_G(e), \sigma_R(e)) \ \forall e \in G, R$, where $\sigma_S(e)$ indicates the number of appearances of element e in set S.

In ROUGE-n, the multiset E is defined as the set of all n-grams, the desired reference multiset R contains all n-grams in a reference summary, and the multiset G contains all n-grams in the system summary. We use multisets and not sets since the same n-gram can be contained multiple times in a text.

When ROUGE was first introduced as the evaluation metric for the DUC 2003 shared task (Over et al., 2007), Lin and Hovy (2003) reported that metrics based on ROUGE recall scores have a good agreement with human judgments. A summary with a high ROUGE recall will contain many n-grams which also appear in the reference summaries. Owczarzak et al. (2012) showed that ROUGE-2 recall is the best variant (highest agreement with human judgments) of ROUGE recall if automatically generated summaries have to be evaluated. ROUGE-2 recall is therefore often used to evaluate automatic summarization systems.

Crucial for the use of ROUGE recall is the length limitation of the generated summaries.

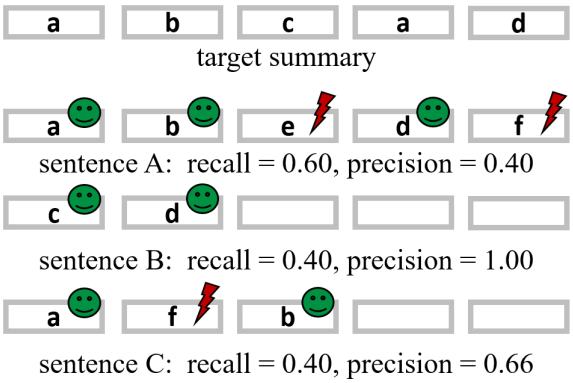


Figure 1: Exemplary illustration of selecting sentences according to precision and recall. The target summary has 5 slots. Sentence A will be selected according to recall since it has a recall scores of 0.6 whereas sentence B and C only have a recall score of 0.4. Sentence A, however, occupies already all available slots in the summary. No more sentence can be selected. Sentence B will be first selected according to precision due to a precision scores of 1.0. After the selection of sentence B, 3 slots are still available in the summary which can be used to fit sentence C to improve the overall summary recall to 0.8.

Usually, the generated summaries are limited to a fixed number of words or characters. Without such a length restriction, systems would be able to generate arbitrary long texts to increase the recall.

Summarization systems aim at maximizing ROUGE recall scores of the generated summaries, since the final summaries are evaluated with ROUGE recall. Greedy extractive summarization approaches try to maximize the overall ROUGE recall of a summary by incrementally adding sentences with a high ROUGE recall to the summary. The idea of this strategy is to pack as much important content as possible into the summary in every step in order to increase the ROUGE recall of the resulting summary. What is usually not considered is the fact that this strategy tends to select longer sentences, since longer sentences tend to have a higher recall. They, however, can contain proportionally more unimportant information, for example in subordinate clauses. As a result, fewer sentences can be selected since the maximum length of the summary is reached earlier.

An alternative strategy, which has not been discussed in the literature so far, is to select sentences according to their ROUGE precision scores. The idea behind this approach is not to cover as much

as information as possible but to waste as little space as possible. Selecting sentences according to precision will not have a bias for longer sentences but for short and dense sentences. Since this strategy tends to selected shorter sentences, more sentence can be included in the summary, which can, in turn, again result in a higher ROUGE recall of the resulting summary.

Figure 1 shows an example in which selecting sentences according to ROUGE precision leads to a higher ROUGE recall score of the resulting summary than selecting sentences according to ROUGE recall. In the following section, we will show that the intuition described in this section is not only appealing in theory, but can also be substantiated in empirical experiments.

We summarize that selecting sentences according to ROUGE precision scores can, intuitively, be better than selecting sentences according to ROUGE recall scores even though the final summaries are always evaluated with ROUGE recall metrics.

5 Experimental Setup

We now present the experimental setups in which we test different regressand candidates for sentence regression in three different, well-known multi-document summarization (MDS) corpora. We used the MDS corpora from the DUC 2004[1], TAC 2008, and TAC 2009[2] summarization shared tasks. All corpora contain 10 input documents and 4 reference summaries for each topic. The number of topics are 50, 46, and 44, respectively. We simulate in the experiments the outcomes of regression models which use different regressands. This will provide us with theoretical insights on which regressand candidates should be considered in regression models and will answer the main question of this paper: *Which scores to predict in sentence regression for text summarization?* For our experiments, we produce summaries containing 665 characters for DUC2004 and summaries containing 100 words for TAC2008 and TAC2009.

5.1 Regressand Candidates

The key ingredient of greedy extractive summarization is the utility function $u(.)$, which is used for sorting the sentences in the first step of Algorithm 1. In this paper, we examine 7 different regressand candidates (**in boldface**) which can be used as regressands when the utility function u is learned via supervised regression.

ROUGE-1 recall (**R1 Rec**) and ROUGE-2 recall (**R2 Rec**) are computed according to Equation 1 for all sentences in the input documents. ROUGE-n recall counts the n-gram overlap of the input sentence and the reference summaries. The more n-grams in the reference documents are covered by a sentence, the higher the score is. These regressands are usually used by prior sentence regression works.

We also compute the ROUGE-1 precision (**R1 Prec**) and ROUGE-2 precision (**R2 Prec**) for all sentences according to Equation 2. A sentence has a high ROUGE-n precision if a high rate of n-grams in the sentence match with n-grams in the reference documents. Sentences with a high density of matching n-grams are therefore preferred by ROUGE precision. The main claim of this paper is that ROGUE precision scores should be primarily considered in sentence regression works instead of ROUGE recall scores. We therefore expect that R1 Prec and R2 Prec will perform better than R1 Rec and R2 Rec.

As a reference point, we compute for each sentence the maximum similarity (**maxADW**) for and the average similarity (**avgADW**) with all sentences in the reference summaries (denoted by list S) according to a state-of-the-art ADW similarity measure (Pilehvar et al., 2013). ADW computes the semantic similarity of two sentences by finding an optimal alignment of word senses contained in the two sentences.

$$\text{maxADW}(s) = \max_{t \in \text{S}} \text{ADWsim}(s, t) \quad (3)$$

$$\text{avgADW}(s) = \frac{1}{|S|} \cdot \sum_{t \in \text{S}} \text{ADWsim}(s, t) \quad (4)$$

Computing the maximum similarity aligns with the idea that a good sentence in the input documents matches well with one sentence in the reference summary. A sentence is representative for the whole summary if it has a high average similarity with all the reference summary sentences. For each sentence, we also randomly generated (**random**) sentence scores which are used as regressand.

[1] http://duc.nist.gov
[2] https://tac.nist.gov

6 Results

6.1 Optimal Prediction without Redundancy Avoidance

In the first experiment, we investigate how helpful the predicted scores are under the assumption that the regressand candidates can be predicted perfectly. The experiment therefore shows how a systems will perform in the optimal case. We do not consider redundancy avoidance strategies in this experiment so that observed performance differences are solely due to differences in the used regressand candidates.

	DUC2004		TAC2008		TAC2009	
	R-1	R-2	R-1	R-2	R-1	R-2
R1 Rec	38.63	08.99	39.28	11.08	34.31	08.37
R2 Rec	39.23	12.07	42.39	16.20	37.42	13.03
R1 Prec	**41.29**	11.18	**43.56**	14.65	**39.45**	12.17
R2 Prec	39.18	**12.73**	43.46	**18.19**	37.81	**13.64**
maxADW	37.60	10.13	42.55	15.46	34.56	11.05
avgADW	38.50	09.62	40.97	12.43	35.48	09.34
random	31.76	04.66	29.58	04.60	29.88	04.63

Table 1: Summarization results in three different multi-document summarization corpora without redundancy avoidance. Columns R-1 and R-2 display the summary quality according to ROUGE-1 recall and ROUGE-2 recall scores, respectively.

The results of the experiment are shown in Table 1. It can be seen that in all corpora the use of ROUGE-1 precision regressands of the sentences leads to better results than using ROUGE-1 recall regressands if ROUGE-1 recall is used as evaluation metric for the final summary. Analogous results can be observed for ROUGE-2 scores. This indicates that using ROUGE recall as regressand in a sentences regression framework is not very promising. Thus, the results are a first confirmation of the previously described intuition that predicting precision scores can be better than predicting recall scores.

Table 2 provides details about the lengths of the produced summaries according to number of stems and number of sentences. The hypothesis that an algorithm that selects sentences according to recall tends to select longer sentences (stated in Section 4) is confirmed. The results therefore also confirm that longer sentences tend to have a higher recall.

In addition to the standard DUC and TAC corpora, we also report results for 2 German datasets, namely the DBS corpus (Benikova et al., 2016) and a subset of the German part of the auto-hMDS corpus (Zopf et al., 2016; Zopf, 2018). The DBS corpus contains topics from the educational domain. auto-hMDS contains heterogeneous topics retrieved from Wikipedia and automatically collected source documents retrieved from web sites. The results are displayed in Table3 and show that the results can be transferred to German. We additionally observe that ROUGE-1 precision seems to be a bit stronger in DBS compared to ROUGE-2 precision even if the resulting summaries are evaluated with ROUGE-2 recall.

	avg. stems			avg. sentences		
	D04	T08	T09	D04	T08	T09
R1 Rec	166	132	141	3.42	2.67	2.70
R2 Rec	160	129	132	4.26	3.46	3.55
R1 Prec	157	125	127	7.76	6.75	6.07
R2 Prec	157	129	126	7.10	6.13	6.09
maxADW	158	127	129	6.56	5.06	5.11
avgADW	158	126	126	5.12	4.13	4.02
random	164	131	131	6.66	5.21	4.89

Table 2: Averaged lengths of resulting summaries measured in number of stems (avg. stems) and number of sentences (avg. sentences). D04 refers to DUC2004 and T08 and T09 refer to TAC2008 and TAC2009, respectively. We count also partially contained sentences which have been cut by the ROUGE length limitation.

	DBS		hMDS	
	R-1	R-2	R-1	R-2
R1 Rec	33.48	13.89	31.94	13.38
R2 Rec	38.67	21.77	40.67	24.39
R1 Prec	**42.20**	**25.55**	**43.25**	23.01
R2 Prec	37.01	23.12	41.65	**24.96**
random	23.27	04.23	20.63	02.36

Table 3: Results as in Table 1, but for 2 datasets (DBS and auto-hMDS) containing German documents.

6.2 Optimal Prediction of F-Scores

The previous experiment clearly showed that selecting sentences according to ROUGE precision outperforms a selection according to ROUGE recall. In this experiment, we will evaluate if a trade-off between recall and precision can lead to even better results. It is, e.g., known that in inductive rule learning, parametrized measures such as the m-estimate, which may be viewed as a trade-off between precision and weighted relative accuracy, can be tuned to outperform its constituent heuristics (Janssen and Fürnkranz, 2010). In retrieval tasks, the F-measure provides a more commonly

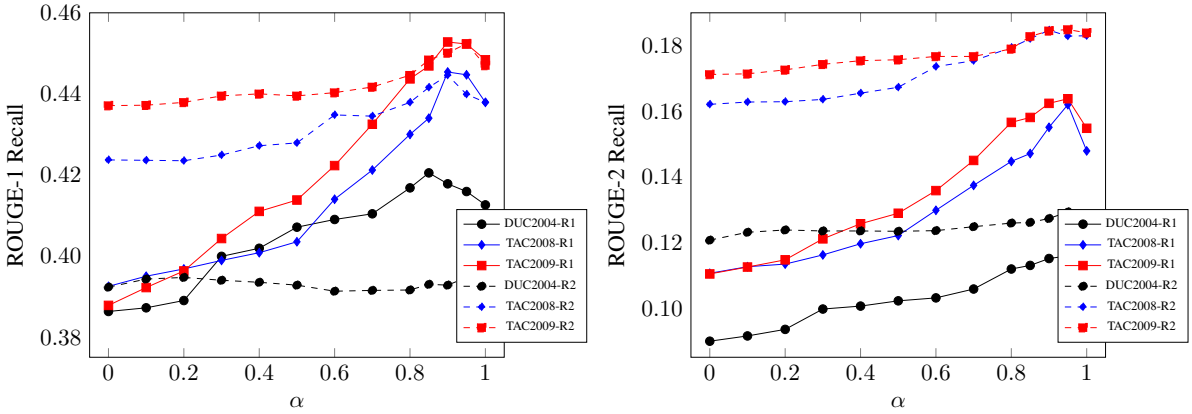


Figure 2: Results of mixing ROUGE-1/2 precision and ROUGE-1/2 recall using $F_\alpha(p, r)$-Measure in different datasets evaluated with ROUGE-1 recall (left) and ROUGE-2 recall (right). For example, the curve labeled DUC2004-R1 shows the results of mixing ROUGE-1 precision and ROUGE-1 recall in the DUC 2004 corpus.

used trade-off between precision and recall, so we chose to use this measure for our experiments. We compute for all sentences the F-measure with $0 \leq \alpha \leq 1$ as

$$F_\alpha(p, r) = \frac{1}{\frac{\alpha}{p} + \frac{1-\alpha}{r}} \quad (5)$$

where a $\alpha = 0$ is equivalent to recall and $\alpha = 1$ equals precision.

The results of the experiment, which are displayed in Figure 2, show that precision ($\alpha = 1.0$) is already close to the optimum but that incorporating also a small fraction of recall ($\alpha \approx 0.9$) leads to the best results which indicates that a slight bias towards longer sentences can improve the result even further. A possible explanation is that there are short sentences in the input documents which are considerably redundant to other high precision sentences. However, overall the trend in the results (increasing evaluation scores with increasing α, which means increasing impact of ROUGE precision) substantiate the general hypothesis of this paper, namely that sentence selection measures should target precision instead of recall.

6.3 Optimal Prediction with Redundancy Avoidance

Summarization systems usually apply a redundancy avoidance strategy in order to avoid including the same information multiple times in the summary. In this experiment, we investigate whether incorporating a simple redundancy avoidance strategy will lead to different results.

During the greedy selection process, we compute the similarity of the currently highest scoring sentence and all already selected sentences (see Algorithm 1, line 4). The highest scoring sentence will be skipped if the maximum similarity of the sentence and the already selected sentences is higher than a predefined threshold θ. We use the state-of-the-art ADW similarity measure to compute the similarities and test the quality of the generated summaries as in the previous experiments with ROUGE-1 and ROUGE-2 recall. The results of the experiment for the thresholds $\theta = 0.4, 0.5, \ldots, 1.0$ are displayed in Figure 3.

We see that sentence selection using ROUGE-1/2 precision scores (red and blue solid lines) consistently leads to better results than with ROUGE-1/2 recall scores (red and blue dashed lines) for all chosen redundancy thresholds. Selecting according to maximum ADW similarity leads to consistently better results than selecting according to the average ADW similarity. This indicates that it is better to search for sentences which align well with a part of the summary than selecting sentences which align relatively well with the whole summary. The best results are achieved with thresholds of $\theta = 0.5$ and $\theta = 0.6$ which worked well for both ROUGE-1 and ROUGE-2 recall in both datasets.

6.4 Noisy Predictions

In the previous experiments, we showed the results of a greedy summarizer which selects sentences according to perfectly predicted scores. Summarization systems are, however, not capable of pre-

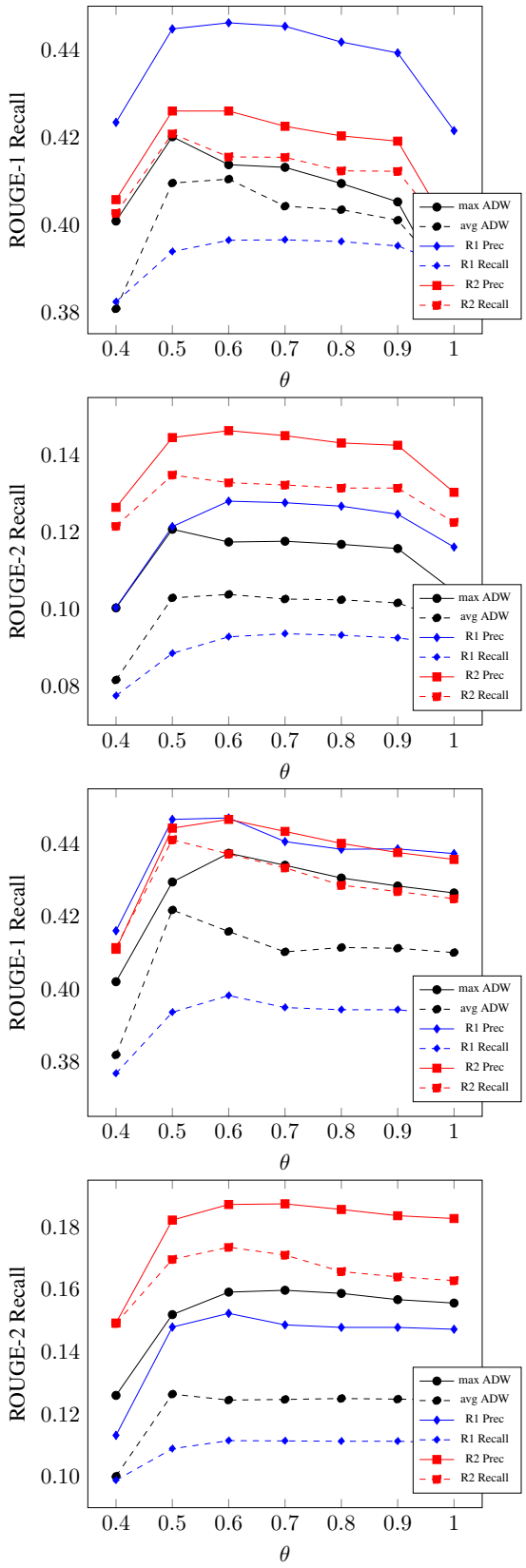


Figure 3: Summary quality assessed with ROUGE-1 recall and ROUGE-2 recall with different redundancy avoidance thresholds θ in the DUC 2004 (top half) and TAC 2008 (bottom half) datasets.

dicting the scores perfectly. We will therefore investigate whether imperfect predictions have an influence on our results in the next experiment. This will also give insights about the robustness of a greedy summarizer in the presence of imprecise predictions.

In order to get model-independent results, we simulate imperfect precisions by adding two different kinds of noise to simulate imperfect predictions, namely additive uniformly distributed continuous noise $\mathcal{U}(a, b)$ and additive Gaussian noise $\mathcal{N}(\mu, \sigma^2)$. For the uniform noise $\mathcal{U}(a, b)$, we test boundaries from $a = -0.2$, $b = 0.2$ to $a = -0.4$, $b = 0.4$. For Gaussian noise, we use mean $\mu = 0$ and variance $\sigma^2 \in \{0.05, 0.1, 0.2\}$. Based on the results in the previous section, we fix the redundancy threshold to 0.6 in this experiment. Due to the random noise, the experiments are no longer deterministic. We therefore run each experiment 10 times and report averaged results.

The results of these experiments (see Table 4) confirm that predicting ROUGE precision is always better than predicting ROUGE recall, in the presence of different kinds of noises and different noise intensities. In case strong Gaussian noise is applied (Table 4, last block), the quality of the

	score	DUC2004		TAC2008		TAC2009	
		R-1	R-2	R-1	R-2	R-1	R-2
$\mathcal{U}(-0.2, 0.2)$	R1 Rec	37.22	07.71	36.73	08.79	37.06	08.99
	R2 Rec	36.93	08.74	36.45	09.91	37.83	11.06
	R1 Prec	**42.53**	10.87	**42.19**	12.57	**43.65**	13.58
	R2 Prec	40.37	**12.04**	40.63	**14.23**	42.25	**15.49**
$\mathcal{U}(-0.3, 0.3)$	R1 Rec	36.78	07.43	35.70	08.00	36.04	08.27
	R2 Rec	35.45	07.54	34.62	08.58	36.08	09.43
	R1 Prec	**42.02**	10.45	**41.42**	11.75	**42.75**	12.83
	R2 Prec	39.56	**11.16**	38.94	**12.64**	40.91	**14.29**
$\mathcal{U}(-0.4, 0.4)$	R1 Rec	36.10	06.92	34.91	07.48	35.85	07.93
	R2 Rec	34.92	07.32	34.08	07.85	3.545	08.70
	R1 Prec	**41.27**	09.98	**40.44**	11.04	**41.63**	11.92
	R2 Prec	39.02	**10.63**	38.22	**11.74**	39.51	**12.97**
$\mathcal{N}(0, 0.05)$	R1 Rec	37.53	07.93	36.99	09.31	37.40	09.36
	R2 Rec	35.46	07.60	35.50	09.41	36.07	09.96
	R1 Prec	**43.55**	11.99	**43.59**	13.98	**45.58**	15.56
	R2 Prec	41.06	**12.92**	42.80	**16.46**	43.97	**17.48**
$\mathcal{N}(0, 0.1)$	R1 Rec	35.63	06.83	34.45	07.31	35.06	07.57
	R2 Rec	33.39	06.04	32.76	06.93	32.88	07.98
	R1 Prec	**41.70**	10.19	**41.41**	12.09	**43.06**	13.23
	R2 Prec	38.41	**10.33**	38.27	**12.43**	40.15	**13.94**
$\mathcal{N}(0, 0.2)$	R1 Rec	33.59	05.72	32.00	05.78	32.36	05.99
	R2 Rec	32.64	05.28	30.76	05.48	31.47	06.01
	R1 Prec	**38.19**	**08.01**	**37.34**	**09.00**	**38.75**	**10.06**
	R2 Prec	35.07	07.45	34.08	08.45	34.71	09.08

Table 4: Summarization results in three different multi-document summarization corpora with noisy score prediction with uniform noise (top) and Gaussian noise (bottom).

summaries decreases more strongly if ROUGE-2 precision scores are predicted, which means that predicting ROUGE-1 precision might be better than predicting ROUGE-2 precision in the case of low prediction quality.

7 Conclusions

Current state-of-the-art sentence regression systems for automatic summarization learn to predict ROUGE recall scores of individual sentences and apply a greedy sentence selection strategy in order to generate summaries. We show in a wide range of experiments that this design choice leads to suboptimal results. In all experiments, we observed the same pattern. The resulting summaries will have a lower quality if ROUGE recall scores for sentences are used instead of ROUGE precision – no matter whether or not redundancy avoidance is considered and whether or not the scores can be predicted perfectly.

In an experiment where we combined both ROUGE recall and ROUGE precision with an F-score computation, we confirmed the previously described observation that the quality of summaries tends to improve with a growing ratio of ROUGE precision vs. ROUGE recall, with a maximum performance for a ratio of $\alpha \approx 0.9$. Biasing the sentence selection slightly to longer sentences is therefore promising. This goes in line with an often applied pre-processing step in which very short sentences are discarded without further analysis (Erkan and Radev, 2004; Cao et al., 2015b).

We also presented an intuition why a selection according to ROUGE precision leads to better results. A system which selects according to ROUGE recall will tend to select longer sentences, since longer sentences tend to have a higher recall. We conclude that systems should instead of fitting iteratively as much as possible into a summary rather aim at wasting as little space as possible in every step.

For future works, it is very simple to incorporate the findings presented in this paper. Instead of learning to predict ROUGE recall scores, the regressand can simply be exchanged and the ROUGE precision can be used instead. Based on the findings in this paper, we expect that the models will benefit from this modification. We furthermore conclude that comparisons between ILP and greedy methods (Cao et al., 2015a) are biased in favor of ILP. A better comparison is possible if precision scores are used as input for greedy systems instead of recall scores.

Acknowledgments

This work has been supported by the German Research Foundation (DFG) as part of the Research Training Group Adaptive Preparation of Information from Heterogeneous Sources (AIPHES) under grant No. GRK 1994/1.

References

Darina Benikova, Margot Mieskes, Christian M Meyer, and Iryna Gurevych. 2016. Bridging the gap between extractive and abstractive summaries: Creation and evaluation of coherent extracts from heterogeneous sources. In *Proceedings of the 22nd International Conference on Computational Linguistics: Technical Papers*. pages 1039–1050.

Ziqiang Cao, Furu Wei, Li Dong, Sujian Li, and Ming Zhou. 2015a. Ranking with recursive neural networks and its application to multi-document summarization. *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence* pages 2153–2159. https://doi.org/10.1162/153244303322533223.

Ziqiang Cao, Furu Wei, Sujian Li, Wenjie Li, Ming Zhou, and Houfeng Wang. 2015b. Learning Summary Prior Representation for Extractive Summarization. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing* pages 829–833.

Jaime Carbonell and Jade Goldstein. 1998. The use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries. In *Proceedings of the 21st Annual ACM SIGIR International Conference on Research and Development in Information Retrieval (SIGIR 1998)*. ACM, pages 335–336.

John M. Conroy and Dianne P. O'leary. 2001. Text summarization via hidden Markov models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. pages 406–407. https://doi.org/10.1145/383952.384042.

Günes Erkan and Dragomir R Radev. 2004. LexRank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research* 22:457–479.

Elena Filatova and Vasileios Hatzivassiloglou. 2004. A formal model for information selection in multi-sentence extraction. *Proc. International Conference on Computational Linguistics (COLING)* pages 397–403.

Johannes Fürnkranz and Peter A. Flach. 2005. ROC 'n' Rule Learning - Towards a Better Understanding of Covering Algorithms. *Machine Learning* 58(1):39–77. https://doi.org/10.1007/s10994-005-5011-x.

Mahak Gambhir and Vishal Gupta. 2016. Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review* 47(1):1–66. https://doi.org/10.1007/s10462-016-9475-9.

Dan Gillick, Benoit Favre, and Dilek Hakkani-Tür. 2008. The ICSI Summarization System at TAC 2008. *In Proceedings of the Text Analysis Conf. Workshop* .

Kai Hong and Ani Nenkova. 2014. Improving the Estimation of Word Importance for News Multi-Document Summarization. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. pages 712–721.

Frederik Janssen and Johannes Fürnkranz. 2010. On the quest for optimal rule learning heuristics. *Machine Learning* 78(3):343–379. https://doi.org/10.1007/s10994-009-5162-2.

Julian Kupiec, Jan Pedersen, and Francine Chen. 1995. A Trainable Document Summarizer. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. pages 68–73.

Nada Lavrač, Branko Kavšek, Peter Flach, and Ljupčo Todorovski. 2004. Subgroup Discovery with CN2-SD. *The Journal of Machine Learning Research* 5:153–188.

Sujian Li, You Ouyang, and Bin Sun. 2006. Peking University at DUC 2006. *Proceedings of Document Understanding Conference 2006* .

Sujian Li, You Ouyang, Wei Wang, and Bin Sun. 2007. Multi-document summarization using support vector regression. *Proceedings of Document Understanding Conference 2007* .

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*. pages 25–26.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*. Association for Computational Linguistics, pages 71–78.

Inderjeet Mani. 2001. *Automatic Summarization*. John Benjamins Publishing Co.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into texts. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. volume 85, pages 404–411. https://doi.org/10.3115/1219044.1219064.

Ani Nenkova and Kathleen McKeown. 2011. Automatic Summarization. *Foundations and Trends in Information Retrieval* 5(3):103–233. https://doi.org/10.1561/1500000015.

You Ouyang, Wenjie Li, Sujian Li, and Qin Lu. 2011. Applying regression models to query-focused multi-document summarization. *Information Processing and Management* 47(2):227–237. https://doi.org/10.1016/j.ipm.2010.03.005.

Paul Over, Hoa Dang, and Donna Harman. 2007. DUC in context. *Information Processing and Management* 43(6):1506–1520. https://doi.org/10.1016/j.ipm.2007.01.019.

Karolina Owczarzak, John M Conroy, Hoa Trang Dang, and Ani Nenkova. 2012. An assessment of the accuracy of automatic evaluation in summarization. In *Proceedings of Workshop on Evaluation Metrics and System Comparison for Automatic Summarization*. pages 1–9.

Mohammad Taher Pilehvar, David Jurgens, and Roberto Navigli. 2013. Align, Disambiguate and Walk: A Unified Approach for Measuring Semantic Similarity. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics* pages 1341–1351.

Pengjie Ren, Furu Wei, and Zhumin Chen. 2016. A Redundancy-Aware Sentence Regression Framework for Extractive Summarization. In *Proceedings of the 26th International Conference on Computational Linguistics*. pages 33–43.

Jin-ge Yao, Xiaojun Wan, and Jianguo Xiao. 2017. Recent advances in document summarization. *Knowledge and Information Systems* 53(2):297–336. https://doi.org/10.1007/s10115-017-1042-4.

Markus Zopf. 2018. auto-hMDS: Automatic Construction of a Large Heterogeneous Multi-Document Summarization Corpus. In *Proceedings of the 11th International Conference on Language Resources and Evaluation*. page (to appear).

Markus Zopf, Maxime Peyrard, and Judith Eckle-Kohler. 2016. The Next Step for Multi-Document Summarization : A Heterogeneous Multi-Genre Corpus Built with a Novel Construction Approach. In *Proceedings of the 26th International Conference on Computational Linguistics*. pages 1535–1545.

A Hierarchical Latent Structure for Variational Conversation Modeling

Yookoon Park **Jaemin Cho** **Gunhee Kim**
Department of Computer Science and Engineering & Center for Superintelligence
Seoul National University, Korea
yookoonpark@vision.snu.ac.kr, {jaemin895,gunhee}@snu.ac.kr
http://vision.snu.ac.kr/projects/vhcr

Abstract

Variational autoencoders (VAE) combined with hierarchical RNNs have emerged as a powerful framework for conversation modeling. However, they suffer from the notorious degeneration problem, where the decoders learn to ignore latent variables and reduce to vanilla RNNs. We empirically show that this degeneracy occurs mostly due to two reasons. First, the expressive power of hierarchical RNN decoders is often high enough to model the data using only its decoding distributions without relying on the latent variables. Second, the conditional VAE structure whose generation process is conditioned on a context, makes the range of training targets very sparse; that is, the RNN decoders can easily overfit to the training data ignoring the latent variables. To solve the degeneration problem, we propose a novel model named *Variational Hierarchical Conversation RNNs* (VHCR), involving two key ideas of (1) using a hierarchical structure of latent variables, and (2) exploiting an *utterance drop* regularization. With evaluations on two datasets of Cornell Movie Dialog and Ubuntu Dialog Corpus, we show that our VHCR successfully utilizes latent variables and outperforms state-of-the-art models for conversation generation. Moreover, it can perform several new utterance control tasks, thanks to its hierarchical latent structure.

1 Introduction

Conversation modeling has been a long interest of natural language research. Recent approaches for data-driven conversation modeling mostly build upon recurrent neural networks (RNNs) (Vinyals and Le, 2015; Sordoni et al., 2015b; Shang et al., 2015; Li et al., 2017; Serban et al., 2016). Serban et al. (2016) use a hierarchical RNN structure to model the context of conversation. Serban et al. (2017) further exploit an utterance latent variable in the hierarchical RNNs by incorporating the variational autoencoder (VAE) framework (Kingma and Welling, 2014; Rezende et al., 2014).

VAEs enable us to train a latent variable model for natural language modeling, which grants us several advantages. First, latent variables can learn an interpretable holistic representation, such as topics, tones, or high-level syntactic properties. Second, latent variables can model inherently abundant variability of natural language by encoding its global and long-term structure, which is hard to be captured by shallow generative processes (*e.g.* vanilla RNNs) where the only source of stochasticity comes from the sampling of output words.

In spite of such appealing properties of latent variable models for natural language modeling, VAEs suffer from the notorious *degeneration* problem (Bowman et al., 2016; Chen et al., 2017) that occurs when a VAE is combined with a powerful decoder such as autoregressive RNNs. This issue makes VAEs ignore latent variables, and eventually behave as vanilla RNNs. Chen et al. (2017) also note this degeneration issue by showing that a VAE with a RNN decoder prefers to model the data using its decoding distribution rather than using latent variables, from bits-back coding perspective. To resolve this issue, several heuristics have been proposed to weaken the decoder, enforcing the models to use latent variables. For example, Bowman et al. (2016) propose some heuristics, including *KL annealing* and *word drop* regularization. However, these heuristics cannot be a complete solution; for example, we observe that they fail to prevent the degeneracy in VHRED (Serban et al., 2017), a conditional VAE model equipped with hierarchical RNNs for conversation modeling.

The objective of this work is to propose a novel VAE model that significantly alleviates the degen-

Proceedings of NAACL-HLT 2018, pages 1792–1801
New Orleans, Louisiana, June 1 - 6, 2018.

eration problem. Our analysis reveals that the causes of the degeneracy are two-fold. First, the hierarchical structure of autoregressive RNNs is powerful enough to predict a sequence of utterances without the need of latent variables, even with the word drop regularization. Second, we newly discover that the conditional VAE structure where an utterance is generated conditioned on context, *i.e*. a previous sequence of utterances, induces severe data sparsity. Even with a large-scale training corpus, there only exist very few target utterances when conditioned on the context. Hence, the hierarchical RNNs can easily memorize the context-to-utterance relations without relying on latent variables.

We propose a novel model named *Variational Hierarchical Conversation RNN* (VHCR), which involves two novel features to alleviate this problem. First, we introduce a global conversational latent variable along with local utterance latent variables to build a hierarchical latent structure. Second, we propose a new regularization technique called *utterance drop*. We show that our hierarchical latent structure is not only crucial for facilitating the use of latent variables in conversation modeling, but also delivers several additional advantages, including gaining control over the global context in which the conversation takes place.

Our major contributions are as follows:

(1) We reveal that the existing conditional VAE model with hierarchical RNNs for conversation modeling (*e.g*. (Serban et al., 2017)) still suffers from the degeneration problem, and this problem is caused by data sparsity per context that arises from the conditional VAE structure, as well as the use of powerful hierarchical RNN decoders.

(2) We propose a novel variational hierarchical conversation RNN (VHCR), which has two distinctive features: a hierarchical latent structure and a new regularization of utterance drop. To the best of our knowledge, our VHCR is the first VAE conversation model that exploits the hierarchical latent structure.

(3) With evaluations on two benchmark datasets of Cornell Movie Dialog (Danescu-Niculescu-Mizil and Lee, 2011) and Ubuntu Dialog Corpus (Lowe et al., 2015), we show that our model improves the conversation performance in multiple metrics over state-of-the-art methods, including HRED (Serban et al., 2016), and VHRED (Serban et al., 2017) with existing degeneracy solutions such as the word drop (Bowman et al., 2016), and the bag-of-words loss (Zhao et al., 2017).

2 Related Work

Conversation Modeling. One popular approach for conversation modeling is to use RNN-based encoders and decoders, such as (Vinyals and Le, 2015; Sordoni et al., 2015b; Shang et al., 2015). Hierarchical recurrent encoder-decoder (HRED) models (Sordoni et al., 2015a; Serban et al., 2016, 2017) consist of utterance encoder and decoder, and a context RNN which runs over utterance representations to model long-term temporal structure of conversation.

Recently, latent variable models such as VAEs have been adopted in language modeling (Bowman et al., 2016; Zhang et al., 2016; Serban et al., 2017). The VHRED model (Serban et al., 2017) integrates the VAE with the HRED to model Twitter and Ubuntu IRC conversations by introducing an utterance latent variable. This makes a conditional VAE where the generation process is conditioned on the context of conversation. Zhao et al. (2017) further make use of discourse act labels to capture the diversity of conversations.

Degeneracy of Variational Autoencoders. For sequence modeling, VAEs are often merged with the RNN encoder-decoder structure (Bowman et al., 2016; Serban et al., 2017; Zhao et al., 2017) where the encoder predicts the posterior distribution of a latent variable $\mathbf{z}$, and the decoder models the output distributions conditioned on $\mathbf{z}$. However, Bowman et al. (2016) report that a VAE with a RNN decoder easily degenerates; that is, it learns to ignore the latent variable $\mathbf{z}$ and falls back to a vanilla RNN. They propose two techniques to alleviate this issue: *KL annealing* and *word drop*. Chen et al. (2017) interpret this degeneracy in the context of bits-back coding and show that a VAE equipped with autoregressive models such as RNNs often ignores the latent variable to minimize the code length needed for describing data. They propose to constrain the decoder to selectively encode the information of interest in the latent variable. However, their empirical results are limited to an image domain. Zhao et al. (2017) use an auxiliary bag-of-words loss on the latent variable to force the model to use $\mathbf{z}$. That is, they train an auxiliary network that predicts bag-of-words representation of the target utterance based on $\mathbf{z}$. Yet this loss works in an opposite di-

rection to the original objective of VAEs that minimizes the minimum description length. Thus, it may be in danger of forcibly moving the information that is better modeled in the decoder to the latent variable.

3 Approach

We assume that the training set consists of N i.i.d samples of conversations $\{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_N\}$ where each $\mathbf{c}_i$ is a sequence of utterances (*i.e.* sentences) $\{\mathbf{x}_{i1}, \mathbf{x}_{i2}, ..., \mathbf{x}_{in_i}\}$. Our objective is to learn the parameters of a generative network $\boldsymbol{\theta}$ using Maximum Likelihood Estimation (MLE):

$$\arg\max_{\boldsymbol{\theta}} \sum_i \log p_{\boldsymbol{\theta}}(\mathbf{c}_i) \tag{1}$$

We first briefly review the VAE, and explain the degeneracy issue before presenting our model.

3.1 Preliminary: Variational Autoencoder

We follow the notion of Kingma and Welling (2014). A datapoint $\mathbf{x}$ is generated from a latent variable $\mathbf{z}$, which is sampled from some prior distribution $p(\mathbf{z})$, typically a standard Gaussian distribution $\mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I})$. We assume parametric families for conditional distribution $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$. Since it is intractable to compute the log-marginal likelihood $\log p_{\boldsymbol{\theta}}(\mathbf{x})$, we approximate the intractable true posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ with a recognition model $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ to maximize the *variational lower-bound*:

$$\begin{aligned} &\log p_{\boldsymbol{\theta}}(\mathbf{x}) \geq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}) \\ &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[-\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) + \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})] \\ &= -D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})) + \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})] \end{aligned} \tag{2}$$

Eq. 2 is decomposed into two terms: KL divergence term and reconstruction term. Here, KL divergence measures the amount of information encoded in the latent variable $\mathbf{z}$. In the extreme where KL divergence is zero, the model completely ignores $\mathbf{z}$, *i.e.* it degenerates. The expectation term can be stochastically approximated by sampling $\mathbf{z}$ from the variational posterior $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$. The gradients to the recognition model can be efficiently estimated using the *reparameterization* trick (Kingma and Welling, 2014).

3.2 VHRED

Serban et al. (2017) propose Variational Hierarchical Recurrent Encoder Decoder (VHRED) model for conversation modeling. It integrates an utterance latent variable $\mathbf{z}_t^{\text{utt}}$ into the HRED structure (Sordoni et al., 2015a) which consists of three RNN components: *encoder RNN*, *context RNN*, and *decoder RNN*. Given a previous sequence of utterances $\mathbf{x}_1, ...\mathbf{x}_{t-1}$ in a conversation, the VHRED generates the next utterance $\mathbf{x}_t$ as:

$$\mathbf{h}_{t-1}^{\text{enc}} = f_{\boldsymbol{\theta}}^{\text{enc}}(\mathbf{x}_{t-1}) \tag{3}$$

$$\mathbf{h}_t^{\text{cxt}} = f_{\boldsymbol{\theta}}^{\text{cxt}}(\mathbf{h}_{t-1}^{\text{cxt}}, \mathbf{h}_{t-1}^{\text{enc}}) \tag{4}$$

$$p_{\boldsymbol{\theta}}(\mathbf{z}_t^{\text{utt}}|\mathbf{x}_{<t}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t\mathbf{I}) \tag{5}$$

$$\text{where } \boldsymbol{\mu}_t = \text{MLP}_{\boldsymbol{\theta}}(\mathbf{h}_t^{\text{cxt}}) \tag{6}$$

$$\boldsymbol{\sigma}_t = \text{Softplus}(\text{MLP}_{\boldsymbol{\theta}}(\mathbf{h}_t^{\text{cxt}})) \tag{7}$$

$$p_{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{x}_{<t}) = f_{\boldsymbol{\theta}}^{\text{dec}}(\mathbf{x}|\mathbf{h}_t^{\text{cxt}}, \mathbf{z}_t^{\text{utt}}) \tag{8}$$

At time step t, the encoder RNN $f_{\boldsymbol{\theta}}^{\text{enc}}$ takes the previous utterance $\mathbf{x}_{t-1}$ and produces an encoder vector $\mathbf{h}_{t-1}^{\text{enc}}$ (Eq. 3). The context RNN $f_{\boldsymbol{\theta}}^{\text{cxt}}$ models the context of the conversation by updating its hidden states using the encoder vector (Eq. 4). The context $\mathbf{h}_t^{\text{cxt}}$ defines the conditional prior $p_{\boldsymbol{\theta}}(\mathbf{z}_t^{\text{utt}}|\mathbf{x}_{<t})$, which is a factorized Gaussian distribution whose mean $\boldsymbol{\mu}_t$ and diagonal variance $\boldsymbol{\sigma}_t$ are given by feed-forward neural networks (Eq. 5-7). Finally the decoder RNN $f_{\boldsymbol{\theta}}^{\text{dec}}$ generates the utterance $\mathbf{x}_t$, conditioned on the context vector $\mathbf{h}_t^{\text{cxt}}$ and the latent variable $\mathbf{z}_t^{\text{utt}}$ (Eq. 8). We make two important notes: (1) the context RNN can be viewed as a high-level decoder, and together with the decoder RNN, they comprise a hierarchical RNN decoder. (2) VHRED follows a conditional VAE structure where each utterance $\mathbf{x}_t$ is generated conditioned on the context $\mathbf{h}_t^{\text{cxt}}$ (Eq. 5-8).

The variational posterior is a factorized Gaussian distribution where the mean and the diagonal variance are predicted from the target utterance and the context as follows:

$$q_{\boldsymbol{\phi}}(\mathbf{z}_t^{\text{utt}}|\mathbf{x}_{\leq t}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_t', \boldsymbol{\sigma}_t' I) \tag{9}$$

$$\text{where } \boldsymbol{\mu}_t' = \text{MLP}_{\boldsymbol{\phi}}(\mathbf{x}_t, \mathbf{h}_t^{\text{cxt}}) \tag{10}$$

$$\boldsymbol{\sigma}_t' = \text{Softplus}(\text{MLP}_{\boldsymbol{\phi}}(\mathbf{x}_t, \mathbf{h}_t^{\text{cxt}})) \tag{11}$$

3.3 The Degeneration Problem

A known problem of a VAE that incorporates an autoregressive RNN decoder is the degeneracy that ignores the latent variable $\mathbf{z}$. In other words, the KL divergence term in Eq. 2 goes to zero and the decoder fails to learn any dependency between the latent variable and the data. Eventually, the model behaves as a vanilla RNN. This problem is

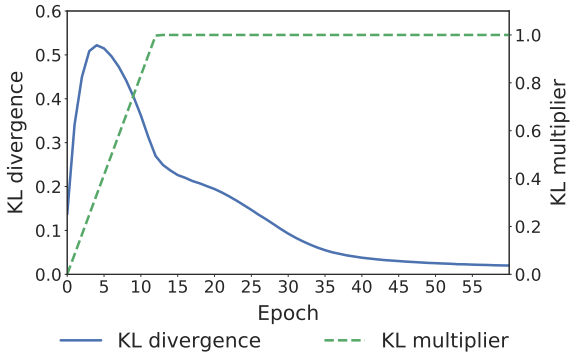


Figure 1: Degeneration of VHRED. The KL divergence term continuously decreases as training proceeds, meaning that the decoder ignores the latent variable $\mathbf{z}^{\text{utt}}$. We train the VHRED on Cornell Movie Dialog Corpus with word drop and KL annealing.

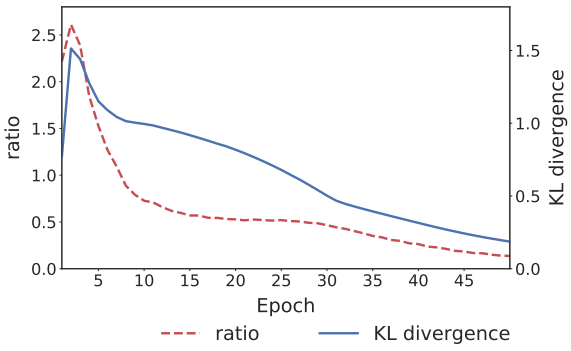


Figure 2: The average ratio $\mathbb{E}[\boldsymbol{\sigma}_t^2]/\text{Var}(\boldsymbol{\mu}_t)$ when the decoder is only conditioned on $\mathbf{z}_t^{\text{utt}}$. The ratio drops to zero as training proceeds, indicating that the conditional priors $p_{\boldsymbol{\theta}}(\mathbf{z}_t^{\text{utt}}|\mathbf{x}_{<t})$ degenerate to separate point masses.

first reported in the sentence VAE (Bowman et al., 2016), in which following two heuristics are proposed to alleviate the problem by weakening the decoder.

First, the *KL annealing* scales the KL divergence term of Eq. 2 using a KL multiplier λ, which gradually increases from 0 to 1 during training:

$$\tilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}) = -\lambda D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})) + \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})] \quad (12)$$

This helps the optimization process to avoid local optima of zero KL divergence in early training. Second, the *word drop* regularization randomly replaces some conditioned-on word tokens in the RNN decoder with the generic unknown word token (UNK) during training. Normally, the RNN decoder predicts each next word in an autoregressive manner, conditioned on the previous sequence of ground truth (GT) words. By randomly replacing a GT word with an UNK token, the word drop regularization weakens the autoregressive power of the decoder and forces it to rely on the latent variable to predict the next word. The word drop probability is normally set to 0.25, since using a higher probability may degrade the model performance (Bowman et al., 2016).

However, we observe that these tricks do not solve the degeneracy for the VHRED in conversation modeling. An example in Fig. 1 shows that the VHRED learns to ignore the utterance latent variable as the KL divergence term falls to zero.

3.4 Empirical Observation on Degeneracy

The decoder RNN of the VHRED in Eq. 8 conditions on two information sources: deterministic $\mathbf{h}_t^{\text{cxt}}$ and stochastic $\mathbf{z}^{\text{utt}}$. In order to check whether the presence of deterministic source $\mathbf{h}_t^{\text{cxt}}$ causes the degeneration, we drop the deterministic $\mathbf{h}_t^{\text{cxt}}$ and condition the decoder only on the stochastic utterance latent variable $\mathbf{z}^{\text{utt}}$:

$$p_{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{x}_{<t}) = f_{\boldsymbol{\theta}}^{\text{dec}}(\mathbf{x}|\mathbf{z}_t^{\text{utt}}) \quad (13)$$

While this model achieves higher values of KL divergence than original VHRED, as training proceeds it again degenerates with the KL divergence term reaching zero (Fig. 2).

To gain an insight of the degeneracy, we examine how the conditional prior $p_{\boldsymbol{\theta}}(\mathbf{z}_t^{\text{utt}}|\mathbf{x}_{<t})$ (Eq. 5) of the utterance latent variable changes during training, using the model above (Eq. 13). Fig. 2 plots the ratios of $\mathbb{E}[\boldsymbol{\sigma}_t^2]/\text{Var}(\boldsymbol{\mu}_t)$, where $\mathbb{E}[\boldsymbol{\sigma}_t^2]$ indicates the *within variance* of the priors, and $\text{Var}(\boldsymbol{\mu}_t)$ is the *between variance* of the priors. Note that traditionally this ratio is closely related to *Analysis of Variance (ANOVA)* (Lomax and Hahs-Vaughn, 2013). The ratio gradually falls to zero, implying that the priors degenerate to separate point masses as training proceeds. Moreover, we find that the degeneracy of priors coincide with the degeneracy of KL divergence, as shown in (Fig. 2). This is intuitively natural: if the prior is already narrow enough to specify the target utterance, there is little pressure to encode any more information in the variational posterior for reconstruction of the target utterance.

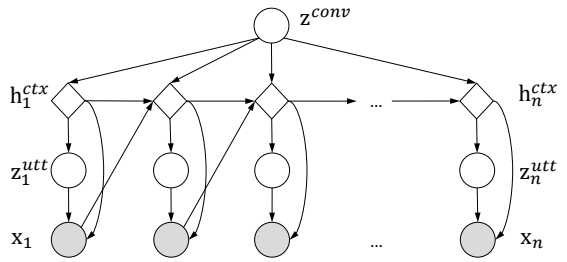


Figure 3: Graphical representation of the Variational Hierarchical Conversation RNN (VHCR). The global latent variable $\mathbf{z}^{\text{conv}}$ provides a global context in which the conversation takes place.

This empirical observation implies that the fundamental reason behind the degeneration may originate from combination of two factors: (1) strong expressive power of the hierarchical RNN decoder and (2) training data sparsity caused by the conditional VAE structure. The VHRED is trained to predict a next target utterance $\mathbf{x}_t$ conditioned on the context $\mathbf{h}_t^{\text{cxt}}$ which encodes information about previous utterances $\{\mathbf{x}_1, \dots, \mathbf{x}_{t-1}\}$. However, conditioning on the context makes the range of training target $\mathbf{x}_t$ very sparse; even in a large-scale conversation corpus such as Ubuntu Dialog (Lowe et al., 2015), there exist one or very few target utterances per context. Therefore, hierarchical RNNs, given their autoregressive power, can easily overfit to training data without using the latent variable. Consequently, the VHRED will not encode any information in the latent variable, i.e. it degenerates. It explains why the word drop fails to prevent the degeneracy in the VHRED. The word drop only regularizes the decoder RNN; however, the context RNN is also powerful enough to predict a next utterance in a given context even with the weakened decoder RNN. Indeed we observe that using a larger word drop probability such as 0.5 or 0.75 only slows down, but fails to stop the KL divergence from vanishing.

3.5 Variational Hierarchical Conversation RNN (VHCR)

As discussed, we argue that the two main causes of degeneration are i) the expressiveness of the hierarchical RNN decoders, and ii) the conditional VAE structure that induces data sparsity. This finding hints us that in order to train a non-degenerate latent variable model, we need to design a model that provides an appropriate way to regularize the hierarchical RNN decoders and alleviate data sparsity per context. At the same time, the model should be capable of modeling complex structure of conversation. Based on these insights, we propose a novel VAE structure named Variational Hierarchical Conversation RNN (VHCR), whose graphical model is illustrated in Fig. 3. Below we first describe the model, and discuss its unique features.

We introduce a global conversation latent variable $\mathbf{z}^{\text{conv}}$ which is responsible for generating a sequence of utterances of a conversation $\mathbf{c} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$:

$$p_{\boldsymbol{\theta}}(\mathbf{c}|\mathbf{z}^{\text{conv}}) = p_{\boldsymbol{\theta}}(\mathbf{x}_1, \dots, \mathbf{x}_n|\mathbf{z}^{\text{conv}}) \tag{14}$$

Overall, the VHCR builds upon the hierarchical RNNs, following the VHRED (Serban et al., 2017). One key update is to form a hierarchical latent structure, by using the global latent variable $\mathbf{z}^{\text{conv}}$ per conversation, along with local the latent variable $\mathbf{z}_t^{\text{utt}}$ injected at each utterance (Fig. 3):

$$\mathbf{h}_t^{\text{enc}} = f_{\boldsymbol{\theta}}^{\text{enc}}(\mathbf{x}_t) \tag{15}$$

$$\mathbf{h}_t^{\text{cxt}} = \begin{cases} \text{MLP}_{\boldsymbol{\theta}}(\mathbf{z}^{\text{conv}}), & \text{if } t = 0 \\ f_{\boldsymbol{\theta}}^{\text{cxt}}(\mathbf{h}_{t-1}^{\text{cxt}}, \mathbf{h}_{t-1}^{\text{enc}}, \mathbf{z}^{\text{conv}}), & \text{otherwise} \end{cases}$$

$$p_{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{x}_{<t}, \mathbf{z}_t^{\text{utt}}, \mathbf{z}^{\text{conv}}) = f_{\boldsymbol{\theta}}^{\text{dec}}(\mathbf{x}|\mathbf{h}_t^{\text{cxt}}, \mathbf{z}_t^{\text{utt}}, \mathbf{z}^{\text{conv}})$$

$$p_{\boldsymbol{\theta}}(\mathbf{z}^{\text{conv}}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I}) \tag{16}$$

$$p_{\boldsymbol{\theta}}(\mathbf{z}_t^{\text{utt}}|\mathbf{x}_{<t}, \mathbf{z}^{\text{conv}}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t\mathbf{I}) \tag{17}$$

$$\text{where } \boldsymbol{\mu}_t = \text{MLP}_{\boldsymbol{\theta}}(\mathbf{h}_t^{\text{cxt}}, \mathbf{z}^{\text{conv}}) \tag{18}$$

$$\boldsymbol{\sigma}_t = \text{Softplus}(\text{MLP}_{\boldsymbol{\theta}}(\mathbf{h}_t^{\text{cxt}}, \mathbf{z}^{\text{conv}})). \tag{19}$$

For inference of $\mathbf{z}^{\text{conv}}$, we use a bi-directional RNN denoted by f^{conv}, which runs over the utterance vectors generated by the encoder RNN:

$$q_{\boldsymbol{\phi}}(\mathbf{z}^{\text{conv}}|\mathbf{x}_1, ..., \mathbf{x}_n) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}^{\text{conv}}, \boldsymbol{\sigma}^{\text{conv}}I) \tag{20}$$

$$\text{where } \mathbf{h}^{\text{conv}} = f^{\text{conv}}(\mathbf{h}_1^{\text{enc}}, ..., \mathbf{h}_n^{\text{enc}}) \tag{21}$$

$$\boldsymbol{\mu}^{\text{conv}} = \text{MLP}_{\boldsymbol{\phi}}(\mathbf{h}^{\text{conv}}) \tag{22}$$

$$\boldsymbol{\sigma}^{\text{conv}} = \text{Softplus}(\text{MLP}_{\boldsymbol{\phi}}(\mathbf{h}^{\text{conv}})). \tag{23}$$

The posteriors for local variables $\mathbf{z}_t^{\text{utt}}$ are then conditioned on $\mathbf{z}^{\text{conv}}$:

$$q_{\boldsymbol{\phi}}(\mathbf{z}_t^{\text{utt}}|\mathbf{x}_1, ..., \mathbf{x}_n, \mathbf{z}^{\text{conv}}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_t', \boldsymbol{\sigma}_t'I) \tag{24}$$

$$\text{where } \boldsymbol{\mu}_t' = \text{MLP}_{\boldsymbol{\phi}}(\mathbf{x}_t, \mathbf{h}_t^{\text{cxt}}, \mathbf{z}^{\text{conv}}) \tag{25}$$

$$\boldsymbol{\sigma}_t' = \text{Softplus}(\text{MLP}_{\boldsymbol{\phi}}(\mathbf{x}_t, \mathbf{h}_t^{\text{cxt}}, \mathbf{z}^{\text{conv}})).$$

Our solution of VHCR to the degeneration problem is based on two ideas. The first idea is to build a hierarchical latent structure of $\mathbf{z}^{\text{conv}}$ for

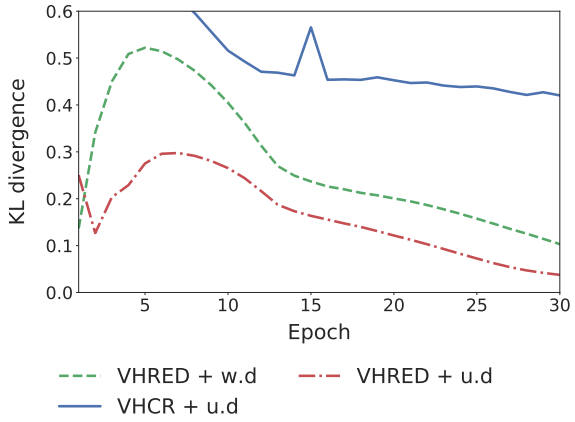


Figure 4: The comparison of KL divergences. The VHCR with the utterance drop shows high and stable KL divergence, indicating the active use of latent variables. w.d and u.d denote the word drop and the utterance drop, respectively.

a conversation and $\mathbf{z}_t^{\text{utt}}$ for each utterance. As $\mathbf{z}^{\text{conv}}$ is independent of the conditional structure, it does not suffer from the data sparsity problem. However, the expressive power of hierarchical RNN decoders makes the model still prone to ignore latent variables $\mathbf{z}^{\text{conv}}$ and $\mathbf{z}_t^{\text{utt}}$. Therefore, our second idea is to apply an *utterance drop* regularization to effectively regularize the hierarchical RNNs, in order to facilitate the use of latent variables. That is, at each time step, the utterance encoder vector $\mathbf{h}_t^{\text{enc}}$ is randomly replaced with a generic unknown vector $\mathbf{h}^{\text{unk}}$ with a probability p. This regularization weakens the autoregressive power of hierarchical RNNs and as well alleviates the data sparsity problem, since it induces noise into the context vector $\mathbf{h}_t^{\text{cxt}}$ which conditions the decoder RNN. The difference with the word drop (Bowman et al., 2016) is that our utterance drop depresses the hierarchical RNN decoders as a whole, while the word drop only weakens the lower-level decoder RNNs. Fig. 4 confirms that with the utterance drop with a probability of 0.25, the VHCR effectively learns to use latent variables, achieving a significant degree of KL divergence.

3.6 Effectiveness of Hierarchical Latent Structure

Is the hierarchical latent structure of the VHCR crucial for effective utilization of latent variables? We investigate this question by applying the utterance drop on the VHRED which lacks any hierarchical latent structure. We observe that the KL divergence still vanishes (Fig. 4), even though the utterance drop injects considerable noise in the context $\mathbf{h}_t^{\text{cxt}}$. We argue that the utterance drop weakens the context RNN, thus it consequently fail to predict a reasonable prior distribution for $\mathbf{z}^{\text{utt}}$ (Eq. 5-7). If the prior is far away from the region of $\mathbf{z}^{\text{utt}}$ that can generate a correct target utterance, encoding information about the target in the variational posterior will incur a large KL divergence penalty. If the penalty outweighs the gain of the reconstruction term in Eq. 2, then the model would learn to ignore $\mathbf{z}^{\text{utt}}$, in order to maximize the variational lower-bound in Eq. 2.

On the other hand, the global variable $\mathbf{z}^{\text{conv}}$ allows the VHCR to predict a reasonable prior for local variable $\mathbf{z}_t^{\text{utt}}$ even in the presence of the utterance drop regularization. That is, $\mathbf{z}^{\text{conv}}$ can act as a *guide* for $\mathbf{z}^{\text{utt}}$ by encoding the information for local variables. This reduces the KL divergence penalty induced by encoding information in $\mathbf{z}^{\text{utt}}$ to an affordable degree at the cost of KL divergence caused by using $\mathbf{z}^{\text{conv}}$. This trade-off is indeed a fundamental strength of hierarchical models that provide *parsimonious* representation; if there exists any shared information among the local variables, it is coded in the global latent variable reducing the code length by effectively reusing the information. The remaining local variability is handled properly by the decoding distribution and local latent variables.

The global variable $\mathbf{z}^{\text{conv}}$ provides other benefits by representing a latent global structure of a conversation, such as a topic, a length, and a tone of the conversation. Moreover, it allows us to control such global properties, which is impossible for models without hierarchical latent structure.

4 Results

We first describe our experimental setting, such as datasets and baselines (section 4.1). We then report quantitative comparisons using three different metrics (section 4.2–4.4). Finally, we present qualitative analyses, including several utterance control tasks that are enabled by the hierarchal latent structure of our VHCR (section 4.5). We defer implementation details and additional experiment results to the supplementary file.

4.1 Experimental Setting

Datasets. We evaluate the performance of conversation generation using two benchmark datasets: 1) Cornell Movie Dialog Corpus (Danescu-

Model	NLL	Recon.	KL div.
HRED	3.873	-	-
VHRED	$\leq$ 3.912	3.619	0.293
VHRED + w.d	$\leq$ 3.904	3.553	0.351
VHRED + bow	$\leq$ 4.149	2.982	1.167
VHCR + u.d	$\leq$ 4.026	3.523	0.503

(a) Cornell Movie Dialog

Model	NLL	Recon.	KL div.
HRED	3.766	-	-
VHRED	$\leq$ 3.767	3.654	0.113
VHRED + w.d	$\leq$ 3.824	3.363	0.461
VHRED + bow	$\leq$ 4.237	2.215	2.022
VHCR + u.d	$\leq$ 3.951	3.205	0.756

(b) Ubuntu Dialog

Table 1: Results of Negative Log-likelihood. The inequalities denote the variational bounds. w.d and u.d., and bow denote the word drop, the utterance drop, and the auxiliary bag-of-words loss respectively.

	Cornell			Ubuntu		
Model	Total	$\mathbf{z}^{\text{conv}}$	$\mathbf{z}^{\text{utt}}$	Total	$\mathbf{z}^{\text{conv}}$	$\mathbf{z}^{\text{utt}}$
VHRED	0.351	-	0.351	0.461	-	0.461
VHCR	0.503	0.189	0.314	0.756	0.198	0.558

Table 2: KL divergence decomposition. VHRED and VHCR are trained with word drop and utterance drop respectively.

Niculescu-Mizil and Lee, 2011), containing 220,579 conversations from 617 movies. 2) Ubuntu Dialog Corpus (Lowe et al., 2015), containing about 1 million multi-turn conversations from Ubuntu IRC channels. In both datasets, we truncate utterances longer than 30 words.

Baselines. We compare our approach with four baselines. They are combinations of two state-of-the-art models of conversation generation with different solutions to the degeneracy. (i) Hierarchical recurrent encoder-decoder (HRED) (Serban et al., 2016), (ii) Variational HRED (VHRED) (Serban et al., 2017), (iii) VHRED with the word drop (Bowman et al., 2016), and (iv) VHRED with the bag-of-words (bow) loss (Zhao et al., 2017).

Performance Measures. Automatic evaluation of conversational systems is still a challenging problem (Liu et al., 2016). Based on literature, we report three quantitative metrics: i) the negative log-likelihood (the variational bound for variational models), ii) embedding-based metrics (Serban et al., 2017), and iii) human evaluation via Amazon Mechanical Turk (AMT).

4.2 Results of Negative Log-likelihood

Table 1 summarizes the per-word negative log-likelihood (NLL) evaluated on the test sets of two datasets. For variational models, we instead present the variational bound of the negative log-likelihood in Eq. 2, which consists of the reconstruction error term and the KL divergence term. The KL divergence term can measure how much each model utilizes the latent variables.

We observe that the NLL is the lowest by the HRED. Variational models show higher NLLs, because they are regularized methods that are forced to rely more on latent variables. Independent of NLL values, we later show that the latent variable models often show better generalization performance in terms of embedding-based metrics and human evaluation. In the VHRED, the KL divergence term gradually vanishes even with the word drop regularization; thus, early stopping is necessary to obtain a meaningful KL divergence. The VHRED with the bag-of-words loss (bow) achieves the highest KL divergence, however, at the cost of high NLL values. That is, the variational lower-bound minimizes the minimum description length, to which the bow loss works in an opposite direction by forcing latent variables to encode bag-of-words representation of utterances. Our VHCR achieves stable KL divergence without any auxiliary objective, and the NLL is lower than the VHRED + bow model.

Table 2 summarizes how global and latent variable are used in the VHCR. We observe that VHCR encodes a significant amount of information in the global variable $\mathbf{z}^{\text{conv}}$ as well as in the local variable $\mathbf{z}^{\text{utt}}$, indicating that the VHCR successfully exploits its hierarchical latent structure.

4.3 Results of Embedding-Based Metrics

The embedding-based metrics (Serban et al., 2017; Rus and Lintean, 2012) measure the textual similarity between the words in the model response and the ground truth. We represent words using Word2Vec embeddings trained on the Google News Corpus[1]. The *average* metric projects each utterance to a vector by taking the mean over word embeddings in the utterance, and computes the cosine similarity between the model response vector and the ground truth vector. The *extrema* metric is similar to the average metric, only except that it takes the extremum of each di-

[1]https://code.google.com/archive/p/word2vec/.

Model	Average	Extrema	Greedy
1-turn			
HRED	0.541	0.370	0.387
VHRED	0.543	0.356	0.393
VHRED + w.d	0.554	0.365	0.404
VHRED + bow	0.555	0.350	0.411
VHCR + u.d	**0.585**	**0.376**	**0.434**
3-turn			
HRED	0.556	0.372	0.395
VHRED	0.554	0.360	0.398
VHRED + w.d	0.566	0.369	0.408
VHRED + bow	0.573	0.360	0.423
VHCR + u.d	**0.588**	**0.378**	**0.429**

(a) Cornell Movie Dialog

Model	Average	Extrema	Greedy
1-turn			
HRED	0.567	**0.337**	0.412
VHRED	0.547	0.322	0.398
VHRED + w.d	0.545	0.314	0.398
VHRED + bow	0.545	0.306	0.398
VHCR + u.d	**0.570**	0.312	**0.425**
3-turn			
HRED	0.559	**0.324**	0.402
VHRED	0.551	0.315	0.397
VHRED + w.d	0.551	0.309	0.399
VHRED + bow	0.552	0.303	0.398
VHCR + u.d	**0.574**	0.311	**0.422**

(b) Ubuntu Dialog

Table 3: Results of embedding-based metrics. 1-turn and 3-turn responses of models per context.

mension, instead of the mean. The *greedy* metric first finds the best non-exclusive word alignment between the model response and the ground truth, and then computes the mean over the cosine similarity between the aligned words.

Table 3 compares the different methods with three embedding-based metrics. Each model generates a single response (1-turn) or consecutive three responses (3-turn) for a given context. For 3-turn cases, we report the average of metrics measured for three turns. We use the greedy decoding for all the models.

Our VHCR achieves the best results in most metrics. The HRED is the worst on the Cornell Movie dataset, but outperforms the VHRED and VHRED + bow on the Ubuntu Dialog dataset. Although the VHRED + bow shows the highest KL divergence, its performance is similar to that of VHRED, and worse than that of the VHCR model. It suggests that a higher KL divergence does not necessarily lead to better performance; it is more important for the models to balance the modeling powers of the decoder and the latent variables. The VHCR uses a more sophisticated hierarchical latent structure, which better reflects the structure of natural language conversations.

4.4 Results of Human Evaluation

Table 4 reports human evaluation results via Amazon Mechanical Turk (AMT). The VHCR outperforms the baselines in both datasets; yet the performance improvement in Cornell Movie Dialog are less significant compared to that of Ubuntu. We empirically find that Cornell Movie dataset is small in size, but very diverse and complex in content and style, and the models often fail to generate sensible responses for the context. The performance gap with the HRED is the smallest, suggesting that the VAE models without hierarchical latent structure have overfitted to Cornell Movie dataset.

4.5 Qualitative Analyses

Comparison of Predicted Responses. Table 5 compares the generated responses of algorithms. Overall, the VHCR creates more consistent responses within the context of a given conversation. This is supposedly due to the global latent variable $\mathbf{z}^{\text{conv}}$ that provides a more direct and effective way to handle the global context of a conversation. The context RNN of the baseline models can handle long-term context to some extent, but not as much as the VHCR.

Interpolation on $\mathbf{z}^{\text{conv}}$. We present examples of one advantage by the hierarchical latent structure of the VHCR, which cannot be done by the other existing models. Table 6 shows how the generated responses vary according to the interpolation on $\mathbf{z}^{\text{conv}}$. We randomly sample two $\mathbf{z}^{\text{conv}}$ from a standard Gaussian prior as references (*i.e.* the top and the bottom row of Table 6), and interpolate points between them. We generate 3-turn conversations conditioned on given $\mathbf{z}^{\text{conv}}$. We see that $\mathbf{z}^{\text{conv}}$ controls the overall tone and content of conversations; for example, the tone of the response is friendly in the first sample, but gradually becomes hostile as $\mathbf{z}^{\text{conv}}$ changes.

Generation on a Fixed $\mathbf{z}^{\text{conv}}$. We also study how fixing a global conversation latent variable $\mathbf{z}^{\text{conv}}$ affects the conversation generation. Table 7 shows an example, where we randomly fix a reference $\mathbf{z}^{\text{conv}}$ from the prior, and generate multiple examples of 3-turn conversation using randomly sampled local variables $\mathbf{z}^{\text{utt}}$. We observe that $\mathbf{z}^{\text{conv}}$ heavily affects the form of the first utterance; in the examples, the first utterances all start with a "where" phrase. At the same time, responses show

	Cornell			Ubuntu		
Opponent	Wins	Losses	Ties	Wins	Losses	Ties
VHCR vs HRED	$\mathbf{28.5 \pm 1.9}$	28.2 ± 1.9	43.3 ± 2.1	$\mathbf{52.9 \pm 2.1}$	42.2 ± 2.1	4.9 ± 0.9
VHCR vs VHRED + w.d	$\mathbf{29.9 \pm 1.9}$	28.0 ± 1.9	42.1 ± 2.1	$\mathbf{48.1 \pm 2.1}$	40.1 ± 3.6	11.9 ± 1.4
VHCR vs VHRED + bow	$\mathbf{31.3 \pm 2.0}$	26.9 ± 1.9	41.7 ± 2.1	$\mathbf{46.1 \pm 2.1}$	39.9 ± 2.1	14.0 ± 1.5

Table 4: Results of human evaluation via AMT. Human turkers are asked to choose which response is more appropriate in a given context, without knowing which algorithms generate which responses. For each pair of models, we carry out three evaluation batches, each of which consists of 100 random test samples evaluated by five unique humans. We report mean preferences with $\pm 90\%$ confidence interval.

Context	HRED	VHRED	VHRED+bow	VHCR
i'm going home. → what's wrong? → i don't know. one of these days, i'm gon na see louis. my back's killing me.	i don't know.	i don't want to be here. i'm going to be here.	are you going to go now?	this isn't a joke. you're going to be here.
how's sleeping beauty? → still sleeping. → kiss him, maybe he'll turn into a frog.	he's a good man.	what?	we won't?	maybe he's dead.
when i mount my usb drive i get a permission denied but i can easily see the files if i boot via live cd any ideas on how i → the file permissions of the ext3 file system are likely preventing you → thanks, any idea on options to fix?	no, i don't know how to fix it, but i'm not sure how to fix it	what kind of filesystem is it on?	you did it just first? then you'll need to reboot and then remount it up again, if your problem persists, i'd give you the	the only thing i can think of is to fix the permissions issue, try mounting the drive with the uid and gid options in the forum
hello i need some support, anybody there? → go ahead with your questions. <unk> are not expected on irc. → i have an hp pavilion <unk> here and i can't get it to install ubuntu	what's the problem?	<unk>. what's the problem?	i'm sorry.	they should be supported by canonical. you don't need to use them.

Table 5: Qualitative comparison of generated responses. Top two rows show the samples from Cornell Movie Dialog, while the bottom two rows are from Ubuntu Dialog.

nice to meet you. **→ where's my wife?** **→ she's asleep.**
go on, frank. → i ll tell you what i'm doing. i'm a little tired. → why?
now, you're not going. → not for a long time. → but you're a little <unk>.
get out of here. → all right. → you want to go home?
get out of here!!! **→ it's not your fault, is n't it?** **→ why? what's wrong?**

Table 6: An example of interpolated 3-turn responses over $\mathbf{z}^{\text{conv}}$ on Cornell Movie Dialog.

where is she? → she's the only one who knows where she is, she's going to be all right. → oh, you're the only one who's gon na be. she's a <unk>.
where's my wife? → you've got to get out of here, you know? you're the one who's gon na be here. → oh, that's nice.
where are you? → well, i was just thinking about you and i know what you're doing. i'm going to have to go to the <unk> and i'm → i'm sorry.
where are you going? → to get you to the airport. → you're going to be late?
where are you going? → to the <unk>. i am not going to tell you what i am. i am the only one who has to be. i will be the → you've got to stop!

Table 7: An example of 3-turn responses conditioned on sampled $\mathbf{z}^{\text{utt}}$ for a single fixed $\mathbf{z}^{\text{conv}}$.

variations according to different local variables $\mathbf{z}^{\text{utt}}$. These examples show that the hierarchical latent structure of VHCR allows both global and fine-grained control over generated conversations.

5 Discussion

We introduced the variational hierarchical conversation RNN (VHCR) for conversation modeling. We noted that the degeneration problem in existing VAE models such as the VHRED is persistent, and proposed a hierarchical latent variable model with the utterance drop regularization. Our VHCR obtained higher and more stable KL divergences than various versions of VHRED models without using any auxiliary objective. The empirical results showed that the VHCR better reflected the structure of natural conversations, and outperformed previous models. Moreover, the hierarchical latent structure allowed both global and fine-grained control over the conversation generation.

Acknowledgments

This work was supported by Kakao and Kakao Brain corporations, and Creative-Pioneering Researchers Program through Seoul National University. Gunhee Kim is the corresponding author.

References

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *CoNLL*. https://doi.org/10.18653/v1/K16-1002.

Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2017. Variational lossy autoencoder. In *ICLR*.

Cristian Danescu-Niculescu-Mizil and Lillian Lee. 2011. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *CMCL Workshop*.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *ICLR*.

Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547* .

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *EMNLP*.

Richard G Lomax and Debbie L Hahs-Vaughn. 2013. *Statistical concepts: A second course*. Routledge.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *SIGDIAL*.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*.

Vasile Rus and Mihai Lintean. 2012. A comparison of greedy and optimal assessment of natural language student input using word-to-word similarity metrics. In *Building Educational Applications Using NLP Workshop*. ACL. http://www.aclweb.org/anthology/W12-2018.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *ACL*. https://doi.org/10.3115/v1/P15-1152.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015a. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *CIKM*.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015b. A neural network approach to context-sensitive generation of conversational responses. In *NAACL-HLT*. https://doi.org/10.3115/v1/N15-1020.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. In *ICML Deep Learning Workshop*.

Biao Zhang, Deyi Xiong, Jinsong Su, Hong Duan, and Min Zhang. 2016. Variational neural machine translation. In *EMNLP*. https://doi.org/10.18653/v1/D16-1050.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *ACL*. https://doi.org/10.18653/v1/P17-1061.

Detecting Egregious Conversations between Customers and Virtual Agents

Tommy Sandbank,
Michal Shmueli-Scheuer,
Jonathan Herzig, David Konopnicki
IBM Research
Haifa 31905, Israel
{tommy,shmueli,hjon,davidko}@il.ibm.com

John Richards,
David Piorkowski
IBM Research
Yorktown Heights, NY USA
ajtr@us.ibm.com,
david.piorkowski@ibm.com

Abstract

Virtual agents are becoming a prominent channel of interaction in customer service. Not all customer interactions are smooth, however, and some can become almost comically bad. In such instances, a human agent might need to step in and salvage the conversation. Detecting bad conversations is important since disappointing customer service may threaten customer loyalty and impact revenue. In this paper, we outline an approach to detecting such *egregious* conversations, using behavioral cues from the user, patterns in agent responses, and user-agent interaction. Using logs of two commercial systems, we show that using these features improves the detection F1-score by around 20% over using textual features alone. In addition, we show that those features are common across two quite different domains and, arguably, universal.

1 Introduction

Automated conversational agents (chatbots) are becoming widely used for various tasks such as personal assistants or as customer service agents. Recent studies project that 80% of businesses plan to use chatbots by 2020[1], and that chatbots will power 85% of customer service interactions by the year 2020[2]. This increasing usage is mainly due to advances in artificial intelligence and natural language processing (Hirschberg and Manning, 2015) along with increasingly capable chat development environments, leading to improvements in conversational richness and robustness.

Still, chatbots may behave extremely badly, leading to conversations so off-the-mark that only a human agent could step in and salvage them. Consequences of these failures may include loss of customer goodwill and associated revenue, and even exposure to litigation if the failures can be shown to include fraudulent claims. Due to the increasing prevalence of chatbots, even a small fraction of such egregious[3] conversations could be problematic for the companies deploying chatbots and the providers of chatbot services.

In this paper we study detecting these egregious conversations that can arise in numerous ways. For example, incomplete or internally inconsistent training data can lead to false classification of user intent. Bugs in dialog descriptions can lead to dead ends. Failure to maintain adequate context can cause chatbots to miss anaphoric references. In the extreme case, malicious actors may provide heavily biased (e.g., the Tay chatbot[4]) or even hacked misbehaviors.

In this article, we focus on customer care systems. In such setting, a conversation usually becomes egregious due to a combination of the aforementioned problems. The resulting customer frustration may not surface in easily detectable ways such as the appearance of all caps, shouting to a speech recognizer, or the use of profanity or extreme punctuation. Consequently, the chatbot will continue as if the conversation is proceeding well, usually

[1] http://read.bi/2gU0szG
[2] http://gtnr.it/2z428RS
[3] Defined by the dictionary as outstandingly bad.
[4] http://bit.ly/2fwYaa5

Proceedings of NAACL-HLT 2018, pages 1802–1811
New Orleans, Louisiana, June 1 - 6, 2018.

Customer: *I got 2 quotes for the flight ticket, but i'm wondering what the details of each ticket are?*
Agent: *Please select "Buy" next to the ticket you'd like to purchase.*
Customer: *No, I don't want to buy yet till I know the details of the flights.*
Agent: *If you're in the process of renting a car, please continue with by clicking "Next"*
Customer: *Are you a real person?*
Agent: *I am a digital assistant. I've been trained to answer questions about travels. Ask me any questions you have.*
Customer: *I asked a specific question and you gave me a random answer about car rental which I'm not interested in.*
Agent: *I'm not trained on that yet, but I'm still learning. You may want to rephrase your question and try again.*
Customer: *This service is pointless , can i talk to a real live person?*
Agent: *We don't currently have live agents to chat with online.*

Figure 1: Example of an *egregious* conversation.

leading to conversational breakdown.

Consider, for example, the anonymized but representative conversation depicted in Figure 1. Here the customer aims to understand the details of a flight ticket. In the first two turns, the chatbot misses the customer's intentions, which leads to the customer asking "Are you a real person?". The customer then tries to explain what went wrong, but the chatbot has insufficient exposure to this sort of utterance to provide anything but the default response ("I'm not trained on that"). The response seems to upset the customer and leads to a request for a human agent, which is rejected by the system ("We don't currently have live agents"). Such rejection along with the previous responses could lead to customer frustration (Amsel, 1992).

Being able to automatically detect such conversations, either in real time or through log analysis, could help to improve chatbot quality. If detected in real time, a human agent can be pulled in to salvage the conversation. As an aid to chatbot improvement, analysis of egregious conversations can often point to problems in training data or system logic that can be repaired. While it is possible to scan system logs by eye, the sheer volume of conversations may overwhelm the analyst or lead to random sampling that misses important failures. If, though, we can automatically detect the worst conversations (in our experience, typically under 10% of the total), the focus can be on fixing the worst problems.

Our goal in this paper is to study conversational features that lead to egregious conversations. Specifically, we consider customer inputs throughout a whole conversation, and detect cues such as rephrasing, the presence of heightened emotions, and queries about whether the chatbot is a human or requests to speak to an actual human. In addition, we analyze the chatbot responses, looking for repetitions (e.g. from loops that might be due to flow problems), and the presence of "not trained" responses. Finally, we analyze the larger conversational context exploring, for example, where the presence of a "not trained" response might be especially problematic (e.g., in the presence of strong customer emotion).

The main contributions of this paper are twofold: (1) This is the first research focusing on detecting egregious conversations in conversational agent (chatbot) setting and (2) this is the first research using unique agent, customer, and customer-agent interaction features to detect egregiousness.

The rest of this paper is organized as follows. We review related work, then we formally define the methodology for detecting egregious conversations. We describe our data, experimental setting, and results. We then conclude and suggest future directions.

2 Related Work

Detecting egregious conversations is a new task, however, there is related work that aim at measuring the general quality of the interactions in conversational systems. These works studied the complementary problem of detecting and measuring user satisfaction and engagement. Early work by (Walker et al., 1997, 2001) discussed a framework that maximizes the user satisfaction by considering measures such as number of inappropriate utterances, recognition rates, number of times user requests repetitions, number of turns per interaction, etc. Shortcomings of this approach are discussed by (Hajdinjak and Mihelic, 2006). Other works focus on predicting the user engagement in such systems. Examples include (Kiseleva et al., 2016b,a; Jiang et al., 2015). Specifically, these

works evaluated chat functionality by asking users to make conversations with an intelligent agent and measured the user satisfaction along with other features such as the automatic speech recognition (ASR) quality and intent classification quality. In (Sandbank et al., 2017) the authors presented a conversational system enhanced with emotion analysis, and suggested using emotions as triggers for human escalation. In our work, we likewise use emotion analysis as predictive features for egregious conversation. The works of (Sarikaya, 2017; Sano et al., 2017) studied reasons why users reformulated utterances in such systems. Specifically, in (Sarikaya, 2017) they reported on how the different reasons affect the users' satisfaction. In (Sano et al., 2017) they focused on how to automatically predict the reason for user's dissatisfaction using different features. Our work also explores user reformulation (or rephrasing) as one of the features to predict egregious conversations. We build on the previous work by leveraging some of the approaches in our classifier for egregious conversations. In (Walker et al., 2000; Hastie et al., 2002) the authors also looked for problems in a specific setting of spoken conversations. The main difference with our work is that we focus on chat logs for domains for which the expected user utterances are a bit more diverse, using interaction features as well as features that are not sensitive to any architectural aspects of the conversational system (e.g., ASR component). Several other approaches for evaluating chatbot conversations indirectly capture the notion of conversational quality. For example, several prior works borrowed from the field of pragmatics in various metrics around the principles of cooperative conversation (Chakrabarti and Luger, 2013; Saygin A. P., 2002). In (Steidl et al., 2004) they measured dialogue success at the turn level as a way of predicting the success of a conversation as a whole. (Webb et al., 2010) created a measure of dialogue appropriateness to determine its role in maintaining a conversation. Recently, (Liu et al., 2016) evaluated a number of popular measures for dialogue response generation systems and highlighted specific weaknesses in the measures. Similarly, in (Sebastian et al., 2009) they developed a taxonomy of available measures for an end-user's quality of experience for multimodel dialogue systems, some of which touch on conversational quality. All these measures may serve as reasons for a conversation turning egregious, but none try to capture or predict it directly.

In the domain of customer service, researchers mainly studied reasons for failure of such systems along with suggestions for improved design (Mimoun et al., 2012; Gnewuch et al., 2017). In (Mimoun et al., 2012) the authors analyzed reasons sales chatbots fail by interviewing chatbots experts. They found that a combination of exaggerated customer expectations along with a reduction in agent performance (e.g., failure to listen to the consumer, being too intrusive) caused customers to stop using such systems. Based on this qualitative study, they proposed an improved model for sales chatbots. In (Gnewuch et al., 2017) they studied service quality dimensions (i.e., reliability, empathy, responsiveness, and tangibility) and how to apply them during agent design. The main difference between those works and ours is that they focus on qualitative high-level analysis while we focus on automatic detection based on the conversations logs.

3 Methodology

The objective of this work is to reliably detect egregious conversations between a human and a virtual agent. We treat this as a binary classification task, where the target classes are "egregious" and "non-egregious". While we are currently applying this to complete conversations (i.e., the classification is done on the whole conversation), some of the features examined here could likely be used to detect egregious conversations as they were unfolding in real time. To perform egregious conversation detection, features from both customer inputs and agent responses are extracted, together with features related to the combination of specific inputs and responses. In addition, some of these features are *contextual*, meaning that they are dependent on *where* in the conversation they appear.

Using this set of features for detecting egre-

gious conversations is novel, and as our experimental results show, improves performance compared to a model based solely on features extracted from the conversation's text. We now describe the agent, customer, and combined customer-agent features.

3.1 Agent Response Features

A virtual agent is generally expected to closely simulate interactions with a human operator (Reeves and Nass, 1996; Nass and Moon,Y, 2000; Krämer, 2008). When the agent starts losing the context of a conversation, fails in understanding the customer intention, or keeps repeating the same responses, the illusion of conversing with a human is lost and the conversation may become extremely annoying. With this in mind, we now describe the analysis of the agent's responses and associated features (summarized in the top part of Table 1).

3.1.1 Repeating Response Analysis

As typically implemented, the virtual agent's task is to reliably detect the intent of each customer's utterance and respond meaningfully. Accurate intent detection is thus a fundamental characteristic of well-trained virtual agents, and incorrect intent analysis is reported as the leading cause of user dissatisfaction (Sarikaya, 2017). Moreover, since a classifier (e.g., SVM, neural network, etc.) is often used to detect intents, its probabilistic behavior can cause the agent to repeat the same (or semantically similar) response over and over again, despite the user's attempt to rephrase the same intent.

Such agent repetitions lead to an unnatural interaction (Klüwer, 2011). To identify the agent's repeating responses, we measured similarity between agent's subsequent (not necessarily sequential) turns. We represented each sentence by averaging the pre-trained embeddings[5] of each word in the sentence, calculating the cosine similarity between the representations. Turns with a high similarity value[6] are considered as repeating responses.

[5]https://code.google.com/archive/p/word2vec

[6]Empirically, similarity values ≥ 0.8

3.1.2 Unsupported Intent Analysis

Given that the knowledge of a virtual agent is necessarily limited, we can expect that training would not cover all customer intents. If the classifier technology provides an estimate of classification confidence, the agent can respond with some variant of "I'm not trained on that" when confidence is low. In some cases, customers will accept that not all requests are supported. In other cases, unsupported intents can lead to customer dissatisfaction (Sarikaya, 2017), and cascade to an egregious conversation (as discussed below in Section 3.3). We extracted the possible variants of the unsupported intent messages directly from the system, and later matched them with the agent responses from the logs.

3.2 Customer Inputs Features

From the customer's point of view, an ineffective interaction with a virtual agent is clearly undesirable. An ineffective interaction requires the expenditure of relatively large effort from the customer with little return on the investment (Zeithaml et al., 1990; Mimoun et al., 2012). These efforts can appear as behavioral cues in the customer's inputs, and include emotions, repetitions, and more. We used the following customer analysis in our model. Customer features are summarized in the middle part of Table 1.

3.2.1 Rephrasing Analysis

When a customer repeats or rephrases an utterance, it usually indicates a problem with the agent's understanding of the customer's intent. This can be caused by different reasons as described in (Sano et al., 2017). To measure the similarity between subsequent customer turns to detect repetition or rephrasing, we used the same approach as described in Section 3.1.1. Turns with a high similarity value[6] are considered as rephrases.

3.2.2 Emotional Analysis

The customer's emotional state during the conversation is known to correlate with the conversation's quality (Oliver, 2014). In order to analyze the emotions that customers exhibit in each turn, we utilized the IBM Tone Analyzer service, available publicly online[7].

[7]https://ibm.co/2hnYkCv

This service was trained using customer care interactions, and infers emotions such as *frustration*, *sadness*, *happiness*. We focused on negative emotions (denoted as NEG EMO) to identify turns with a negative emotional peak (i.e., single utterances that carried high negative emotional state), as well as to estimate the aggregated negative emotion throughout the conversation (i.e., the averaged negative emotion intensity). In order to get a more robust representation of the customer's negative emotional state, we summed the score of the negative emotions (such as *frustration*, *sadness*, *anger*, etc.) into a single negative sentiment score (denoted as NEG SENT). Note that we used the positive emotions as a filter for other customer features, such as the rephrasing analysis. Usually, high positive emotions capture different styles of "thanking the agent", or indicate that the customer is somewhat satisfied (Rychalski and Hudson, 2017), thus, the conversation is less likely to become egregious.

3.2.3 Asking for a Human Agent

In examining the conversation logs, we noticed that it is not unusual to find a customer asking to be transferred to a human agent. Such a request might indicate that the virtual agent is not providing a satisfactory service. Moreover, even if there are human agents, they might not be available at all times, and thus, a rejection of such a request is sometimes reasonable, but might still lead to customer frustration (Amsel, 1992).

3.2.4 Unigram Input

In addition to the above analyses, we also detected customer turns that contain exactly one word. The assumption is that single word (unigram) sentences are probably short customer responses (e.g., no, yes, thanks, okay), which in most cases do not contribute to the egregiousness of the conversation. Hence, calculating the percentage of those turns out of the whole conversation gives us another measurable feature.

3.3 Customer-Agent Interaction Features

We also looked at features across conversation utterance-response pairs in order to capture a more complete picture of the interaction between the customer and the virtual agent. Here, we considered a pair to be customer utterance followed by an agent response. For example, a pair may contain a turn in which the customer expressed negative emotions and received a response of "not trained" by the agent. In this case, we would leverage the two analyses: emotional and unsupported intent. Figure 1 gives an example of this in the customer's penultimate turn. Such interactions may divert the conversation towards becoming egregious. These features are summarized in the last part of Table 1.

Group	Feature	Description	Contextual?
Agent	AGNT RPT	Similarity of subsequent agent responses	Yes
	#AGNT !TRND	Number of times the agent replied with "not trained"	No
Customer	MAX 3 RPHRS	Max rephrasing similarity score of 3 subsequent turns	Yes
	#RPHRS	Number of customer rephrasing throughout the conversation	Yes
	MAX NEG EMO	Max negative emotion in the conversation	No
	NEG SENT	Aggregated negative sentiment in the conversation	No
	DIFF NEG SENT	Difference between max turn-level negative sentiment and conversation-level	Yes + No
	RPHRS & NEG SENT	Rephrasing of subsequent turns with an average high negative sentiment	Yes
	HMN AGT & NEG SENT	Negative sentiment when asking for a human agent	No
	#1 WRD	Turns that contained only one word	Yes
Customer-Agent Interaction	NEG SENT & AGNT !TRND	Customer negative sentiment with agent replying "not trained"	No
	HMN AGT & AGNT !TRND	Customer asking to talk to a human agent followed by the agent replying "not trained"	No
	LNG SNTNS & AGNT !TRND	Customer long turn followed by an agent "not trained" response	No
	RPHRS & SMLR	The similarity between the customer's turn and the agent's response in case of customer rephrasing	No
	RPHRS & AGNT !TRND	The similarity between the customer's turns when the agent's response is "not trained"	No
	CONV LEN	Total number of customer turns and agent responses	No

Table 1: Features sets description.

3.3.1 Similarity Analysis

We also calculated the similarity between the customer's turn and the virtual agent's response in cases of customer rephrasing. This analysis aims to capture the reason for the customer rephrasing. When a similarity score between the customer's turn and the agent's response is low, this may indicate a misclassified intent, as the agent's responses are likely to share some textual similarity to the customer's utterance. Thus, a low score may indicate a poor interaction, which might lead the conversation to become egregious. Another similarity feature is between two customer's subsequent turns when the agent's response was "not trained".

3.4 Conversation Egregiousness Prediction Classifier

We trained a binary SVM classifier with a linear kernel. A feature vector for a sample in the training data is generated using the scores calculated for the described features, where each feature value is a number between [0,1]. After the model was trained, test conversations are classified by the model, after being transformed to a feature vector in the same way a training sample is transformed. The SVM classification model (denoted *EGR*) outputs a label "egregious" or "non-egregious" as a prediction for the conversation.

4 Experiments

4.1 Dataset

We extracted data from two commercial systems that provide customer support via conversational bots (hereafter denoted as *company A* and *company B*). Both agents are using similar underlying conversation engines, each embedded in a larger system with its own unique business logic. *Company A*'s system deals with sales support during an online purchase, while *company B*'s system deals with technical support for purchased software products. Each system logs conversations, and each conversation is a sequence of tuples, where each tuple consists of {*conversation id, turn id, customer input, agent response*}. From each system, we randomly extracted 10000 conversations. We further removed conversations that contained fewer than 2 turns, as these are too short to be meaningful since the customer never replied or provided more details about the issue at hand. Figure 2 depicts the frequencies of conversation lengths which follow a power-law relationship. The conversations from *company A*'s system tend to be longer, with an average of 8.4 turns vs. an average of 4.4 turns for *company B*.

4.2 Experimental Setup

The first step in building a classification model is to obtain ground truth data. For this purpose, we randomly sampled conversations from our datasets. This sample included 1100 and 200 conversations for *company A* and *company B* respectively. The sampled conversations were tagged using an in-house tagging system designed to increase the consistency of human judgements. Each conversation was tagged by four different expert judges[8]. Given the full conversation, each judge tagged whether the conversation was egregious or not following this guideline: "Conversations which are extraordinarily bad in some way, those conversations where you'd like to see a human jump in and *save* the conversation".

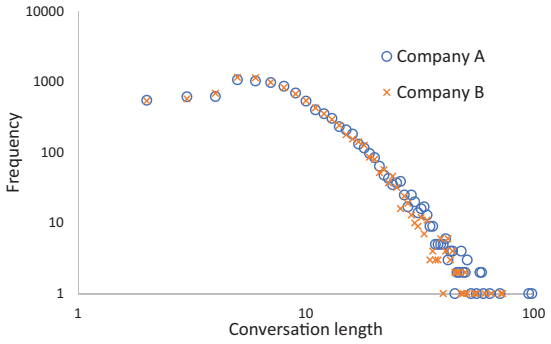


Figure 2: Frequency versus conversation length for *company A* and *company B* on a log-log scale.

We generated true binary labels by considering a conversation to be egregious if at least three of the four judges agreed. The inter-rater reliability between all judges, measured by Cohen's Kappa, was 0.72 which indicates high level agreement. This process generated the egregious class sizes of 95 (8.6%) and 16 (8%) for *company A* and *company B*, respectively. This verifies the unbalanced data expectation as previously discussed.

We also implemented two baseline models, rule-based and text-based, as follows:

Rule-based. In this approach, we look for cases in which the virtual agent responded with a "not trained" reply, or occurrences of the customer requesting to talk to a human agent. As discussed earlier, these may be indicative of the customer's dissatisfaction with the nature of the virtual agent's responses.

Text-based. A model that was trained to predict egregiousness given the conversation's text (all customer and agent's text dur-

[8]judges that are HCI experts and have experience in designing conversational agents systems.

	Egregious			Non-Egregious		
Model	P	R	F	P	R	F
Rule-based	0.28	0.54	0.37	0.95	0.87	0.91
Text-based	0.46	0.56	0.50	0.96	**0.94**	**0.95**
EGR	**0.47**	**0.79**	**0.59**	**0.98**	0.92	**0.95**

Table 2: Cross-validation results for the baselines and *EGR* models.

ing the conversation). This model was implemented using state-of-the-art textual features as in (Herzig et al., 2017). In (Herzig et al., 2017) emotions are detected from text, which can be thought of as similar to our task of predicting egregious conversations.

We evaluated these baseline methods against our classifier using 10-fold cross-validation over *company A*'s dataset (we did not use *company B*'s data for training due to the low number of tagged conversations). Since class distribution is unbalanced, we evaluated classification performance by using precision (P), recall (R) and F1-score (F) for each class. The *EGR* classifier was implemented using an SVM with a linear kernel[9].

4.3 Classification Results

Table 2 depicts the classification results for both classes and the three models we explored. The *EGR* model significantly outperformed both baselines[10]. Specifically, for the egregious class, the precision obtained by the text-based and *EGR* models were similar. This indicates that the text analyzed by both models encodes some information about egregiousness. On the other hand, for the recall and hence the F1-score, the *EGR* model relatively improved the text-based model by 41% and 18%, respectively. We will further analyze the models below.

4.4 Feature Set Contribution Analysis

To better understand the contributions of different sets of features to our *EGR* model, we examined various features in an incremental fashion. Based on the groups of feature sets that we defined in Section 3, we tested the performance of different group combinations, added in the following order: *agent*, *customer* and *customer-agent interactions*.

[9]http://scikit-learn.org/stable/modules/svm.html

[10]*EGR* with $p < 0.001$, using *McNemar's test.*

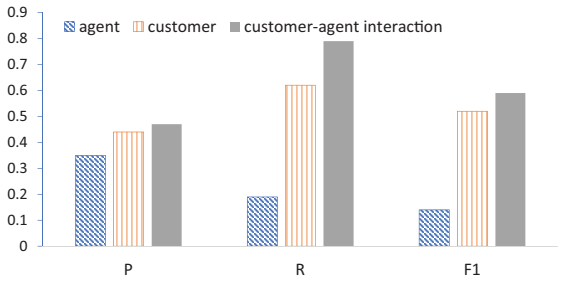


Figure 3: Precision (P), Recall (R), and F1-score (F) for various group combinations.

Figure 3 depicts the results for the classification task. The *x*-axis represents specific combinations of groups, and the *y*-axis represents the performance obtained. Figure 3 shows that adding each group improved performance, which indicates the informative value of each group. The figure also suggests that the most informative group in terms of prediction ability is the *customer* group.

4.5 Cross-Domain Analysis

We also studied how robust our features were: If our features generalize well, performance should not drop much when testing *company B* with the classifier trained exclusively on the data from *company A*. Although *company A* and *company B* share similar conversation engine platforms, they are completely different in terms of objectives, domain, terminology, etc. For this task, we utilized the 200 annotated conversations of *company B* as test data, and experimented with the different models, trained on *company A*'s data. The rule-based baseline does not require training, of course, and could be applied directly.

Table 3 summarizes the results showing that the performance of the *EGR* model is relatively stable (w.r.t the model's performance when it was trained and tested on the same domain), with a degradation of only 9% in F1-score[11]. In addition, the results also show that the text-based model performs poorly when applied to a different domain (F1-score of 0.11). This may occur since textual features are closely tied to the training domain.

[11]*EGR* model results are statistically significant compared to the baselines models with $p < 0.001$, using *McNemar's test.*

	Egregious			Non-Egregious		
Model	P	R	F	P	R	F
Rule-based	0.15	0.12	0.14	0.93	0.94	0.93
Text-based	0.33	0.06	0.11	0.92	**0.99**	**0.96**
EGR	**0.41**	**0.81**	**0.54**	**0.98**	0.90	0.94

Table 3: Cross domain performance (models trained on *company A*'s data, tested on *company B*'s data).

	Egregious	Non-egregious
NLU error	48%	48%
LG limitation	33%	37%
Unsupported intent error	18%	14%

Table 4: Percentage of different customer rephrasing reasons for egregious, and non-egregious conversations.

4.6 Models Analysis

4.6.1 Customer Rephrasing Analysis

Inspired by (Sarikaya, 2017; Sano et al., 2017) we analyzed the customer rephrasing motivations for both the egregious and the non-egregious classes. First, we detected customer rephrasing as described in Section 3.2.1, and then assigned to each its motivation. Specifically, in our setting, the relevant motivations are[12]: (1) *Natural language understanding (NLU) error* - the agent's intent detection is wrong, and thus the agent's response is semantically far from the customer's turn; (2) *Language generation (LG) limitation* - the intent is detected correctly, but the customer is not satisfied by the response (for example, the response was too generic); (3) *Unsupported intent error* - the customer's intent is not supported by the agent.

In order to detect *NLU errors*, we measured the similarity between the first customer turn (before the rephrasing) and the agent response. We followed the methodology presented in (Jovita et al., 2015) claiming that the best answer given by the system has the highest similarity value between the customer turn and the agent answer. Thus, if the similarity was < 0.8 we considered this as an erroneous detection. If the similarity was ≥ 0.8 we considered the detection as correct, and thus the rephrasing occurred due to *LG limitation*. To detect *unsupported intent error* we used the approach described in Section 3.1.2. As reported in table 4, rephrasing due to an unsupported intent is more common in egregious conversations (18% vs. 14%), whereas, rephrasing due to generation limitations (*LG limitation*) is more common in non-egregious conversations (37% vs. 33%). This indicates that customers are more tolerant of cases where the system understood their intent, but the response is not exactly what they expected, rather than cases where the system's response was "not trained". Finally, the percentage of rephrasing due to wrong intent detection (*NLU errors*) is similar for both classes, which is somewhat expected as similar underlying systems provided NLU support.

4.6.2 Recall Analysis

We further investigated why the *EGR* model was better at identifying egregious conversations (i.e., its recall was higher compared to the baseline models). We manually examined 26 egregious conversations that were identified justly so by the *EGR* model, but misclassified by the other models. Those conversations were particularly prevalent with the agent's difficulty to identify correctly the user's intent due to *NLU errors* or *LG limitation*. We did not encounter any *unsupported intent errors* leading to customer rephrasing, which affected the ability of the rule-based model to classify those conversations as egregious. In addition, the customer intents that appeared in those conversations were very diverse. While customer rephrasing was captured by the *EGR* model, for the text-based model some of the intents were new (did not appear in the training data) and thus were difficult for the model to capture.

5 Conclusions and Future Work

In this paper, we have shown how it is possible to detect egregious conversations using a combination of customer utterances, agent responses, and customer-agent interactional features. As explained, the goal of this work is to give developers of automated agents tools to detect and then solve problems cre-

[12]We did not consider other motivations like automatic speech recognition (ASR) errors, fallback to search, and backend failure as they are not relevant to our setting.

ated by exceptionally bad conversations. In this context, future work includes collecting more data and using neural approaches (e.g., RNN, CNN) for analysis, validating our models on a range of domains beyond the two explored here. We also plan to extend the work to detect egregious conversations in real time (e.g., for escalating to a human operators), and create log analysis tools to analyze the root causes of egregious conversations and suggest possible remedies.

References

Abram Amsel. 1992. *Frustration Theory: An Analysis of Dispositional Learning and Memory*. Problems in the Behavioural Sciences. Cambridge University Press. https://doi.org/10.1017/CBO9780511665561.

Chayan Chakrabarti and George F. Luger. 2013. A framework for simulating and evaluating artificial chatter bot conversations. In *FLAIRS Conference*.

Ulrich Gnewuch, Stefan Morana, and Alexander Maedche. 2017. Towards designing cooperative and social conversational agents for customer service. In *Proceedings of the International Conference on Information Systems (ICIS)*.

Melita Hajdinjak and France Mihelic. 2006. The paradise evaluation framework: Issues and findings. *Comput. Linguist.* 32(2).

Helen Wright Hastie, Rashmi Prasad, and Marilyn A. Walker. 2002. What's the problem: Automatically identifying problematic dialogues in DARPA communicator dialogue systems. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA.*. pages 384–391.

Jonathan Herzig, Michal Shmueli-Scheuer, and David Konopnicki. 2017. Emotion detection from text via ensemble classification using word embeddings. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*. ACM, New York, NY, USA, ICTIR '17, pages 269–272. https://doi.org/10.1145/3121050.3121093.

Julia Hirschberg and Christopher D. Manning. 2015. Advances in natural language processing. *Science* 349(6245):261–266.

Jiepu Jiang, Ahmed Hassan Awadallah, Rosie Jones, Umut Ozertem, Imed Zitouni, Ranjitha Gurunath Kulkarni, and Omar Zia Khan. 2015. Automatic online evaluation of intelligent assistants. In *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*. pages 506–516.

Jovita, Linda, Andrei Hartawan, and Derwin Suhartono. 2015. Using vector space model in question answering system. *Procedia Computer Science* 59:305 – 311. International Conference on Computer Science and Computational Intelligence (ICCSCI 2015).

Julia Kiseleva, Kyle Williams, Ahmed Hassan Awadallah, Aidan C. Crook, Imed Zitouni, and Tasos Anastasakos. 2016a. Predicting user satisfaction with intelligent assistants. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '16, pages 45–54.

Julia Kiseleva, Kyle Williams, Jiepu Jiang, Ahmed Hassan Awadallah, Aidan C. Crook, Imed Zitouni, and Tasos Anastasakos. 2016b. Understanding user satisfaction with intelligent assistants. In *Proceedings of the 2016 ACM on Conference on Human Information Interaction and Retrieval*. CHIIR '16, pages 121–130.

Tina Klüwer. 2011. *"I Like Your Shirt" - Dialogue Acts for Enabling Social Talk in Conversational Agents*, pages 14–27.

Nicole C. Krämer. 2008. Social effects of virtual assistants. a review of empirical results with regard to communication. In *Proceedings of the 8th International Conference on Intelligent Virtual Agents*. IVA '08.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. pages 2122–2132.

Mohammed Slim Ben Mimoun, Ingrid Poncin, and Marion Garnier. 2012. Case study embodied virtual agents: An analysis on reasons for failure. *Journal of Retailing and Consumer Services* 19(6):605 – 612.

C. Nass and Moon,Y. 2000. Machines and mindlessness: Social responses to computers. *Journal of Social Issues* 56:81–103.

Richard L Oliver. 2014. *Satisfaction: A behavioral perspective on the consumer*. Routledge.

Byron Reeves and Clifford Nass. 1996. *The Media Equation: How People Treat Computers, Television, and New Media Like Real People and Places*. Cambridge University Press, New York, NY, USA.

Aude Rychalski and Sarah Hudson. 2017. Asymmetric effects of customer emotions on satisfaction and loyalty in a utilitarian service context. *Journal of Business Research* 71:84 – 91.

Tommy Sandbank, Michal Shmueli-Scheuer, Jonathan Herzig, David Konopnicki, and Rottem Shaul. 2017. Ehctool: Managing emotional hotspots for conversational agents. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces Companion*. ACM, New York, NY, USA, IUI '17 Companion, pages 125–128. https://doi.org/10.1145/3030024.3038260.

Shumpei Sano, Nobuhiro Kaji, and Manabu Sassano. 2017. Predicting causes of reformulation in intelligent assistants. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*. pages 299–309.

R. Sarikaya. 2017. The technology behind personal digital assistants: An overview of the system architecture and key components. *IEEE Signal Processing Magazine* 34(1):67–81.

Cicekli I. Saygin A. P. 2002. Pragmatics in human-computer conversations. *Journal of Pragmatics* 34(3).

Moller Sebastian, Klaus-Peter Engelbrecht, Christine Kuhnel, Ina Wechsung, and Benjamin Weiss. 2009. A taxonomy of quality of service and quality of experience of multimodal human-machine interaction. In *Quality of Multimedia Experience, 2009. QoMEx 2009. International Workshop on. IEEE*.

Stefan Steidl, Christian Hacker, Christine Ruff, Anton Batliner, Elmar Nöth, and Jürgen Haas. 2004. *Looking at the Last Two Turns, I'd Say This Dialogue Is Doomed – Measuring Dialogue Success*, pages 629–636.

Marilyn A. Walker, Diane J. Litman, Candace A. Kamm, and Alicia Abella. 1997. Paradise: A framework for evaluating spoken dialogue agents. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '98, pages 271–280. https://doi.org/10.3115/976909.979652.

Marilyn A. Walker, Rebecca Passonneau, and Julie E. Boland. 2001. Quantitative and qualitative evaluation of darpa communicator spoken dialogue systems. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*. ACL '01, pages 515–522.

Marilyn A. Walker, Jeremy H. Wright, and Irene Langkilde. 2000. Using natural language processing and discourse features to identify understanding errors. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*. pages 1111–1118.

Nick Webb, David Benyon, Preben Hansen, and Oil Mival. 2010. Evaluating human-machine conversation for appropriateness. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*.

Valarie Zeithaml, A Parsu Parasuraman, and Leonard Berry. 1990. Delivering quality service: Balancing customer perceptions and expectations .

Learning to Disentangle Interleaved Conversational Threads with a Siamese Hierarchical Network and Similarity Ranking

Jyun-Yu Jiang[†], **Francine Chen**[‡], **Yan-Ying Chen**[‡] **and Wei Wang**[†]
[†]University of California, Los Angeles, CA, USA
[‡]FX Palo Alto Laboratory, Palo Alto, CA, USA
jyunyu@cs.ucla.edu, {chen,yanying}@fxpal.com, weiwang@cs.ucla.edu

Abstract

An enormous amount of conversation occurs online every day, such as on chat platforms where multiple conversations may take place concurrently. Interleaved conversations lead to difficulties in not only following discussions but also retrieving relevant information from simultaneous messages. Conversation disentanglement aims to separate intermingled messages into detached conversations.

In this paper, we propose to leverage representation learning for conversation disentanglement. A Siamese hierarchical convolutional neural network (SHCNN), which integrates local and more global representations of a message, is first presented to estimate the conversation-level similarity between closely posted messages. With the estimated similarity scores, our algorithm for conversation identification by similarity ranking (CISIR) then derives conversations based on high-confidence message pairs and pairwise redundancy. Experiments were conducted with four publicly available datasets of conversations from Reddit and IRC channels. The experimental results show that our approach significantly outperforms comparative baselines in both pairwise similarity estimation and conversation disentanglement.

1 Introduction

With the growth of ubiquitous internet and mobile devices, people now commonly communicate in the virtual world. Among the various methods of communication, text-based conversational media, such as internet relay chat (IRC) (Werry, 1996) and Facebook Messenger[1], has been and remains one of the most popular choices. In addition, many enterprises have started to use conversational chat platforms such as Slack[2] to enhance team collaboration. However, multiple conversations may occur simultaneously when conversations involve three or more participants. Aoki et al. (2006) found an average of 1.79 conversations among eight participants at a time. Moreover, some platforms like chatrooms in Twitch may have more concurrent conversations (Hamilton et al., 2014). Interleaved conversations can lead to difficulties in both grasping discussions and identifying messages related to a search result. For example, Figure 1 shows a segment of conversations from the real-world IRC dataset as an example. Five interleaved threads are involved in only ten messages. Messages in the same thread may not have identical keywords. Moreover, a user (i.e., *Elma*) can participate in multiple threads. Hence, a robust mechanism to disentangle interleaved conversations can improve a user's satisfaction with a chat system.

Thread	Message
⋮	⋮
T31	*Malcolm: If running as root, I need to set up a global config rather than ∼/.fetchmailrc ?*
T38	***Elma:*** *i'm sure i missed something but fonts rendering in my gimp works isn't at its best*
T39	*Sena: is there anyway to see what the CPU temperature is?*
T38	***Elma:*** *is it because of gimp or i missed some tuning or such?*
T31	*Rache: Specify a non-default name run control file.*
T41	*Denny: so how does one enforce a permission set and ownership set on a folder and all its children?*
T31	*Malcolm: in the man page it doesn't mention any global fetchmailrc file... that is what was confusing me...*
T42	*Shenna: hi, are sata drives accessed as sda or hda?*
T41	***Elma:*** *-R for recursive...*
T42	***Elma:*** *sda*
⋮	⋮

Figure 1: A segment of real-world conversations involving six users and five (annotated) threads from the IRC dataset.

One solution for conversation disentanglement is to model the task as a topic detection and tracking (TDT) (Allan, 2002) task by deciding whether each incoming message starts a new topic or belongs to an existing conversation. Messages in the same conversation may have higher similarity

[1]Facebook Messenger: https://www.messenger.com/
[2]Slack: https://slack.com/

Proceedings of NAACL-HLT 2018, pages 1812–1822
New Orleans, Louisiana, June 1 - 6, 2018.

scores (Shen et al., 2006; Mayfield et al., 2012) or similar context messages (Wang and Oard, 2009). However, similarity thresholds for determining new topics vary depending on context. Embedding of earlier messages, resulting in duplication of parts of messages, can alter the similarity score. More specifically, the similarity scores obtained in previous work cannot well represent conversation-level relationships between messages.

Several studies have examined the use of statistical (Du et al., 2017) and linguistic features (Elsner and Charniak, 2008, 2010, 2011; Mayfield et al., 2012) for predicting user annotations of paired message similarity. These studies employed bag-of-words representations which do not capture term similarity and cannot distinguish word importance and relationships between words in a message. Thus, better representations of messages and their relationships are needed.

Recent studies have demonstrated the effectiveness of deep learning methods in representation learning (Bengio et al., 2013), aiming to infer low-dimensional distributed representations for sparse data such as text (Hinton and Salakhutdinov, 2006). These representations can be derived not only for words (Mikolov et al., 2013) but also sentences and documents (Le and Mikolov, 2014). In particular, convolutional neural networks (CNNs) have been shown to efficiently and effectively preserve important semantic and syntactic information from embedded text sequences (Blunsom et al., 2014). It has been demonstrated that CNNs produce state-of-the-art results in many NLP tasks such as text classification (Kim, 2014; Lai et al., 2015; Zhang et al., 2015) and sentiment analysis (Tang et al., 2014; Poria et al., 2015). Existing approaches, however, do not take advantage of deep learning techniques to model relationships between messages for disentangling conversations. (Mehri and Carenini, 2017) defined many statistical features for use with a random forest for in-thread classification and used a recurrent neural network (RNN) only to model adjacent messages with an external dataset as a feature.

In this paper, we aim to leverage deep learning for conversation disentanglement. Our proposed approach consists of two stages: (1) message pair similarity estimation and (2) conversation identification. In the first stage, we propose the Siamese hierarchical convolutional neural network (SHCNN) to estimate conversation-level similarity between pairs of closely posted messages. SHCNN is framed as a Siamese architecture (Mueller and Thyagarajan, 2016) concatenating the outputs of two hierarchical convolutional neural networks and additional features. Compared to other conventional CNN-based Siamese networks (Severyn and Moschitti, 2015; Yin et al., 2016), SHCNN models not only local information in adjacent words but also more global semantic information in a message. In the second stage, the algorithm of conversation identification by similarity ranking (CISIR) ranks messages within a time window paired with each message and constructs a message graph involving high-rank connections with strong confidence. Although only high-confidence relations are represented in the constructed graph, the redundancy of pairwise relationships can capture the connectivity of messages within a conversation.

In summary, the main contributions of this paper are threefold: **(1) Deep similarity estimation for conversation disentanglement:** To the best of our knowledge, this is the first study applying deep learning to estimate similarities between messages for disentangling conversations. SHCNN simultaneously captures and compares local and global characteristics of two messages to estimate their similarity. Message representations are also optimized towards the task of conversation disentanglement. **(2) Efficient and effective method:** The selection of message pairs posted closely in time and the proposed CISIR algorithm significantly reduces the computational time from $O\left(|M|^2\right)$ to $O\left(k|M|\right)$, where $|M|$ is the number of messages, and k is the maximum number of messages posted within a fixed-length time window. When many messages are posted over a long period, the computational time of our approach could be near-linear. **(3) Empirical improvements over previous work:** Extensive experiments have been conducted on four publicly available datasets, including three synthetic conversation datasets and one real conversation dataset from Reddit[3] and IRC conversations. Our approach outperforms all comparative baselines for both similarity estimation and conversation disentanglement.

2 Related Work

Methods for conversation disentanglement can be simply categorized into unsupervised and supervised approaches. Unsupervised approaches (Wang and Oard, 2009) estimate the relationship between messages through unsupervised similarity functions such cosine similarity, and assign messages to conversations based on a predefined

[3]Reddit: `https://www.reddit.com/`

threshold. In contrast, supervised methods exploit a set of user annotations (Elsner and Charniak, 2008; Mayfield et al., 2012; Shen et al., 2006; Du et al., 2017; Mehri and Carenini, 2017) to adapt to different datasets. Our approach can be classified as a supervised approach because a small set of user annotations is used to train the SHCNN.

In addition to conversations, some studies predict the partial structure of threaded data, especially for online forums (Aumayr et al., 2011; Wang et al., 2011b,a). These studies merely classify parent-child relationships in disentangled, independent threads. Moreover, they focus only on comments to the same post. Indeed, conversation disentanglement is a more difficult task.

Estimating the similarity of text pairs is an essential part in our approach. Many studies also focus on similar tasks aside from conversation disentanglement, such as entailment prediction (Mueller and Thyagarajan, 2016; Wang and Jiang, 2017) and question-answering (Severyn and Moschitti, 2015; Amiri et al., 2016; Yin et al., 2016). However, most of their models are complicated and require a larger amount of labeled training data; limited conversational data can lead to unsatisfactory performance as shown in Section 4.

3 Conversation Disentanglement

In this section, we formally define the objective of this work and notations used. A two-stage approach is then proposed to address the problem.

3.1 Problem Statement

Given a set of speakers $\boldsymbol{S}$, a message m is defined as a tuple $m = (\boldsymbol{w}, s, t)$, where $\boldsymbol{w} = \langle w_1, w_2, \cdots, w_n \rangle$ is a word sequence posted by the speaker $s \in \boldsymbol{S}$ at time t in seconds. Each message m is associated with a conversation $z\,(m)$. Messages in different conversations can be posted concurrently, i.e., conversations can be interleaved.

Following the settings of previous work (Elsner and Charniak, 2008, 2010, 2011; Mayfield et al., 2012), a set of pairwise annotations $\boldsymbol{A} = \{(m_i, m_j, y)\}$, where $y \in \{0, 1\}$, is given for training the model. More specifically, a Boolean value y indicates whether two messages m_i and m_j are in the same conversation, i.e., $z(m_i)$ and $z(m_j)$ are identical.

Given a set of messages $\boldsymbol{M}$ and the pairwise annotations $\boldsymbol{A}$ as training data, the goal is to learn a model that can identify whether messages are posted in the same conversation $z(m)$. Note that the number of conversations $|\boldsymbol{Z} = \{z(m) \mid \forall m \in \boldsymbol{M}\}|$ is always unknown to the system.

3.2 Framework Overview

Figure 2 illustrates our two-stage framework. The first stage aims to estimate pairwise similarity among messages. Message pair selection is applied to focus on the similarity between messages that are posted closely in time and thus more likely to be in the same conversation. The Siamese hierarchical CNN (SHCNN) is proposed for learning message representations and estimating pairwise similarity scores. The overlapping hierarchical structure of SHCNN models a message at multiple semantic levels and obtains representations that are more comprehensive.

In the second stage, our conversation identification by similarity ranking (CISIR) algorithm exploits the redundancy and connectivity of pairwise relationships to identify conversations as connected components in a message graph.

3.3 Message Pair Selection

Most of the previous work on conversation disentanglement focused on pairwise relationships between messages (Mayfield et al., 2012). Especially for single-pass clustering approaches, all pairs of messages need to be enumerated during similarity computation (Wang and Oard, 2009). However, if messages have been collected for a long time, the number of message pairs could be too mammoth to be processed in an acceptable amount of time. More precisely, it leads to at least $O(n^2)$ computational time, where n is the number of messages. As shown in Figure 3, the percentage of messages in the same conversation as a given message becomes significantly lower with a longer elapsed time between consecutive messages. In light of this observation, an assumption is made as follows:

Assumption 1 *The elapsed time between two consecutive messages posted in the same conversation is not greater than T hours, where T is a small number.*

More specifically, in our dataset every message m_i is posted within T hours earlier or later than any other message m_j in the same conversation, i.e., $\frac{|t_i - t_j|}{3600} < T$ for all pairs (m_i, m_j), where t is in seconds. For example, in the IRC dataset the average elapsed time between consecutive messages in a conversation is only 7 minutes. If a conversation is ongoing, there may not be an extended silence before a new message; conversely, an extended silence could be treated as the start of a new

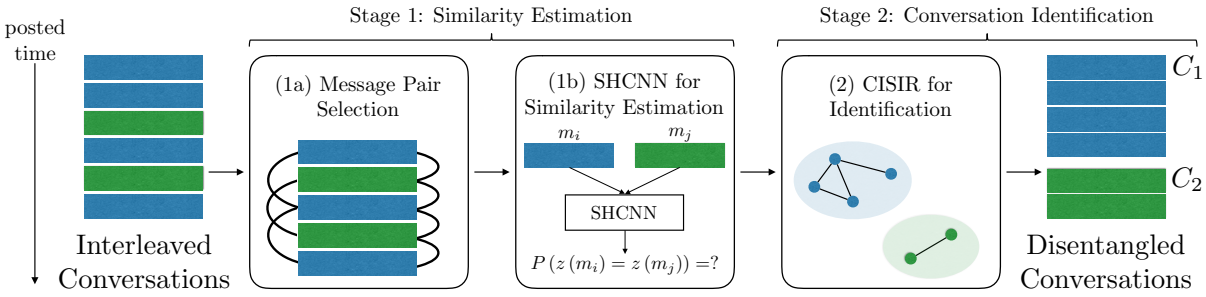


Figure 2: Illustration of our proposed two stage method. In the first stage, (1a) message pairs are selected for (1b) estimating pairwise similarity with a Siamese hierarchical CNN (SHCNN). In the second stage, (2) the algorithm of conversation identification by similarity ranking (CISIR) constructs a graph with strong relationships among messages and finds conversations as connected components.

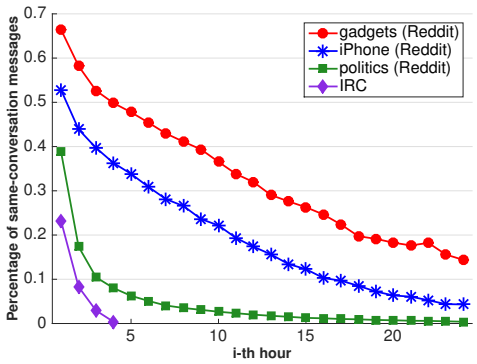


Figure 3: The percentage of messages in the same conversation as a given message with elapsed time between messages no greater than i hours for four experimental datasets.

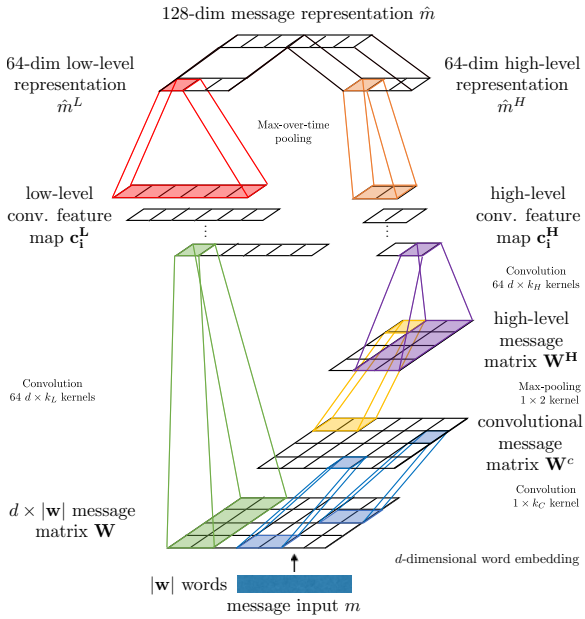


Figure 4: Illustration of hierarchical CNN (HCNN) for message representation. The labels with a larger font size indicate the corresponding tensors, and the labels with a smaller font size explain the operations between tensors.

conversation. With this assumption, the number of pairs can be reduced to $O(kn)$, where k is the maximum number of messages posted in a T-hour time window. By default T is set to 1 hour in our experiments.

In addition, it is worth mentioning that it may be possible to include conversational structure, such as replied-to relations, into the model. For example, after using CISIR to identify conversational threads, structure inference may be performed using methods such as described in (Aumayr et al., 2011) or (Wang et al., 2011b) and the structure used to refine the threads. In this study, we focus on only conversation disentanglement.

3.4 Similarity Estimation with the Siamese Hierarchical CNN (SHCNN)

Given a set of message pairs, we propose the Siamese hierarchical CNN (SHCNN) to estimate the similarity between a pair of messages.

3.4.1 Hierarchical CNN for Message Representation

The effectiveness of CNNs for representing text has already been addressed in previous studies. However, single-layer CNNs (Kim, 2014; Severyn and Moschitti, 2015) may not represent high-level semantics while low-level information could be diluted with multiple-layer CNNs (Yin et al., 2016). The hierarchical CNN (HCNN) is designed to simultaneously capture low- and high-level message meanings as shown in Figure 4.

A message m_i is first represented by a $d \times |\boldsymbol{w}|$ message matrix $\boldsymbol{W} \in \boldsymbol{R}^{d \times |\boldsymbol{w}|}$, where d is the dimension of a word embedding, and $|\boldsymbol{w}|$ is the num-

ber of words in a message. For low-level information, we exploit single-layer CNNs (Kim, 2014; Severyn and Moschitti, 2015) with a set of $d \times k_L$ kernels, where L denotes "Low", to extract n-gram semantics of k_L contiguous words. In this paper, 64 $d \times k_L$ kernels, where $k_L = 5$, are applied to obtain 64 low-level features $\hat{m}^L$. Note that the kernel row dimension is identical to the word embedding dimension to jointly consider the full embedding vector. As a consequence, convolution with each kernel produces a vector $\boldsymbol{c_i^L}$, which is then aggregated by max-over-time pooling (Collobert et al., 2011; Kim, 2014).

To acquire high-level semantics across a message, HCNN uses another multiple-layer CNN for feature extraction. A $1 \times k_C$ kernel is applied to $\boldsymbol{W}$, thereby generating a convolutional message matrix $\boldsymbol{W^C}$. Features covering broader contents are computed by applying a 1×2 kernel to a max-pooling layer with a stride of 2, producing a high-level message matrix $\boldsymbol{W^H}$. The row sizes of the two kernels are set to 1 to capture relations within each embedding dimension, and convolution is performed on $\boldsymbol{W^H}$ with 64 $d \times k_H$ kernels to capture relations across embedding dimensions. The generated convolutional feature maps $\boldsymbol{c_i^H}$ are subject to max-over-time pooling, resulting in 64 features $\hat{m}^H$. Finally, a message representation $\hat{m}$ is constructed by concatenating $\hat{m}^L$ and $\hat{m}^H$, i.e., creating a 128-dimensional feature vector, for characterizing both low- and high-level semantics of a message m. In this paper, both k_C and k_H are set to 5 while computing high-level representations.

3.4.2 Siamese Hierarchical CNN (SHCNN)

A Siamese structure with two identical subnetworks is useful to exploit the affinity between representations of two instances in the same hidden space (Severyn and Moschitti, 2015; Yin et al., 2016; Wang and Jiang, 2017). For similarity estimation, we propose the Siamese hierarchical CNN (SHCNN) using a Siamese structure that blends the outputs from two HCNNs as well as some context features.

Figure 5 shows the structure of the SHCNN for estimating the similarity between two messages m_i and m_j where the message representations $\hat{m}_i$ and $\hat{m}_j$ are generated by two sub-networks HCNNs (See Figure 4). There are many ways to deal with two sub-networks, such as using a similarity matrix (Severyn and Moschitti, 2015) or an attention matrix (Yin et al., 2016). However, both methods lead to an enormous number of parameters for long messages. We propose to independently compute the element-wise absolute differences (Mueller and Thyagarajan, 2016) between a pair of message representations $\hat{m}_i$ and $\hat{m}_j$, each from a sub-network. More formally, the absolute difference d is a vector where the k-th element is computed as $|\hat{m}_i(k) - \hat{m}_j(k)|$. This approach provides not only fewer parameters but also the flexibility to observe interactions among different dimensions in representations. Our experiments also show it outperforms the other two approaches in similarity estimation (See Section 4).

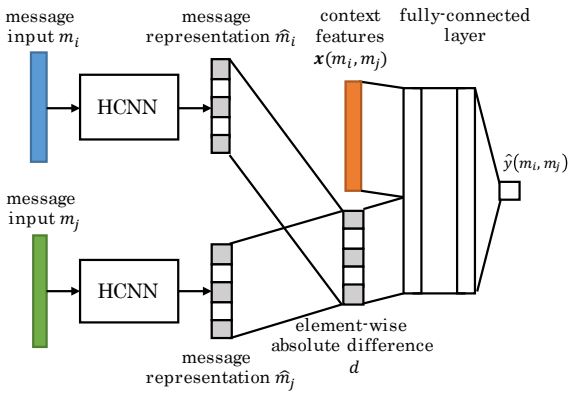


Figure 5: The siamese hierarchical CNN (SHCNN) for similarity estimation. Note that the model structure of an HCNN is shown in Figure 4.

In addition to message contents, contexts such as temporal and user information were also usually considered in previous studies about conversation disentanglement (Wang and Oard, 2009; Elsner and Charniak, 2010, 2011). In this paper, we focus on the performance of message content representations and only incorporate four context features: speaker identicality, absolute time difference and the number of duplicated words with and without weighting by inverse document frequency (Christopher et al., 2008). SHCNN concatenates the context features $\boldsymbol{x}(m_i, m_j)$ with the absolute difference d as the input of a fully-connected layer of the same size.

The final output of SHCNN $\hat{y}(m_i, m_j)$ is normalized by a logistic sigmoid function (Han and Moraga, 1995), representing the probability $P(z(m_i) = z(m_j))$.

3.4.3 Activation Functions

All convolutional layers and the fully-connected layer require activation functions, and the choice affects the performance (Maas et al., 2013). Popular functions include rectified linear units (ReLUs) (LeCun et al., 2015), hyperbolic tangent

units ($\tanh$) and exponential linear units (ELUs) (Clevert et al., 2016). In this study, we conducted informal comparison experiments and ELU was finally chosen for all functions because it performed the best.

3.4.4 Optimization and Implementation Details

Given a set of annotated message pairs $\boldsymbol{A} = \{(m_i, m_j, y)\}$, where y is a Boolean value indicating whether two messages are in the same conversation, SHCNN is optimized with binomial cross entropy (Goodfellow et al., 2016). More formally, the objective function is as follows:

$$\sum_{(m_i,m_j,y)\in A}[y \cdot \log(\hat{y}+\epsilon) + (1-y)\cdot\log(1-\hat{y}+\epsilon)] + \lambda||\boldsymbol{\theta}||^2$$

where $\hat{y}$ simplifies $\hat{y}(m_i, m_j)$, and ϵ is a small number, i.e., 10^{-9} in our experiments, preventing underflow errors. The term λ serves as the weight for L2-regularization for the set of parameters $\boldsymbol{\theta}$.

In our experiments, SHCNN is implemented by TensorFlow (Abadi et al., 2016) and trained by the Adam optimizer (Kingma and Ba, 2015) with an initial learning rate of 10^{-3}. The dropout technique (Srivastava et al., 2014) is utilized in the fully-connected layer with a dropout probability of 0.1. Word embeddings are initialized using the publicly available fastText 300-dimensional pretrained embeddings from Facebook (Bojanowski et al., 2016). The batch size is set to 512, and the maximum number of training epochs is 1,000. The final model is determined by evaluating the mean average precision (MAP) on a validation dataset every 100 iterations.

3.5 Conversation Identification by SImilarity Ranking (CISIR)

In the second stage of conversation disentanglement, i.e., part (2) in Figure 2, we aim to separate conversations based on the identified message pairs and their estimated similarity.

3.5.1 Graph-based Methods and Conversation Connectivity

It is intuitive to apply graph-based methods if pairwise relationships of messages are exploited (Elsner and Charniak, 2008). Furthermore, methods based on single-pass clustering (Wang and Oard, 2009) can be also be treated as graph-based methods. However, graph-based methods have a risky drawback: A single false positive connection between two messages can be propagated to several messages from different conversations. As shown in Figure 3, a certain percentage of message pairs are in different conversations, which can lead to numerous false positive connections.

False alarms may be reduced by raising the threshold that determines whether two messages are connected (Wang and Oard, 2009). However, a high threshold can make disentangled conversations fragmented and the best threshold for each pair could vary.

Algorithm 1: The algorithm of conversation disentanglement by similarity ranking (CISIR).

CISIR $(\boldsymbol{M}, \boldsymbol{D}, r, h)$;
Input : Message set $\boldsymbol{M}$, the set of selected message pairs $\boldsymbol{D}$, the threshold of similarity ranks r and the threshold of similarity scores h.
Output: A set of conversations $\boldsymbol{C}$
Let $G = (\boldsymbol{M}, \emptyset)$ be an undirected message graph
for $m \in M$ **do**
  $\boldsymbol{D}_m = \{(m_i, m_j, \hat{y}) \mid m_i = m \vee m_j = m\}$
  Rank entries in $\boldsymbol{D}_m$ by $\hat{y}$ in a descending order
  for $k = 1$ *to* $\min(r, |\boldsymbol{D}_m|)$ **do**
    Let $(m_i, m_j, \hat{y})$ be the k-th entry in ranked $\boldsymbol{D}_m$
    if $\hat{y} < h$ **then**
      break
    Add an edge (m_i, m_j) into G
$\boldsymbol{C}$ = ConnectedComponents(G)
return $\boldsymbol{C}$

3.5.2 The CISIR Algorithm

Instead of setting a high threshold, we propose the algorithm of Conversation Identification by SImilarity Ranking (CISIR). CISIR focuses on the top messages ranked by similarity scores. Based on Assumption 1, for each message, there exists at least one or more other messages in the same conversation posted closely in time. With this redundancy, a few pairs with stronger confidence, i.e., the top-ranked pairs, can be enough to extend a correct connectivity to earlier or later messages, while the low-ranked pairs can be ignored to reduce the risk of error propagation.

Given a set of selected message pairs with estimated similarity scores $\boldsymbol{D} = \{(m_i, m_j, \hat{y})\}$, Algorithm 1 shows the procedure of CISIR with two parameters r and h, where r is a high threshold

of similarity ranks and h is a lower threshold of similarity scores. Note that CISIR filters out pairs with low scores because a message can have more than r same-conversation pairs posted in its T-hour time window. For each message, CISIR ranks all of its associated pairs by the estimated similarity and only retrieves the top-r pairs whose similarity scores are greater than h. These retrieved high-confidence pairs are treated as the edges in a message graph G. Finally, CISIR divides G into connected components, and the messages in each connected component are treated as a conversation. In this paper, we use grid search to set r and h as 5 and 0.5, respectively.

3.5.3 Improvement of Time Complexity

The efficiency of Algorithm 1 can be further improved. The top-r qualified pairs for each message can be pre-processed by a scan of $\boldsymbol{D}$ with $|M|$ min-heaps which always contain at most $r+1$ elements. When r is a small constant number, it only takes $O(|D|) = O(k \cdot |M|)$ for pre-processing, where k is the maximum number of messages posted in a T-hour time window. With pre-processed top pairs, CISIR can do graph construction and find connected components in $O(k|M|)$, which compares favorably to conventional methods in $O(|M|^2)$.

4 Experiments

In this section, we conduct extensive experiments on four publicly available datasets to evaluate SHCNN and CISIR in two stages.

4.1 Datasets and Experimental Settings

4.1.1 Datasets

Three datasets from Reddit and one dataset of IRC are used as the experimental datasets.

- **Reddit Datasets**[4] The Reddit dataset is comprised of all posts and corresponding comments in all sub-reddits (i.e., forums in Reddit.com) from June 2016 to May 2017. Comments under a post can be treated as messages in one conversational thread. Here we manually merge all comments in a sub-reddit to construct a synthetic dataset of interleaved conversations. Note that although it is called a "synthetic dataset," all messages are written by real users. Three subreddits with different popularity levels as shown in Table 1 are selected to build three datasets: gadgets, iPhone and politics.

[4]The organized Reddit dataset is publicly available in `https://files.pushshift.io/reddit/`.

Dataset	Reddit			IRC
	gadgets	iPhone	politics	
Conversations	287	617	3,671	39
Messages	8,518	12,433	105,663	497
Speakers	5,185	5,231	25,289	71
Train/Valid Pairs	3,445	5,556	244,492	5,995
Test Pairs	27,565	44,450	1,955,943	47,966

Table 1: Statistics of four datasets after pre-processing.

- **IRC Dataset.** An annotated IRC dataset used in (Elsner and Charniak, 2008) is also included in our experiments. The IRC dataset consists of about 6 hours of messages in interleaved conversations. Even though the IRC dataset is significantly smaller and shorter than the Reddit datasets, it consists of natural, interleaved conversations with ground truth annotations, including thread id.

4.1.2 Experimental Settings

Humans may not participate in a large number of simultaneous conversations. e.g., an average of 1.79 for eight people (Aoki et al., 2006), but there could be hundreds of concurrent posts in a subreddit. Hence, we adjusted the datasets to be more similar to real conversations. Specifically we removed some conversations so that every dataset has at most ten conversations at any point in time. Short messages with less than five words are also removed because even for humans they are frequently ambiguous. Too short conversations with less than ten messages are also discarded as outliers (Ren et al., 2011). Training and validation data are randomly chosen from only 10% of the selected message pairs, respectively, because in real situations obtaining labels could be very costly. The remaining 80% of pairs are regarded as testing data. As a result, Table 1 shows the statistics of the four datasets after pre-processing.

4.2 Pairwise Similarity Estimation

Message pair similarity estimation is treated as a ranking task and evaluated with three ranking evaluation metrics: precision at 1 (P@1), mean average precision (MAP) and mean reciprocal rank (MRR) (Christopher et al., 2008). We compare the performance with six baseline methods, including the difference of posted time (*TimeDiff*), sameness of speakers (*Speaker*), cosine similarity of text (*Text-Sim*), the approach proposed by Elsner and Charniak (2008) (Elsner), *DeepQA* (Severyn and Moschitti, 2015) and *ABCNN* (Yin et al., 2016). Note that DeepQA and ABCNN are neural network-based models for question-answering. The approach of Mehri and Carenini

Dataset	Reddit Datasets									IRC Dataset		
	gadgets			iPhone			politics					
Metric	P@1	MRR	MAP	P@1	MRR	MAP	P@1	MRR	MAP	P@1	MRR	MAP
TimeDiff	0.6916	0.8237	0.8170	0.6085	0.7651	0.7495	0.4412	0.6362	0.5644	0.3262	0.5180	0.4384
Speaker	0.5643	0.7046	0.7425	0.5364	0.6595	0.6590	0.4021	0.4620	0.3914	0.4356	0.6263	0.6891
Text-Sim	0.7913	0.8746	0.8440	0.7347	0.8318	0.7872	0.5245	0.6672	0.5326	0.3712	0.5269	0.3108
Elsner	0.7758	0.8651	0.8321	0.6809	0.7935	0.7471	0.4643	0.6132	0.4884	0.1094	0.1886	0.2063
DeepQA	0.8011	0.8755	0.8511	0.7156	0.8112	0.7766	0.5593	0.6759	0.5685	0.7811	0.8182	0.8050
ABCNN	0.8374	0.8511	0.8502	0.8112	0.8520	0.8118	0.7419	0.6221	0.6644	0.7008	0.4142	0.5858
SHCNN	**0.8834**	**0.9281**	**0.9005**	**0.8375**	**0.8944**	**0.8497**	**0.7696**	**0.8392**	**0.6967**	0.9785	**0.9838**	**0.9819**
SHCNN (L)	0.8470	0.9080	0.8702	0.8066	0.8792	0.8275	0.7225	0.8070	0.6438	**0.9807**	0.9834	0.9750
SHCNN (H)	0.8490	0.9105	0.8704	0.8158	0.8851	0.8313	0.7228	0.8110	0.6283	0.9635	0.9728	0.8632

Table 2: Performance of pairwise similarity estimation in four datasets. Our approach is denoted as SHCNN.The performance with only low-level or high-level representations are denoted as SHCNN (L) and SHCNN (H). All improvements of SHCNN against the best baseline are significant at the 1% level of significance in a paired t-test.

$z(m_i)$	Message
T16	*"**Arlie:** Wow, maybe we just missed it when we were driving around"*
T18	*"**Arlie:** i've been very close to that situation myself"*

Figure 6: An example message pair in two different conversations from IRC shows how SHCNN discriminates between messages on different topics. The leftmost column is the conversation IDs of the corresponding messages. SHCNN predicts 0.67% of being in the same conversation for this pair while DeepQA with single-layer CNNs predicts 69.81%.

$z(m_i)$	Message
T16	*"Very well, I seem to be trying to show Arlie how its done and am **coding a webserver**."*
T16	*"Arlie: Good enough doesnt cut it! Is **the 'faster' method** a big change in design? Could I **implement** later without wanting to kill myself?"*

Figure 7: An example message pair in a conversation from IRC shows how SHCNN captures similarity in local information. The leftmost column is the conversation IDs of the corresponding messages. SHCNN predicts 70.41% for this pair while ABCNN with multiple-layer CNNs predicts 36.50%.

(2017) was not compared in our experiments because the RNN requires additional message sequences; moreover, its performance was only mildly better than Elsner, which performed poorly on IRC in Table 2.

Table 2 shows the performance of similarity estimation. Among all methods, neural network approaches (Severyn and Moschitti, 2015; Yin et al., 2016) perform better than other methods in most cases, indicating that message content representation has considerable impact on estimating pairwise similarity. SHCNN outperforms most of the baselines even if only low-level (L) or high-level (H) representations are exploited. When SHCNN captures both low- and high-level semantics, it significantly outperforms all baselines across the four datasets. For example, ABCNN can outperform SHCNN using only either low- or high-level representations in the politics dataset; however, SHCNN turns the tables after using both representations. An interesting observation is that ABCNN is the best baseline in every dataset except for IRC; this may be because the IRC data is too small to train complicated attention structures. On the contrary, our SHCNN can precisely capture semantics even with few parameters and limited data.

To shed deeper insights of how SHCNN surpasses other methods, we exhibit the prediction results of the IRC data and demonstrate the capability of SHCNN to simultaneously preserve local and more global information. Figure 6 presents an example to show how SHCNN is better than other methods in capturing more high-level topical information. Even though the main sentences of two messages are clearly on different topics, the baseline method DeepQA (Severyn and Moschitti, 2015) still predicts a high similarity. This could be attributed to the context of author mention (Wang and Oard, 2009) and a bias on the local information, i.e., the exact same term "Arlie", in the Siamese network used in DeepQA. On the contrary, SHCNN can capture more global information that differentiates the topics and correctly predicts a very low score. Figure 7 illustrates another example of how SHCNN outperforms other methods in preserving the similarity of local information. Both of the messages in the example have some segments related to software engineering. A baseline method ABCNN (Yin et al., 2016) with multiple-layer CNNs, however, still predicts a low score. This might be because both sentences are long so that the local information is diluted after processing by multiple CNN layers. Differently, SHCNN is able to seize local information, correctly predicting a high score.

Dataset	Reddit Datasets									IRC Dataset		
	gadgets			iPhone			politics					
Metric	NMI	ARI	F1	NMI	ARI	F1	NMI	ARI	F1	NMI	ARI	F1
Doc2Vec	0.1757	0.0008	0.0589	0.2318	0.0002	0.0718	0.2672	0.0001	0.0506	0.2046	0.0048	0.1711
Block-10	0.7745	0.1840	0.3411	0.8203	0.2349	0.4251	0.8338	0.1724	0.3451	0.4821	0.0819	0.2087
Speaker	0.7647	0.0440	0.2094	0.7861	0.1001	0.3339	0.7480	0.0637	0.2207	0.7394	0.4572	0.6310
CBME	0.6913	0.0212	0.1465	0.7280	0.0339	0.1966	0.7883	0.0165	0.1382	0.2818	0.0324	0.1970
GTM	0.7942	0.1787	0.2986	0.8198	0.0536	0.2566	0.8496	0.3076	0.4292	0.0226	0.0001	0.2064
CISIR	**0.8254**	**0.4287**	**0.4939**	**0.8552**	**0.4236**	**0.5187**	**0.8825**	**0.3561**	**0.4950**	**0.9330**	**0.9543**	**0.8798**
Oracle	0.8608	0.4852	0.5560	0.9003	0.5448	0.6358	0.9651	0.8286	0.8863	0.9838	0.9850	0.9819

Table 3: Performance of conversation disentanglement in four datasets. Our approach is denoted as CISIR. "Oracle" indicates the optimal performance if CISIR correctly retrieves all message pairs in identical conversations. All improvements of CISIR against the best baseline are significant at the 1% level of significance in a paired t-test.

4.3 Conversation Identification

For conversation identification, three clustering metrics are adopted for evaluation: normalized mutual information (NMI), adjusted rand index (ARI) and F_1 score (F1). Six methods are implemented as the baselines for conversation disentanglement, including *Doc2Vec* (Le and Mikolov, 2014), blocks of 10 messages (*Block-10*), messages of respective speakers (*Speaker*) (Elsner and Charniak, 2011), context-based message expansion (*CBME*) (Wang and Oard, 2009) and a graph-theoretical model with chat- and content-specific features (Elsner and Charniak, 2008) (*GTM*). The embedding-based clustering method, i.e., *Doc2Vec*, applies affinity propagation (Frey and Dueck, 2007) to cluster messages embedded using *Doc2Vec* without being given the number of clusters, with the idea that messages in the same conversation would form a cluster. Note that message pairs in the training and validation data are not utilized in prediction for a fair comparison to all methods.

Table 3 shows the performance of conversation disentanglement. Note that "Oracle" represents the optimal performance for CISIR when all message pairs in identical conversations in $\boldsymbol{D}$ are correctly retrieved. Because pairs in $\boldsymbol{D}$ may not have enough coverage to connect all messages in a coversation, the optimal performance could be lower than 1.0. CISIR performs better than all baseline methods for all datasets, and achieves excellent performance in IRC, due in part to the high-performing similarity estimates from the first stage. Among the baseline methods, GTM performs relatively well on all datasets except for IRC. This is because messages are more frequently posted in the IRC dataset, thereby increasing the number of incorrect pairs in the constructed graph. Examining the graph constructed by GTM, there are only two connected components, indicating that many conversations were incorrectly combined; in contrast, CISIR may be exempt from error propagation because it only relies on top-ranked pairs. Doc2Vec is trained to predict words in a document in an unsupervised manner. Its lowest performance in the experiments may point out a need for supervised learning in the specific task of conversation disentanglement to tackle the variation in semantic patterns. Time and author contextual cues do help conversation disentanglement as seen in the results of Block-10 and Speaker. Both of these contexts are integrated into our model.

5 Conclusions

In this paper, we propose a novel framework for disentangling conversations, including similarity estimation for message pairs and conversation identification. In contrast to previous work, we assume that we do not need to select all message pairs in the first stage, thereby reducing computational time without sacrificing performance too much. To estimate conversation-level similarity, a Siamese Hierarchical Convolutional Neural Network, SHCNN, is proposed to minimize the estimation error as well as preserve both the low- and high-level semantics of messages. In the second stage, we developed the Conversation Identification by SImilarity Ranking, CISIR, algorithm, which exploits the assumption made in the first stage and identifies individual, entangled conversations with high-ranked message pairs. Extensive experiments conducted on four publicly available datasets show that SHCNN and CISIR outperform several existing approaches in both similarity estimation and conversation identification.

Acknowledgement

We would like to thank the anonymous reviewers for their helpful comments. The work was partially supported by NIH U01HG008488, NIH R01GM115833, NIH U54GM114833, and NSF IIS-1313606.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *OSDI'16*. volume 16, pages 265–283.

James Allan. 2002. Introduction to topic detection and tracking. *Topic detection and tracking* pages 1–16.

Hadi Amiri, Philip Resnik, Jordan Boyd-Graber, and Hal Daumé III. 2016. Learning text pair similarity with context-sensitive autoencoders. In *ACL'16*. ACL, pages 1882–1892.

Paul M Aoki, Margaret H Szymanski, Luke Plurkowski, James D Thornton, Allison Woodruff, and Weilie Yi. 2006. Where's the party in multi-party?: Analyzing the structure of small-group sociable talk. In *CSCW'06*. ACM, pages 393–402.

Erik Aumayr, Jeffrey Chan, and Conor Hayes. 2011. Reconstruction of threaded conversations in online discussion forums. In *ICWSM'11*. pages 26–33.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *TPAMI* 35(8):1798–1828.

Phil Blunsom, Edward Grefenstette, and Nal Kalchbrenner. 2014. A convolutional neural network for modelling sentences. In *ACL'14*. ACL, pages 655–665.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606* .

D Manning Christopher, Raghavan Prabhakar, and SCHÜTZE Hinrich. 2008. Introduction to information retrieval. *An Introduction To Information Retrieval* 151:177.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2016. Fast and accurate deep network learning by exponential linear units (elus). In *ICLR'16*.

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

Wenchao Du, Pascal Poupart, and Wei Xu. 2017. Discovering conversational dependencies between messages in dialogs. In *AAAI'17*. pages 4917–4918.

Micha Elsner and Eugene Charniak. 2008. You talking to me? a corpus and algorithm for conversation disentanglement. In *ACL'08*. ACL, pages 834–842.

Micha Elsner and Eugene Charniak. 2010. Disentangling chat. *Computational Linguistics* 36(3):389–409.

Micha Elsner and Eugene Charniak. 2011. Disentangling chat with local coherence models. In *ACL-HLT'11*. ACL, pages 1179–1189.

Brendan J Frey and Delbert Dueck. 2007. Clustering by passing messages between data points. *science* 315(5814):972–976.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

William A Hamilton, Oliver Garretson, and Andruid Kerne. 2014. Streaming on twitch: fostering participatory communities of play within live mixed media. In *CHI'14*. ACM, pages 1315–1324.

Jun Han and Claudio Moraga. 1995. The influence of the sigmoid function parameters on the speed of backpropagation learning. *From Natural to Artificial Neural Computation* pages 195–201.

Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science* 313(5786):504–507.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP'14*. ACL.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR'16*.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI'15*. volume 333, pages 2267–2273.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML'14*. pages 1188–1196.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521(7553):436–444.

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *ICML'13*. volume 30.

Elijah Mayfield, David Adamson, and Carolyn Penstein Rosé. 2012. Hierarchical conversation structure prediction in multi-party chat. In *SIGDIAL'12*. ACL, pages 60–69.

Shikib Mehri and Giuseppe Carenini. 2017. Chat disentanglement: Identifying semantic reply relationships with random forests and recurrent neural networks. In *IJCNLP'17*. volume 1, pages 615–623.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS'13*. pages 3111–3119.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *AAAI'16*. pages 2786–2792.

Soujanya Poria, Erik Cambria, and Alexander Gelbukh. 2015. Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis. In *EMNLP'15*. ACL, pages 2539–2544.

Zhaochun Ren, Jun Ma, Shuaiqiang Wang, and Yang Liu. 2011. Summarizing web forum threads based on a latent topic propagation process. In *CIKM'11*. ACM, pages 879–884.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *SIGIR'15*. ACM, pages 373–382.

Dou Shen, Qiang Yang, Jian-Tao Sun, and Zheng Chen. 2006. Thread detection in dynamic text message streams. In *SIGIR'06*. ACM, pages 35–42.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR* 15(1):1929–1958.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL'14*. pages 1555–1565.

Hongning Wang, Chi Wang, ChengXiang Zhai, and Jiawei Han. 2011a. Learning online discussion structures by conditional random fields. In *SIGIR'11*. ACM, pages 435–444.

Li Wang, Marco Lui, Su Nam Kim, Joakim Nivre, and Timothy Baldwin. 2011b. Predicting thread discourse structure over technical web forums. In *EMNLP'11*. ACL, pages 13–25.

Lidan Wang and Douglas W Oard. 2009. Context-based message expansion for disentanglement of interleaved text conversations. In *NAACL'09*. ACL, pages 200–208.

Shuohang Wang and Jing Jiang. 2017. A compare-aggregate model for matching text sequences. In *ICLR'17*.

Christopher C Werry. 1996. Internet relay chat. *Computer-mediated communication: Linguistic, social and cross-cultural perspectives* pages 47–63.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *TACL* 4:259–272.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS'15*. pages 649–657.

Variational Knowledge Graph Reasoning

Wenhu Chen, Wenhan Xiong, Xifeng Yan, William Yang Wang
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106
{wenhuchen,xwhan,xyan,william}@cs.ucsb.edu

Abstract

Inferring missing links in knowledge graphs (KG) has attracted a lot of attention from the research community. In this paper, we tackle a practical query answering task involving predicting the relation of a given entity pair. We frame this prediction problem as an inference problem in a probabilistic graphical model and aim at resolving it from a variational inference perspective. In order to model the relation between the query entity pair, we assume that there exists an underlying latent variable (paths connecting two nodes) in the KG, which carries the equivalent semantics of their relations. However, due to the intractability of connections in large KGs, we propose to use variation inference to maximize the evidence lower bound. More specifically, our framework (DIVA) is composed of three modules, i.e. a posterior approximator, a prior (path finder), and a likelihood (path reasoner). By using variational inference, we are able to incorporate them closely into a unified architecture and jointly optimize them to perform KG reasoning. With active interactions among these sub-modules, DIVA is better at handling noise and coping with more complex reasoning scenarios. In order to evaluate our method, we conduct the experiment of the link prediction task on multiple datasets and achieve state-of-the-art performances on both datasets.

1 Introduction

Large-scaled knowledge graph supports a lot of downstream natural language processing tasks like question answering, response generation, etc. However, there are large amount of important facts missing in existing KG, which has significantly limited the capability of KG's application. Therefore, automated reasoning, or the ability for computing systems to make new inferences from the observed evidence, has attracted lots of attention from the research community. In recent years, there are surging interests in designing machine learning algorithms for complex reasoning tasks, especially in large knowledge graphs (KGs) where the countless entities and links have posed great challenges to traditional logic-based algorithms. Specifically, we situate our study in this large KG multi-hop reasoning scenario, where the goal is to design an automated inference model to complete the missing links between existing entities in large KGs. For examples, if the KG contains a fact like *president*(*BarackObama*, *USA*) and *spouse*(*Michelle, BarackObama*), then we would like the machines to complete the missing link *livesIn*(*Michelle*, *USA*) automatically. Systems for this task are essential to complex question answering applications.

To tackle the multi-hop link prediction problem, various approaches have been proposed. Some earlier works like PRA (Lao et al., 2011; Gardner et al., 2014, 2013) use bounded-depth random walk with restarts to obtain paths. More recently, DeepPath (Xiong et al., 2017) and MINERVA (Das et al., 2018), frame the path-finding problem as a Markov Decision Process (MDP) and utilize reinforcement learning (RL) to maximize the expected return. Another line of work along with ours are Chain-of-Reasoning (Das et al., 2016) and Compositional Reasoning (Neelakantan et al., 2015), which take multi-hop chains learned by PRA as input and aim to infer its relation.

Here we frame the KG reasoning task as a two sub-steps, i.e. "Path-Finding" and "Path-Reasoning". We found that most of the related research is only focused on one step, which leads to major drawbacks—lack of interactions between these two steps. More specifically, DeepPath (Xiong et al., 2017) and MINERVA (Das et al., 2018) can be interpreted as enhancing the "Path-Finding" step while compositional reasoning (Neelakantan et al., 2015) and chains of rea-

Proceedings of NAACL-HLT 2018, pages 1823–1832
New Orleans, Louisiana, June 1 - 6, 2018.

soning (Das et al., 2016) can be interpreted as enhancing the "Path-Reasoning" step. DeepPath is trained to find paths more efficiently between two given entities while being agnostic to whether the entity pairs are positive or negative, whereas MINERVA learns to reach target nodes given an entity-query pair while being agnostic to the quality of the searched path[1]. In contrast, chains of reasoning and compositional reasoning only learn to predict relation given paths while being agnostic to the path-finding procedure. The lack of interaction prevents the model from understanding more diverse inputs and make the model very sensitive to noise and adversarial samples.

In order to increase the robustness of existing KG reasoning model and handle noisier environments, we propose to combine these two steps together as a whole from the perspective of the latent variable graphic model. This graphic model views the paths as discrete latent variables and relation as the observed variables with a given entity pair as the condition, thus the path-finding module can be viewed as a prior distribution to infer the underlying links in the KG. In contrast, the path-reasoning module can be viewed as the likelihood distribution, which classifies underlying links into multiple classes. With this assumption, we introduce an approximate posterior and design a variational auto-encoder (Kingma and Welling, 2013) algorithm to maximize the evidence lower-bound. This variational framework closely incorporates two modules into a unified framework and jointly train them together. By active cooperations and interactions, the path finder can take into account the value of searched path and resort to the more meaningful paths. Meanwhile, the path reasoner can receive more diverse paths from the path finder and generalizes better to unseen scenarios. Our contributions are three-fold:

- We introduce a variational inference framework for KG reasoning, which tightly integrates the path-finding and path-reasoning processes to perform joint reasoning.
- We have successfully leveraged negative samples into training and increase the robustness of existing KG reasoning model.
- We show that our method can scale up to large KG and achieve state-of-the-art results on two popular datasets.

The rest of the paper is organized as follow. In Section 2 we will outline related work on KG embedding, multi-hop reasoning, and variational auto-encoder. We describe our variational knowledge reasoner DIVA in Section 3. Experimental results are presented in Section 4, and we conclude in Section 5.

2 Related Work

2.1 Knowledge Graph Embeddings

Embedding methods to model multi-relation data from KGs have been extensively studied in recent years (Nickel et al., 2011; Bordes et al., 2013; Socher et al., 2013; Lin et al., 2015; Trouillon et al., 2017). From a representation learning perspective, all these methods are trying to learn a projection from symbolic space to vector space. For each triple (e_s, r, e_d) in the KG, various score functions can be defined using either vector or matrix operations. Although these embedding approaches have been successful capturing the semantics of KG symbols (entities and relations) and achieving impressive results on knowledge base completion tasks, most of them fail to model multi-hop relation paths, which are indispensable for more complex reasoning tasks. Besides, since all these models operate solely on latent space, their predictions are barely interpretable.

2.2 Multi-Hop Reasoning

The Path-Ranking Algorithm (PRA) method is the first approach to use a random walk with restart mechanism to perform multi-hop reasoning. Later on, some research studies (Gardner et al., 2014, 2013) have revised the PRA algorithm to compute feature similarity in the vector space. These formula-based algorithms can create a large fan-out area, which potentially undermines the inference accuracy. To mitigate this problem, a Convolutional Neural Network(CNN)-based model (Toutanova et al., 2015) has been proposed to perform multi-hop reasoning. Recently, DeepPath (Xiong et al., 2017) and MINERVA (Das et al., 2018) view the multi-hop reasoning problem as a Markov Decision Process, and leverages REINFORCE (Williams, 1992) to efficiently search for paths in large knowledge graph. These two methods are reported to achieve state-of-the-art results, however, these two models both use heuristic rewards to drive the policy

[1] MINERVA assigns constant rewards to all paths reaching the destination while ignoring their qualities.

search, which could make their models sensitive to noises and adversarial examples.

2.3 Variational Auto-encoder

Variational Auto-Encoder (Kingma and Welling, 2013) is a very popular algorithm to perform approximate posterior inference in large-scale scenarios, especially in neural networks. Recently, VAE has been successfully applied to various complex machine learning tasks like image generation (Mansimov et al., 2015), machine translation (Zhang et al., 2016), sentence generation (Guu et al., 2017a) and question answering (Zhang et al., 2017). Zhang et al. (2017) is closest to ours, this paper proposes a variational framework to understand the variability of human language about entity referencing. In contrast, our model uses a variational framework to cope with the complex link connections in large KG. Unlike the previous research in VAE, both Zhang et al. (2017) and our model uses discrete variables as the latent representation to infer the semantics of given entity pairs. More specifically, we view the generation of relation as a stochastic process controlled by a latent representation, i.e. the connected multi-hop link existed in the KG. Though the potential link paths are discrete and countable, its amount is still very large and poses challenges to direct optimization. Therefore, we resort to variational auto-encoder as our approximation strategy.

3 Our Approach

3.1 Background

Here we formally define the background of our task. Let $\mathcal{E}$ be the set of entities and R be the set of relations. Then a KG is defined as a collection of triple facts (e_s, r, e_d), where $e_s, e_d \in \mathcal{E}$ and $r \in \mathcal{R}$. We are particularly interested in the problem of relation inference, which seeks to answer the question in the format of $(e_s, ?, e_d)$, the problem setting is slightly different from standard link prediction to answer the question of $(e_s, r, ?)$. Next, in order to tackle this classification problem, we assume that there is a latent representation for given entity pair in the KG, i.e. the collection of linked paths, these hidden variables can reveal the underlying semantics between these two entities. Therefore, the link classification problem can be decomposed into two modules – acquire underlying paths (Path Finder) and infer relation from latent representation (Path Reasoner).

Path Finder The state-of-the-art approach (Xiong et al., 2017; Das et al., 2018) is to view this process as a Markov Decision Process (MDP). A tuple $< S, A, P >$ is defined to represent the MDP, where S denotes the current state, e.g. the current node in the knowledge graph, A is the set of available actions, e.g. all the outgoing edges from the state, while P is the transition probability describing the state transition mechanism. In the knowledge graph, the transition of the state is deterministic, so we do not need to model the state transition P.

Path Reasoner The common approach (Lao et al., 2011; Neelakantan et al., 2015; Das et al., 2016) is to encode the path as a feature vector and use a multi-class discriminator to predict the unknown relation. PRA (Lao et al., 2011) proposes to encode paths as binary features to learn a log-linear classifier, while (Das et al., 2016) applies recurrent neural network to recursively encode the paths into hidden features and uses vector similarity for classification.

3.2 Variational KG Reasoner (DIVA)

Here we draw a schematic diagram of our model in Figure 1. Formally, we define the objective function for the general relation classification problem as follows:

$$\begin{aligned} Obj &= \sum_{(e_s,r,e_d)\in D} \log p(r|(e_s,e_d)) \\ &= \sum_{(e_s,r,e_d)\in D} \log \sum_{L} p_\theta(L|(e_s,e_d))p(r|L) \end{aligned} \tag{1}$$

where D is the dataset, (e_s, r, e_d) is the triple contained in the dataset, and L is the latent connecting paths. The evidence probability $p(r|(e_s, e_d))$ can be written as the marginalization of the product of two terms over the latent space. However, this evidence probability is intractable since it requires summing over the whole latent link space. Therefore, we propose to maximize its variational lower bound as follows:

$$\begin{aligned} ELBO = &\underset{L\sim q_\varphi(L|r,(e_s,e_d))}{E}[\log p_\theta(r|L)] - \\ &KL(q_\varphi(L|r,(e_s,e_d))||p_\beta(L|(e_s,e_d))) \end{aligned} \tag{2}$$

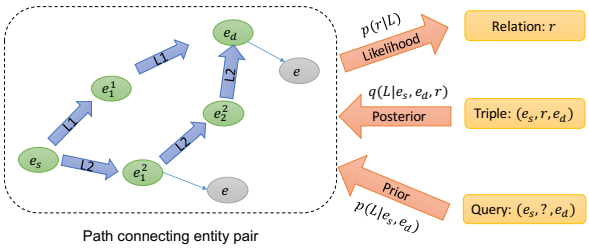


Figure 1: The probabilistic graphical model of our proposed approach. Arrows with dotted border represent the approximate posterior, which is modeled as a multinomial distribution over the whole link space. Arrows with solid border represent the prior and likelihood distributions.

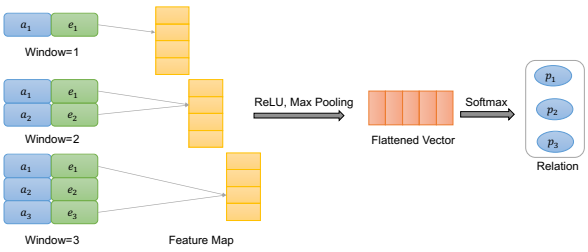


Figure 2: Overview of the CNN Path Reasoner.

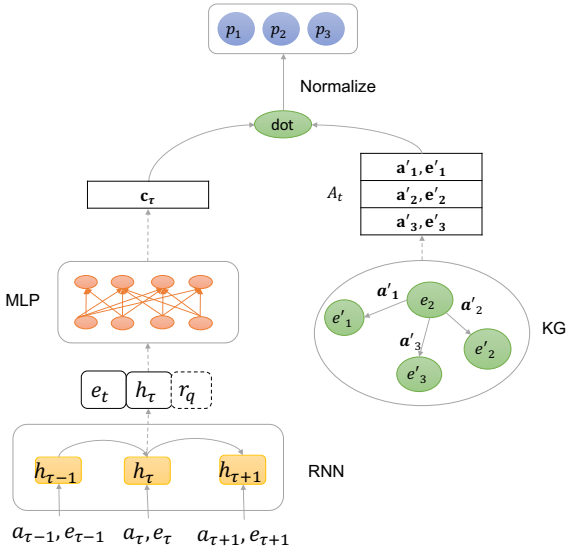


Figure 3: An overview of the path finder model. Note that r_q (query relation) exists in the approximate posterior while disappearing in the path finder model and e_t represents the target entity embedding, c_τ is the output of MLP layer at time step τ, a', e' denotes the connected edges and ends in the knowledge graphs.

Specifically, the ELBO (Kingma and Welling, 2013) is composed of three different terms – likelihood $p_\theta(r|L)$, prior $p_\beta(L|(e_s, e_t))$, and posterior $q_\varphi(L|(e_s, e_d), r)$. In this paper, we use three neural network models to parameterize these terms and then follow (Kingma and Welling, 2013) to apply variational auto-encoder to maximize the approximate lower bound. We describe these three models in details below:

Path Reasoner (Likelihood). Here we propose a path reasoner using Convolutional Neural Networks (CNN) (LeCun et al., 1995) and a feed-forward neural network. This model takes path sequence $L = \{a_1, e_1, \cdots, a_i, e_i, \cdots a_n, e_n\}$ to output a softmax probability over the relations set R, where a_i denotes the i-th intermediate relation and e_i denotes the i-th intermediate entity between the given entity pair. Here we first project them into embedding space and concatenate i-th relation embedding with i-th entity embedding as a combined vector, which we denote as $\{f_1, f_2, \cdots, f_n\}$ and $f_i \in \mathcal{R}^{2E}$. As shown in Figure 2, we pad the embedding sequence to a length of N. Then we design three convolution layers with window size of $(1 \times 2E), (2 \times 2E), (3 \times 2E)$, input channel size 1 and filter size D. After the convolution layer, we use $(N \times 1), (N - 1 \times 1), (N - 2 \times 1)$ to max pool the convolution feature map. Finally, we concatenate the three vectors as a combined vector $F \in \mathcal{R}^{3D}$. Finally, we use two-layered MLP with intermediate hidden size of M to output a softmax distribution over all the relations set R.

$$\begin{aligned} F &= f(f_1, f_2, \cdots, f_N) \\ p(r|L;\theta) &= softmax(W_r F + b_r) \end{aligned} \tag{3}$$

where f denotes the convolution and max-pooling function applied to extract reasoning path feature F, and W_r, b_r denote the weights and bias for the output feed-forward neural network.

Path Finder (Prior). Here we formulate the path finder $p(L|(e_s, e_d))$ as an MDP problem, and recursively predict actions (an outgoing relation-entity edge (a, e)) in every time step based on the previous history h_{t-1} as follows:

$$\begin{aligned} c_t &= ReLU(W_h[h_t; e_d] + b_h) \\ p((a_{t+1}, e_{t+1})|h_t, \beta) &= softmax(A_t c_t) \end{aligned} \tag{4}$$

where the $h_t \in \mathcal{R}^H$ denotes the history embedding, $e_d \in \mathcal{R}^E$ denotes the entity embedding, $A_t \in \mathcal{R}^{|A| \times 2E}$ is outgoing matrix which stacks the concatenated embeddings of all outgoing edges

and $|A|$ denotes the number of outgoing edge, we use W_h and b_h to represent the weight and bias of the feed-forward neural network outputting feature vector $c_t \in \mathcal{R}^{2E}$. The history embedding h_t is obtained using an LSTM network (Hochreiter and Schmidhuber, 1997) to encode all the previous decisions as follows:

$$h_t = LSTM(h_{t-1}, (a_t, e_t)) \tag{5}$$

As shown in Figure 3, the LSTM-based path finder interacts with the KG in every time step and decides which outgoing edge (a_{t+1}, e_{t+1}) to follow, search procedure will terminate either the target node is reached or the maximum step is reached.

Approximate Posterior. We formulate the posterior distribution $q(L|(e_s, e_d), r)$ following the similar architecture as the prior. The main difference lies in the fact that posterior approximator is aware of the relation r, therefore making more relevant decisions. The posterior borrows the history vector from finder as h_t, while the feed-forward neural network is distinctive in that it takes the relation embedding also into account. Formally, we write its outgoing distribution as follows:

$$\begin{aligned} u_t &= ReLU(W_{hp}[h_t; e_d; r] + b_{hp}) \\ q((a_{t+1}, e_{t+1})|h_t; \varphi) &= softmax(A_t u_t) \end{aligned} \tag{6}$$

where W_{hp} and b_{hp} denote the weight and bias for the feed-forward neural network.

3.3 Optimization

In order to maximize the ELBO with respect to the neural network models described above, we follow VAE (Kingma and Welling, 2013) to interpret the negative ELBO as two separate losses and minimize these them jointly using a gradient descent:

Reconstruction Loss. Here we name the first term of negative ELBO as reconstruction loss:

$$J_R = \underset{L \sim q_\varphi(L|r,(e_s,e_d))}{E}[-\log p_\theta(r|L)] \tag{7}$$

this loss function is motivated to reconstruct the relation R from the latent variable L sampled from approximate posterior, optimizing this loss function jointly can not only help the approximate posterior to obtain paths unique to particular relation r, but also teaches the path reasoner to reason over multiple hops and predict the correct relation.

KL-divergence Loss. We name the second term as KL-divergence loss:

$$J_{KL} = KL(q_\varphi(L|r, (e_s, e_d))|p_\beta(L|(e_s, e_d))) \tag{8}$$

this loss function is motivated to push the prior distribution towards the posterior distribution. The intuition of this loss lies in the fact that an entity pair already implies their relation, therefore, we can teach the path finder to approach the approximate posterior as much as possible. During test-time when we have no knowledge about relation, we use path finder to replace posterior approximator to search for high-quality paths.

Derivatives. We show the derivatives of the loss function with respect to three different models. For the approximate posterior, we re-weight the KL-diverge loss and design a joint loss function as follows:

$$J = J_R + w_{KL} J_{KL} \tag{9}$$

where w_{KL} is the re-weight factor to combine these two losses functions together. Formally, we write the derivative of posterior as follows:

$$\frac{\partial J}{\partial \varphi} = \underset{L \sim q_\varphi(L))}{E}[-f_{re}(L)\frac{\partial \log q_\varphi(L|(e_s, e_d), r)}{\partial \varphi} \tag{10}$$

where $f_{re}(L) = \log p_\theta + w_{KL} \log \frac{p_\beta}{q_\varphi}$ denotes the probability assigned by path reasoner. In practice, we found that the KL-reward term $\log \frac{p_\beta}{q_\varphi}$ causes severe instability during training, so we finally leave this term out by setting w_{KL} as 0. For the path reasoner, we also optimize its parameters with regard to the reconstruction as follows:

$$\frac{\partial J_R}{\partial \theta} = \underset{L \sim q_\varphi(L)}{E} - \frac{\partial \log p_\theta(r|L)}{\partial \theta} \tag{11}$$

For the path finder, we optimize its parameters with regard to the KL-divergence to teach it to infuse the relation information into the found links.

$$\frac{\partial J_{KL}}{\partial \beta} = \underset{L \sim q_\varphi(L)}{E} - \frac{\partial \log p_\beta(L|(e_s, e_d))}{\partial \beta} \tag{12}$$

Train & Test During training time, in contrast to the preceding methods like Das et al. (2018); Xiong et al. (2017), we also exploit negative samples by introducing an pseudo "n/a" relation,

Algorithm 1 The DIVA Algorithm.

procedure TRAINING & TESTING
Train:
  for episode ← 1 to N **do**
    Rollout K paths from posterior p_φ
    if Train-Posterior **then**
      $\varphi \leftarrow \varphi - \eta \times \frac{\partial L_r}{\partial \varphi}$
    else if Train-Likelihood **then**
      $\theta \leftarrow \theta - \eta \times \frac{\partial L_r}{\partial \theta}$
    else if Train-Prior **then**
      $\beta \leftarrow \beta - \eta \times \frac{\partial L_{KL}}{\partial \beta}$
    end if
  end for
Test MAP:
  Restore initial parameters θ, β
  Given sample $(e_s, r_q, (e_1, e_2, \cdots, e_n))$
  $L_i \leftarrow BeamSearch(p_\beta(L|e_s, e_i))$
  $S_i \leftarrow \frac{1}{|L_i|} \sum_{l \in L_i} p_\theta(r_q|l)$
  Sort S_i and find positive rank ra^+
  $MAP \leftarrow \frac{1}{1+ra^+}$
end procedure

which indicates "no-relation" between two entities. Therefore, we manage to decompose the data sample $(e_q, r_q, [e_1^-, e_2^-, \cdots, e_n^+])$ into a series of tuples (e_q, r'_q, e_i), where $r'_q = r_q$ for positive samples and $r'_q = n/a$ for negative samples. During training, we alternatively update three submodules with SGD. During test, we apply the path-finder to beam-search the top paths for all tuples and rank them based on the scores assign by path-reasoner. More specifically, we demonstrate the pseudo code in Algorithm 1.

3.4 Discussion

We here interpret the update of the posterior approximator in equation Equation 10 as a special case of REINFORCE (Williams, 1992), where we use Monte-Carlo sampling to estimate the expected return $\log p_\theta(r|L)$ for current posterior policy. This formula is very similar to DeepPath and MINERVA (Xiong et al., 2017; Das et al., 2018) in the sense that path-finding process is described as an exploration process to maximize the policy's long-term reward. Unlike these two models assigning heuristic rewards to the policy, our model assigns model-based reward $\log p_\theta(r|L)$, which is known to be more sophisticated and considers more implicit factors to distinguish between good and bad paths. Besides, our update formula for path reasoner Equation 11 is also similar to chain-of-reasoning (Das et al., 2016), both models are aimed at maximizing the likelihood of relation given the multi-hop chain. However, our model is distinctive from theirs in a sense that the obtained paths are sampled from a dynamic policy, by exposing more diverse paths to the path reasoner, it can generalize to more conditions. By the active interactions and collaborations of two models, DIVA is able to comprehend more complex inference scenarios and handle more noisy environments.

4 Experiments

To evaluate the performance of DIVA, we explore the standard link prediction task on two different-sized KG datasets and compare with the state-of-the-art algorithms. Link prediction is to rank a list of target entities $(e_1^-, e_2^-, \cdots, e_n^+)$ given a query entity e_q and query relation r_q. The dataset is arranged in the format of $(e_q, r_q, [e_1^-, e_2^-, \cdots, e_n^+])$, and the evaluation score (Mean Averaged Precision, MAP) is based on the ranked position of the positive sample.

4.1 Dataset and Setting

We perform experiments on two datasets, and the details of the statistics are described in Table 1. The samples of FB15k-237 (Toutanova et al., 2015) are sampled from FB15k (Bordes et al., 2013), here we follow DeepPath (Xiong et al., 2017) to select 20 relations including Sports, Locations, Film, etc. Our NELL dataset is downloaded from the released dataset[2], which contains 12 relations for evaluation. Besides, both datasets contain negative samples obtained by using the PRA code released by Lao et al. (2011). For each query r_q, we remove all the triples with r_q and r_q^{-1} during reasoning. During training, we set number of rollouts to 20 for each training sample and update the posterior distribution using Monte-Carlo REINFORCE (Williams, 1992) algorithm. During testing, we use a beam of 5 to approximate the whole search space for path finder. We follow MINERVA (Das et al., 2018) to set the maximum reasoning length to 3, which lowers the burden for the path-reasoner model. For both datasets, we set the embedding size E to 200, the history embedding size H to 200, the convolution kernel feature size D to 128, we set the hidden size of MLP for both path finder and path reasoner to 400.

[2]https://github.com/xwhan/DeepPath

Dataset	#Ent	#R	#Triples	#Tasks
FB15k-237	14,505	237	310,116	20
NELL-995	75,492	200	154,213	12

Table 1: Dataset statistics.

Model	12-rel MAP	9-rel MAP
RPA (Lao et al., 2011)	67.5	-
TransE (Bordes et al., 2013)	75.0	-
TransR (Lin et al., 2015)	74.0	-
TransD (Ji et al., 2015)	77.3	-
TransH (Wang et al., 2014)	75.1	-
MINERVA (Das et al., 2018)	-	**88.2**
DeepPath (Xiong et al., 2017)	79.6	80.2
RNN-Chain (Das et al., 2016)	79.0	80.2
CNN Path-Reasoner	82.0	82.2
DIVA	**88.6**	87.9

Table 2: MAP results on the NELL dataset. Since MINERVA (Das et al., 2018) only takes 9 relations out of the original 12 relations, we report the known results for both version of NELL-995 dataset.

4.2 Quantitative Results

We mainly compare with the embedding-based algorithms (Bordes et al., 2013; Lin et al., 2015; Ji et al., 2015; Wang et al., 2014), PRA (Lao et al., 2011), MINERVA (Das et al., 2018), DeepPath (Xiong et al., 2017) and Chain-of-Reasoning (Das et al., 2016), besides, we also take our standalone CNN path-reasoner from DIVA. Besides, we also try to directly maximize the marginal likelihood $p(r|e_s, e_d) = \sum_L p(L|e_s, e_d)p(r|L)$ using only the prior and likelihood model following MML (Guu et al., 2017b), which enables us to understand the superiority of introducing an approximate posterior. Here we first report our results for NELL-995 in Table 2, which is known to be a simple dataset and many existing algorithms already approach very significant accuracy. Then we test our methods in FB15k (Toutanova et al., 2015) and report our results in Table 3, which is much harder than NELL and arguably more relevant for real-world scenarios.

Besides, we also evaluate our model on FB-15k 20-relation subset with HITS@N score. Since our model only deals with the relation classification problem $(e_s, ?, e_d)$ with e_d as input, so it's hard for us to directly compare with MINERVA (Das et al., 2018). However, here we compare with chain-RNN (Das et al., 2016) and CNN Path-Reasoner model, the results are demonstrated as Table 4. Please note that the HITS@N score is computed against relation rather than entity.

Model	20-rel MAP
PRA (Lao et al., 2011)	54.1
TransE (Bordes et al., 2013)	53.2
TransR (Lin et al., 2015)	54.0
MINERVA (Das et al., 2018)	55.2
DeepPath (Xiong et al., 2017)	57.2
RNN-Chain (Das et al., 2016)	51.2
CNN Path-Reasoner	54.2
MML (Guu et al., 2017b)	58.7
DIVA	**59.8**

Table 3: Results on the FB15k dataset, please note that MINERVA's result is obtained based on our own implementation.

Model	HITS@3	HITS@5
RNN-Chain (Das et al., 2016)	0.80	0.82
CNN Path-Reasoner	0.82	0.83
DIVA	**0.84**	**0.86**

Table 4: HITS@N results on the FB15k dataset

Result Analysis We can observe from the above tables Table 3 and Table 2 that our algorithm has significantly outperformed most of the existing algorithms and achieves a very similar result as MINERVA (Das et al., 2018) on NELL dataset and achieves state-of-the-art results on FB15k. We conclude that our method is able to deal with more complex reasoning scenarios and is more robust to the adversarial examples. Besides, we also observe that our CNN Path-Reasoner can outperform the RNN-Chain (Das et al., 2016) on both datasets, we speculate that it is due to the short lengths of reasoning chains, which can extract more useful information from the reasoning chain.

From these two pie charts in Figure 5, we can observe that in NELL-995, very few errors are coming from the path reasoner since the path length is very small. A large proportion only contains a single hop. In contrast, most of the failures in the FB15k dataset are coming from the path reasoner, which fails to classify the multi-hop chain into correct relation. This analysis demonstrates that FB15k is much harder dataset and may be closer to real-life scenarios.

4.3 Beam Size Trade-offs

Here we are especially interested in studying the impact of different beam sizes in the link prediction tasks. With larger beam size, the path finder can obtain more linking paths, meanwhile, more noises are introduced to pose greater challenges for the path reasoner to infer the relation. With smaller beam size, the path finder will struggle to find connecting paths between positive entity

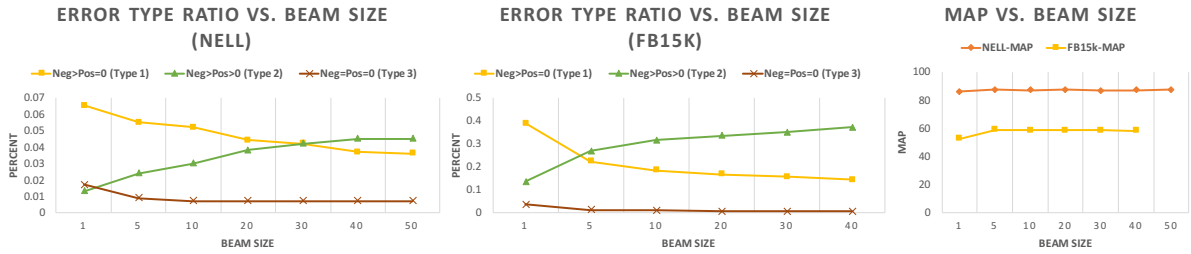


Figure 4: MAP results varying beam size and the error type's occurrence w.r.t to beam size. A beam size that is too large or too small would cause performance to drop.

Type	Reasoning Path	Score
Negative	athleteDirkNowitzki → *(athleteLedSportsteam)* → sportsteamMavericks	0.98
Positive	athleteDirkNowitzki → *(athleteLedSportsteam)* → sportsteamDallasMavericks	0.96
Explanation	*"maverick" is equivalent to "dallas-maverick", but treated as negative sample*	-
Negative	athleteRichHill → *(personBelongsToOrganization)* → sportsteamChicagoCubs	0.88
Positive	athleteRichHill → *(personBelongsToOrganization)* → sportsteamBlackhawks	0.74
Explanation	*Rich Hill plays in both sportsteam but the knowledge graph only include one*	-
Negative	coachNikolaiZherdev → *(athleteHomeStadium)* → stadiumOreventvenueGiantsStadium → *(teamHomestadium^{-1})* → sportsteamNewyorkGiants	0.98
Positive	coachNikolaiZherdev → *(athleteHomeStadium)* → stadiumOreventvenueGiantsStadium → *(teamHomestadium^{-1})* → sportsteam-rangers	0.72
Explanation	*The home stadium accommodates multiple teams, therefore the logic chain is not valid*	-

Table 5: The three samples separately indicates three frequent error types, the first one belongs to "duplicate entity", the second one belongs to "missing entity", while the last one is due to "wrong reasoning". Please note that the parenthesis terms denote relations while the non-parenthesis terms denote entities.

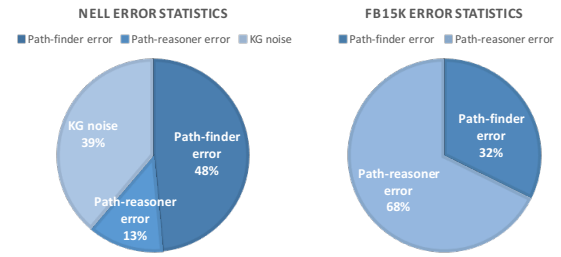


Figure 5: Error analysis for the NELL and FB15k link prediction task. Since FB15k dataset uses placeholders for entities, we are not able to analyze whether the error comes from KG noise.

pairs, meanwhile eliminating many noisy links. Therefore, we first mainly summarize three different types and investigate their changing curve under different beam size conditions:

1. No paths are found for positive samples, while paths are found for negative samples, which we denote as Neg>Pos=0.

2. Both positive samples and negative samples found paths, but the reasoner assigns higher scores to negative samples, which we denote as Neg>Pos>0.

3. Both negative and positive samples are not able to find paths in the knowledge graph, which we denote as Neg=Pos=0.

We draw the curves for MAP and error ratios in Figure 4 and we can easily observe the trade-offs, we found that using beam size of 5 can balance the burden of path-finder and path-reasoner optimally, therefore we keep to this beam size for the all the experiments.

4.4 Error Analysis

In order to investigate the bottleneck of DIVA, we take a subset from validation dataset to summarize the causes of different kinds of errors. Roughly, we classify errors into three categories, 1) KG noise: This error is caused by the KG itself, e.g some important relations are missing; some entities are duplicate; some nodes do not have valid outgoing edges. 2) Path-Finder error: This error is caused by the path finder, which fails to arrive destination. 3) Path-Reasoner error: This error

is caused by the path reasoner to assign a higher score to negative paths. Here we draw two pie charts to demonstrate the sources of reasoning errors in two reasoning tasks.

4.5 Failure Examples

We also show some failure samples in Table 5 to help understand where the errors are coming from. We can conclude that the "duplicate entity" and "missing entity" problems are mainly caused by the knowledge graph or the dataset, and the link prediction model has limited capability to resolve that. In contrast, the "wrong reasoning" problem is mainly caused by the reasoning model itself and can be improved with better algorithms.

5 Conclusion

In this paper, we propose a novel variational inference framework for knowledge graph reasoning. In contrast to prior studies that use a random walk with restarts (Lao et al., 2011) and explicit reinforcement learning path finding (Xiong et al., 2017), we situate our study in the context of variational inference in latent variable probabilistic graphical models. Our framework seamlessly integrates the path-finding and path-reasoning processes in a unified probabilistic framework, leveraging the strength of neural network based representation learning methods. Empirically, we show that our method has achieved the state-of-the-art performances on two popular datasets.

6 Acknowledgement

The authors would like to thank the anonymous reviewers for their thoughtful comments. This research was sponsored in part by the Army Research Laboratory under cooperative agreements W911NF09-2-0053 and NSF IIS 1528175. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notice herein.

References

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*. pages 2787–2795.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *International Conference on Learning Representations (ICLR)* .

Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. 2016. Chains of reasoning over entities, relations, and text using recurrent neural networks. *arXiv preprint arXiv:1607.01426* .

Matt Gardner, Partha Pratim Talukdar, Bryan Kisiel, and Tom Mitchell. 2013. Improving learning and inference in a large knowledge-base using latent syntactic cues .

Matt Gardner, Partha Pratim Talukdar, Jayant Krishnamurthy, and Tom Mitchell. 2014. Incorporating vector space similarity in random walk inference over knowledge bases .

Kelvin Guu, Tatsunori B Hashimoto, Yonatan Oren, and Percy Liang. 2017a. Generating sentences by editing prototypes. *arXiv preprint arXiv:1709.08878* .

Kelvin Guu, Panupong Pasupat, Evan Zheran Liu, and Percy Liang. 2017b. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. *arXiv preprint arXiv:1704.07926* .

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

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *ACL (1)*. pages 687–696.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* .

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

Yann LeCun, Yoshua Bengio, et al. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks* 3361(10):1995.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*. pages 2181–2187.

Elman Mansimov, Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. 2015. Generating images from captions with attention. *arXiv preprint arXiv:1511.02793* .

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base inference. In *2015 aaai spring symposium series*.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *ICML*. volume 11, pages 809–816.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*. pages 926–934.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *EMNLP*. volume 15, pages 1499–1509.

Théo Trouillon, Christopher R Dance, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2017. Knowledge graph completion via complex tensor factorization. *arXiv preprint arXiv:1702.06879* .

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*. pages 1112–1119.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. *arXiv preprint arXiv:1707.06690* .

Biao Zhang, Deyi Xiong, Jinsong Su, Hong Duan, and Min Zhang. 2016. Variational neural machine translation. *arXiv preprint arXiv:1605.07869* .

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J Smola, and Le Song. 2017. Variational reasoning for question answering with knowledge graph. *arXiv preprint arXiv:1709.04071* .

Inducing Temporal Relations from Time Anchor Annotation

Fei Cheng and **Yusuke Miyao**
Research Center for Financial Smart Data
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, Japan
{fei-cheng, yusuke}@nii.ac.jp

Abstract

Recognizing temporal relations among events and time expressions has been an essential but challenging task in natural language processing. Conventional annotation of judging temporal relations puts a heavy load on annotators. In reality, the existing annotated corpora include annotations on only "salient" event pairs, or on pairs in a fixed window of sentences. In this paper, we propose a new approach to obtain temporal relations from absolute time value (a.k.a. *time anchors*), which is suitable for texts containing rich temporal information such as news articles. We start from time anchors for events and time expressions, and temporal relation annotations are induced automatically by computing relative order of two time anchors. This proposal shows several advantages over the current methods for temporal relation annotation: it requires less annotation effort, can induce inter-sentence relations easily, and increases informativeness of temporal relations. We compare the empirical statistics and automatic recognition results with our data against a previous temporal relation corpus. We also reveal that our data contributes to a significant improvement of the downstream time anchor prediction task, demonstrating 14.1 point increase in overall accuracy.

1 Introduction

Temporal information extraction is becoming an active research field in natural language processing (NLP) due to the rapidly growing need for NLP applications such as timeline generation and question answering (Llorens et al., 2015; Meng et al., 2017). It has great potential to create many practical applications. For example, SemEval-2015 Task 4 (Minard et al., 2015) collects news articles about a target entity and the task required participants automatically ordering the events involving that entity in a timeline. The timeline representation of news can help people more easily comprehend a mass of information. This work aims to contribute to such timeline applications by extracting temporal information in specific domains like news articles.

TimeBank[1] (Pustejovsky et al., 2003) is the first widely used corpus with temporal information annotated in the NLP community. It contains 183 news articles that have been annotated with events, time expressions and temporal relations between events and time expressions. The annotation follows the TimeML[2] specification (Saurı et al., 2006). Along with the TimeBank and other temporal information corpora, a series of competitions on temporal information extraction (TempEval-1,2,3) (Verhagen et al., 2009, 2010; UzZaman et al., 2012) are attracting growing research efforts.

A majority of temporal information corpora adopt temporal links (TLINKs) to encode temporal information in documents. A TLINK denotes a temporal relation between mentions, i.e., events, time expressions and document creation time (DCT) (Setzer, 2002). However, annotating TLINKs is a painful work, because annotation candidates are quadratic to the number of mentions in a document. The original TimeBank only annotated those "salient" mention pairs judged by annotators, while the definition of "salient" is not necessarily clear. Annotators had to face a complicated task; identify "salient" mention pairs, and label temporal relations. For solving this, many dense annotation schemata are proposed to force annotators to annotate more or even complete graph pairs. However, dense annotation is time-consuming and unstable human judgments

[1] https://catalog.ldc.upenn.edu/LDC2006T08
[2] http://www.timeml.org/

Proceedings of NAACL-HLT 2018, pages 1833–1843
New Orleans, Louisiana, June 1 - 6, 2018.

on "salient" pairs are not improved at all. As a consequence, a high proportion of "vague" or "no-link" pairs appears in these dense corpora such as TimeBank-Dense (Cassidy et al., 2014).

In this work, we propose a new approach to obtain temporal relations from time anchors, i.e. absolute time value, of all mentions. We assume that a temporal relation can be induced by comparing the relative temporal order of two time anchors (e.g. *YYYY-MM-DD*) in a time axis. We use pre-defined rules (Section 3) to generate temporal order (TORDER) relations (e.g. BEFORE, AFTER, SAME_DAY, etc.) by taking two annotated time anchors as input. This proposal requires the annotation of time anchors, of which the annotation effort is linear with the number of mentions. This is the first work to obtain temporal relations shifted from the annotation of individual mentions, which is distinguished from most annotation work of manually annotating mention pairs.

This approach brings several advantages over the current temporal relation annotation. First, as long as time anchors of all mentions in a document are given, our pre-defined rules can induce the temporal relations for all the quadratic pairs. This skips the step of identifying "salient" pairs. Second, annotating the time anchors is relatively easy, as the annotation work is linear to the number of mentions. Third, the automatic generation rules can provide flexible relation types based on our definition and this increased informativeness might contribute positively to downstream tasks.

In our first evaluation (Section 4), we compare the correspondence and difference between the new TORDERs and conventional TLINKs. The comparison of empirical statistics shows the new data is label balanced, contains informative relations and reduces "vague" relations. Besides, the classification performance suggests the new data achieve reasonable accuracy, although accuracy numbers are not directly comparable.

Many text processing tasks are often requiring to know time anchors when events occurred in a timeline. In Section 5, we evaluate the data in a downstream time anchor prediction task (Reimers et al., 2016) by using the temporal relation recognizers separately trained with TORDERs or TLINKs. The main results show that the recognizer trained with our TORDERs significantly outperforms the recognizer trained with the TLINKs by 14.1 point *exact match* accuracy.

2 Background

2.1 Temporal Relation Annotation

TimeBank started a wave of data-driven temporal information extraction research in the NLP community. The original TimeBank only annotated relations judged to be *salient* by annotators and resulted in sparse annotations. Subsequent TempEval-1,2,3 competitions (Verhagen et al., 2009, 2010; UzZaman et al., 2012) mostly relied on TimeBank, but also aimed to improve coverage by annotating relations between all events and time expressions *in the same sentence*. However, most missing relations between mentions in different sentences are not considered.

In order to solve the sparsity issue, researchers started the work towards denser annotation schema. Bramsen et al. (2006) annotated multi-sentence segments of text to build directed acyclic graphs. Kolomiyets et al. (2012) annotated temporal dependency structures, though they only focused on relations between pairs of events. Do et al. (2012) produced the densest annotation and the annotator was required to annotate pairs "as many as possible". Cassidy et al. (2014) proposed a compulsory mechanism to force annotators to label every pair in a given sentence window. They performed the annotation (TimeBank-Dense) on a subset (36 documents) of TimeBank, which achieved a denser corpus with 6.3 TLINKs per event and time expression, comparing to 0.7 in the original TimeBank corpus. However, it raises the issue that hand-labeling all dense TLINKs is extremely time-consuming and the unclear definition of "salient" is not improved at all.

2.2 Temporal Relation Classification

The majority of the temporal relation classifiers focus on exploiting a variety of features to improve the performance in TimeBank. Laokulrat et al. (2013) extracted lexical and morphological features derived from WordNet synsets. Mani et al. (2006); D'Souza and Ng (2013) incorporated semantic relations between verbs from VerbOcean.

Recently, more researchers move on to diverse approaches on the TimeBank-Dense corpus. Chambers et al. (2014) proposed a multi-sieve classifier composed of several rule-based and machine learning based sieves ranked by their precision. Mirza and Tonelli (2016) started to mine the value of low-dimensional word embeddings by concatenating them with traditional sparse feature

vectors to improve their classifier.

Inspired by the success of the deep learning work in the similar task: relation extraction, Cheng and Miyao (2017) proposed the shortest dependency path based Bi-directional Long short-term memory (Hochreiter and Schmidhuber, 1997) (Bi-LSTM) to achieve state-of-the-art performance in the TimeBank-Dense corpus, which is adopted for the experiments in this paper. There are two reasons to use this classifier: 1) inter-sentence temporal relations are well treated. 2) only word, part-of-speech and dependency relation embeddings are required as input.

2.3 Time Anchor Annotation

A related task: Cross-Document Event Ordering (Minard et al., 2015) aims to order the events involving a target entity in a timeline given written news in English. Compared to traditional TLINKs, annotating time anchors of events is intuitively more straightforward in such tasks.

Reimers et al. (2016) proposed an annotation scheme, which requires annotators to infer the exact time of each individual event. They distinguished events that occur at a *Single-Day* from that span over *Multi-Day* by setting the granularity as one day. For *Single-Day* events, the event time is written in the format '*YYYY-MM-DD*' when the precise event time can be determined. Otherwise, they required annotators to narrow down the possible time as precisely as possible. An imprecise *Single-Day* event can be annotated as a tuple *(after, before)*, e.g. '*(after 1998-10-02,)*', '*(, before 2000-01-31)*' or '*(after 1998-10-02, before 2000-01-31)*'. In the case of *Multi-Day*, an event is annotated as a tuple *(begin, end)*, where *begin* and *end* are represented with *Single-Day*. For instance of a sentence:

> *The economy **created** jobs at a surprisingly robust pace in January, the government **reported** on Friday, evidence that America's economic stamina has withstood any **disruption** caused so far by the financial tumult in Asia.*

The *Multi-Day* event ***created*** is annotated as *(begin=1998-01-01, end=1998-01-31)*. The *Single-Day* event ***reported*** is annotated as the same day as DCT *(1998-02-06)*. The imprecise *Multi-Day* event ***disruption*** is annotated as *(begin=(, before1998-02-06), end=(, before1998-02-06))* as the event must have occurred before the time of this news, but the precise *begin* and *end* dates cannot be inferred from the text. Time anchors have the capability of anchoring all the events from a document into the same timeline as shown in Figure 1. They annotated the time anchors of total 1,498 events from 36 documents of TimeBank-Dense.

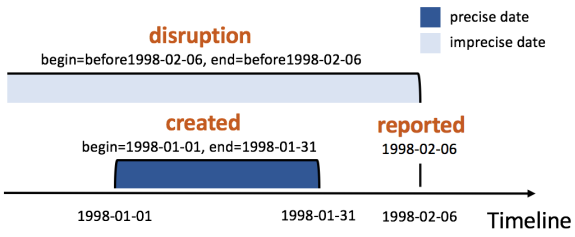


Figure 1: Anchoring events in a timeline

In temporal information retrieval, Berberich et al. (2010) proposed a four-tuple representation *('earliest begin', 'latest begin', 'earliest end', 'latest end')* for uncertain time expression (e.g. '1990s') in order to integrate such temporal information into language model. In the time anchor annotation, an event 'in 1990s' will be annotated as a *Multi-Day* event with imprecise *begin* and *end* points, i.e. *(begin=(after 1990-01-01, before1999-12-31), end=(after 1990-01-01, before1999-12-31))*, which is quite similar to their four-tuple representation.

3 Automatic generation of TORDERs

TimeML states that TLINKs present a temporal relation between event to event, event to time expression, and event to DCT. The sparse TLINK coverage in the majority of temporal information corpora is attributed to the unstable identification of "salient" pairs by human annotators. Denser annotation schemata somehow improved sparseness, but the annotation work became very time-consuming. These issues plague the development of temporal information extraction work.

Our temporal order (TORDER) proposal is designed with the goal of solving unstable recognition of "salient" pairs and reducing annotation effort. We hypothesize that a temporal relation can be automatically computed by comparing the relative temporal order between two time anchors (e.g. *YYYY-MM-DD*) in a time axis. We propose a set of pre-defined generation rules, which have the capability to rigorously induce a TORDER by taking the two annotated time anchors as input. Annotat-

TORDER	Condition
Two precise S_1 and S_2	
BEFORE	if $S_1 < S_2$
AFTER	if $S_1 > S_2$
SAME_DAY	if $S_1 = S_2$
A precise S_1 and an imprecise S_2 ($after_2, before_2$)	
BEFORE	if $S_1 \leq after_2$
AFTER	if $S_1 \geq before_2$
VAGUE	other cases
Two imprecise S_1 ($after_1, before_1$) and S_2 ($after_2, before_2$)	
BEFORE	if $before_1 \leq after_2$
AFTER	if $after_1 \geq before_2$
PVAGUE	if $before_1 = before_2$ and $after_1 = after_2$
VAGUE	other cases

Table 1: Definition of the temporal orders between two *Single-Day* events. '<', '>', '=' denote one event is in the left of, right of and same position as the other event in a left-to-right time axis.

TORDER	Condition
A *Single-Day* S_1 and a *Multi-Day* M_2 ($begin_2, end_2$)	
BEFORE	if S_1 **BEFORE** $begin_2$
AFTER	if S_1 **AFTER** end_2
IS_INCLUDED	if S_1 **AFTER/SAME_DAY** $begin_2$ and S_1 **BEFORE/SAME_DAY** end_2
VAGUE	other case
Two *Multi-Day* M_1 ($begin_1, end_1$) and M_2 ($begin_2, end_2$)	
BEFORE	if end_1 **BEFORE** $begin_2$
AFTER	if $begin_1$ **AFTER** end_2
SAME_SPAN	if $begin_1$ **SAME_DAY** $begin_2$ and end_1 **SAME_DAY** end_2
IS_INCLUDED	if $begin_1$ **AFTER/SAME_DAY** $begin_2$ and end_1 **BEFORE/SAME_DAY** end_2 (*)
INCLUDES	if $begin_1$ **BEFORE/SAME_DAY** $begin_2$ and end_1 **AFTER/SAME_DAY** end_2 (*)
PVAGUE	if $begin_1$ **PVAGUE/SAME_DAY** $begin_2$ and end_1 **PVAGUE/SAME_DAY** end_2 (*)
VAGUE	other cases

Table 2: Definition of the temporal orders involving *Multi-Day* events M $(begin, end)$. '*' denotes excluding the **SAME_SPAN** case in the current condition.

ing time anchors of individual mentions extremely reduces annotation effort, as it is linear with mention numbers. As long as time anchors are given, our pre-defined rules can induce the temporal relations for all the quadratic pairs, which skips the step of identifying "salient" pairs.

TimeBank contains the normalized date '*YYYY-MM-DD*' of time expressions and DCT, but does not include events' time. Our proposal of inducing a TORDER by comparing two time anchors requires the time anchor annotation of events in the same granularity as time expressions and DCT. Therefore, annotating the events with '*YYYY-MM-DD*' is a reasonable setting and one day is used as the minimal granularity of annotation. We choose the annotation (Reimers et al., 2016) of the day-level time anchors of events as the source of our automatic TORDER generator. In the case that a corpus can provide more specific time information '*YYYY-MM-DD, hh-mm-ss*' (e.g. this morning, three o'clock in the afternoon), our TORDER generator can be flexible to handle this information as long as the time anchors of all mentions are annotated in the same granularity.

For the clear demonstration of the definition of the auto-generated temporal order, we separately describe the generation of the pairs with two *Single-Day* mentions, and the pairs involving *Multi-Day* mentions. In this paper, TORDER labels are written in the upper-case bold font to be distinguished from TLINK labels written in the lower-case italic font. Table 1 introduces the definition of temporal orders between two *Single-Day* pairs S_1 and S_2. **PVAGUE** (i.e. partially vague) denotes that two imprecise time anchors are equivalent. For instance, we cannot induce a clear temporal relation between two events both occurring on *(,before1998-02-06)*, but nevertheless both events provide partially equivalent date information *'1998-02-06'*. It can possibly provide useful information for the future processes of classification or time inference. **PVAGUE** in the *Multi-Day* definition takes the same consideration.

In order to introduce the temporal orders involving *Multi-Day* events, a *Multi-Day* event M is denoted as a tuple of two *Single-Day* dates $(begin, end)$. A temporal order between a *Single-Day* S_1 and *Multi-Day* M_2 $(begin_2, end_2)$ can be derived by computing the temporal order of two *Single-Day* S_1 and $begin_2$, or S_1 and end_2 first. All the types of temporal orders involving *Multi-Day* events are defined in Table 2. One additional INCLUDES relation that *Multi-Day* event includes a *Single-Day* event can be obtained by reversing the symmetric IS_INCLUDED.

The example of automatically computing temporal orders can be demonstrated by using the events in Figure 1. Both *Multi-Day* ***created*** and ***disruption*** are clearly BEFORE the *Single-Day* ***reported***, because ***reported*** is AFTER the *end* dates of ***created*** and ***disruption***. The relation between ***created*** and ***disruption*** is induced as VAGUE, as the imprecise *begin*, *end* of ***disruption*** cannot be determined with a relation to ***created***.

In this paper, the definition adopts a similar relation set to TLINK for the purpose that we can perform fair comparison and evaluation in the next two sections. However, our inducing proposal can be very scalable to introduce more temporal relations. For instance, Allen's interval algebra (Allen, 1990) defines 'starts', 'finish' relations, which are not included in our current defini-

tion. We can easily extend our definition by detecting whether two time anchors have the equivalent *begin* or *end* points.

Our inducing proposal takes human annotated time expressions and normalized values as inputs to generate TORDER relations as the training data of the next processes (e.g. classification). In the case of processing raw texts, we can perform detection and normalization of time expressions by using existing temporal taggers, e.g. HeidelTime (Strötgen and Gertz, 2015), SUTime (Chang and Manning, 2012), etc.

4 Comparison of TORDERs and TLINKs

Fairly evaluating the TORDER's capability of encoding temporal order information compared to the existing data is difficult but necessary work. This section provides empirical statistics of TORDER and TLINK annotations, and compare the performance of automatic recognition. Additionally, we evaluate these two frameworks in a downstream task performance in Section 5.

4.1 Correspondences and Differences

Our new TORDERs are formally similar to the conventional TLINKs, as both state a temporal relation between two mentions. BEFORE and AFTER represent that one mention occurs before or after in a timeline, which is close to *before* and *after*. INCLUDES and IS_INCLUDED are more clearly conditioned as a *Single-Day* or *Multi-Day* mention occurs during the other *Multi-Day* mention, compared to *includes* and *is_included*. SAME_DAY and SAME_SPAN are designed for the one-day minimal granularity. Ideally, these two relations will include *simultaneous* and other TLINKs with two mentions occurring in the same day. VAGUE and PVAGUE state that our generation rules cannot induce the relations, similar to *vague* (i.e. annotators cannot judge the relations).

The one-day minimal granularity is the main reason causing the difference between TORDER and TLINK types. For a sentence:

> *I went to **sleep** after taking a **bath**.*

According to the TimeML specification, ***sleep*** is obviously *after* ***bath***. But in the one-day granularity, the relation is shifted to SAME_DAY. This brings the obstacle that we cannot measure whether the temporal information encoded in TORDERs is more informative than TLINKs by directly comparing the classification results.

Our TORDER definition shows the capability of capturing some relations which cannot be encoded by TLINK. For instance:

> Stocks ***rose***, ***pushing*** the Dow Jones industrial average up 72.24 points, to 8,189.49, ***leaving*** the index within 70 points of its record.

These TLINKs among the three events are annotated as *vague* in TimeBank-Dense, as the annotators cannot state their temporal orders. However, we can easily obtain SAME_DAY relations, since their day-level time anchors are the same.

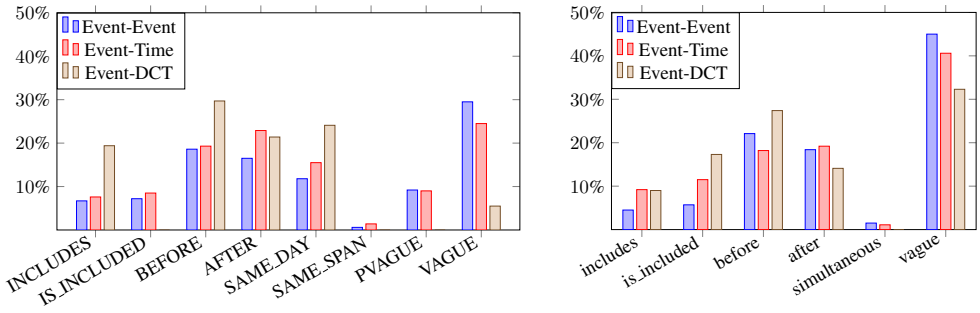
Imprecisely represented time anchors (e.g. *after YYYY-MM-DD*) are the major drawback of losing temporal order information. For instance:

> America's economic stamina has **withstood** any **disruption**...

The TLINK between ***withstood*** and ***disruption*** is annotated as *after*. While both of them were annotated as the same time anchor *(begin=before 1998-02-06, end= before 1998-02-06)*, our TORDER generator induced a PVAGUE relation and temporal order information is lost.

The hypothesis that our proposal skipping the unstable manual identification of "salient" pairs can reduce the VAGUE relations in the new data. This can be measured by comparing the numbers of the TORDER and TLINK relations on the same mention pairs. If the observation of a part of *vague* TLINKs induced as non-VAGUE TORDERs in the new data can be found, it will be the evidence.

Depending on the text domain, TLINKs or TORDERs can be advantageous in different scenarios. TLINKs can capture the temporal ordering information between events, when time expressions are often absent in the documents such as novels and narratives. But the annotation work is time consuming and a part of relations will be neglected by the unstable human identification of "salient" pairs. TORDERs have the capability of capturing more informative relations by skipping the "salient" pairs recognition and need less annotation effort. But they require that the events can be anchored in a timeline from a document (e.g. often the case of news articles) and imprecise time anchors cause some information loss.



(a) TORDER relations

(b) TLINK relations

Figure 2: The label distribution of the TORDER and TLINK relations

	b	***a***	***s***	***i***	***ii***	***v***
BEFORE	1486	24	0	22	26	542
AFTER	19	1242	5	26	66	503
SAME_DAY	155	93	83	164	343	647
SAME_SPAN	4	0	9	5	6	42
INCLUDES	104	61	2	225	25	372
IS_INCLUDED	56	71	1	25	214	333
PVAGUE	91	40	41	23	36	336
VAGUE	331	261	33	145	136	1464

Table 3: The comparison of the numbers of TORDER and TLINK annotations for the same mention pairs. ***b***:*before*, ***a***:*after*, ***s***:*simultaneous*, ***i***:*includes*, ***ii***:*is_included*, ***v***:*vague*.

4.2 Empirical Comparison

Investigating the quality of auto-generated TORDERs is important to demonstrate the value of this research. In this section, we empirically compare the statistics of the auto-generated TORDERs and human-annotated TLINKs. Theoretically, a TORDER between two mentions with any distance in a document can be automatically computed. However, it is important to make the new data in a comparable manner to the existing data. In this paper, we follow the process of TimeBank-Dense (Cassidy et al., 2014) to generate the complete graph of the 10,007 mention pairs in the same and adjacent sentences. The TORDER data used in this paper are publicly available[3] and our scalable generation method can be easily applied for inducing relations of longer distance pairs.

Table 3 shows the comparison between the numbers of the TimeBank-Dense TLINKs and the new TORDERs. One observation as we expected is that our approach captures new relations for a considerable part of the mention pairs that were judged as ***v*** (*vague*) in the human-annotated TLINKs. 542 *vague* relations are induced as AFTER in the new TORDERs, as well as other relation types. However, a part of non-*vague* TLINKs are shifted to VAGUE TORDERs. This matches our description of the imprecise time anchor issue. It is a trade-off between the part of mention pairs obtaining richer temporal information and the part of pairs losing information. That is the reason why we need a downstream task (i.e. Time Anchors Prediction in Section 5) to measure how much temporal order information is encoded in TORDERs and TLINKs. The shift of TLINK relations to SAME_DAY due to the one-day minimal granularity setting can also be clearly observed.

Figure 2 shows the label distributions of the auto-generated TORDERs and the TimeBank-Dense TLINKs. We investigate the statistics of Event-Event, Event-Time, and Event-DCT pairs. The TimeBank-Dense corpus is obviously sparser due to the high proportion of *vague* in all three types of pairs. Our TORDERs show a more balanced distribution of labels, which suggests that this method possibly encodes more informative temporal orders compared to the traditional TLINKs. In particular, TORDERs show extremely rare VAGUE labels in Event-DCT pairs. When given the precise *Single-Day* DCT of a document, our proposal to compare the temporal order between the time anchor of a event and the DCT manages to avoid the most unstable judgments made by the human annotators in the Event-DCT pairs. Although the different definition of TORDERs from TLINKs makes direct comparison difficult, the more balanced distribution of TORDERs can possibly provide more informative classification results to benefit the downstream tasks.

[3]https://github.com/racerandom/temporalorder

	Event-DCT		Event-Time		Event-Event	
	F1	**N**	**F1**	**N**	**F1**	**N**
AFTER	0.585	65	0.509	67	0.426	184
BEFORE	0.659	65	0.452	68	0.488	257
INCLUDES	0.400	38	0.136	27	0.158	105
IS_INCLUDED	0	0	0	20	0.077	86
SAME_DAY	0.631	82	0.485	56	0.314	131
SAME_SPAN	0	0	0	2	0	0
PVAGUE	0	0	0	1	0.149	92
VAGUE	0	18	0.417	119	0.487	335
Overall	0.557	268	0.403	360	0.374	1190
Non-VAGUE	0.597	250	0.390	240	0.351	763

(a) TORDER

	Event-DCT		Event-Time		Event-Event	
	F1	**N**	**F1**	**N**	**F1**	**N**
after	0.582	68	0.550	64	0.443	223
before	0.612	58	0.331	91	0.465	326
includes	0.170	22	0.290	31	0.126	45
is_included	0.559	48	0.338	42	0.099	47
simultaneous	0	0	0	6	0	19
vague	0.433	72	0.557	126	0.6116	530
Overall	0.511	268	0.441	360	0.492	1190
Non-*vague*	0.539	196	0.378	234	0.395	660

(b) TLINK

Table 4: The classification results of Event-Event, Event-DCT and Event-Time F1-measure on individual relation types and weighted overall F1. '**N**' denotes the number of the relations in the test split.

4.3 Classification Results

Although the classification results of TORDERs and TLINKs are not directly comparable, they can show some evidence whether TORDERs is functional to provide temporal order information. Table 4 shows the Bi-LSTM classification results with the data split[4](Chambers et al., 2014) (27 training/validation documents, 9 testing documents).

The classification system achieves fairly high F1 0.631 in Event-DCT and 0.485 in Event-Time on the SAME_DAY temporal orders, which are the main information source to predict the precise time of events. The performance on AFTER, BEFORE temporal orders are close to the TLINKs in number, but not meaningfully comparable. The high proportion of *vague* in the TLINKs results in biased predictions. When we use a more meaningful evaluation 'Non-*vague*' overall, the TLINKs performance drops sharply. Generally, the classification results suggest that our proposal of auto-generated TORDERs has sufficient capability to encode temporal information, which can be well classified from the textual inputs.

[4]https://github.com/nchambers/caevo/blob/master/src/main/java/caevo/Evaluate.java

5 Evaluation in Time Anchor Prediction

In this section, we describe a two-step system trained with the existing TLINKs and our data to challenge a downstream time anchor prediction task. The different performance can be seen as the evidence whether our auto-generated TORDERs can capture comparable temporal information to the human-annotated TLINKs.

5.1 Task Definition

Predicting the time of events from the news articles is an attractive goal, which is a necessary step towards automatic event timeline extraction. Reimers et al. (2016) bring the task of time anchor prediction, which aims to predict the time anchor of each *Single-Day* event given a document. They use a general two-step process to determine the event anchors as shown in Figure 3. Given a set of documents with events and time expressions already annotated, the system first obtains a list of possible times for each event. Then, the most precise time is selected for each event.

A serious issue is that their baseline system still depends on the TimeBank-Dense TLINK classifier and the time anchor annotation is only used for the final evaluation. That leaves the space to consider a new method without relying on the human-annotated TLINKs. Our auto-generated TORDERs are a natural alternative to TLINKs to provide the similar temporal order information of mention pairs, but with less annotation efforts. The second-step selection rules just need a slight modification to replace the previous TLINK types with the new TORDER types.

5.2 The Two-step System in Experiments

In this work, we adopt a similar two-step architecture. The first-step temporal order classifier is designed to provide the temporal relations of the mention pairs in a document.

The second-step selects the most precise time by taking all Event-Time and Event-DCT relations of a target event as input. For instance in Figure 3, the second-step received a set of relations e.g. $(is\_included, DCT)$, $(is\_included, Friday)$ and $(vague, January)$ of ***reported***. For the system trained with the TimeBank-Dense TLINKs, we adopt the same selection algorithm as described in (Reimers et al., 2016). When the system is trained

Event Type	Source	TORDER		Gold TORDER		TLINK		Gold TLINK	
		Exact	Partial	Exact	Partial	Exact	Partial	Exact	Partial
Precise	Event-DCT	0.586	0.866	0.739	0.866	0.387	0.570	0.525	0.545
	Event-Time	0.384	0.555	0.577	0.619	0.216	0.288	0.412	0.447
	All	**0.660**	**0.870**	0.835	0.930	0.444	0.611	0.595	0.617
Imprecise	Event-DCT	**0.351**	0.631	0.530	0.647	0.234	0.395	0.364	0.449
	Event-Time	0.074	0.217	0.119	0.184	0.051	0.133	0.200	0.227
	All	0.299	**0.642**	0.509	0.686	0.252	0.429	0.444	0.517
Overall	Event-DCT	0.482	0.762	0.619	0.769	0.319	0.493	0.454	0.503
	Event-Time	0.259	0.419	0.393	0.444	0.149	0.255	0.326	0.358
	All	**0.501**	**0.769**	0.646	0.822	0.360	0.530	0.528	0.573

Table 5: The comparison of the cross-validation performance in the time anchor prediction task. 'Exact' and 'Partial' denote the two evaluation metrics: *exact match* and *partial match* accuracy. 'Gold' denotes the oracle performance of using the gold TORDERs or gold TLINKs as the input of the second-step.

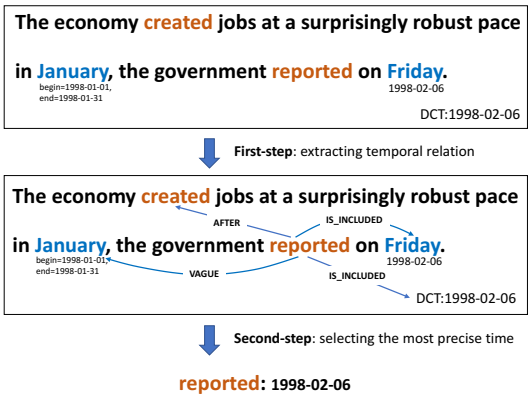


Figure 3: The two-step process to determine the event anchors proposed in (Reimers et al., 2016).

with the TORDERs, we slightly modified the algorithm by replacing the TLINK relations with similar TORDER relations. SAME_DAY replaces *simultaneous* to predict precise dates, although their definition is quite different.

5.3 Experiment Settings

We perform a 6-fold cross-validation strategy to predict all the TORDERs and TLINKs of the mention pairs in the 36 documents of the TimeBank-Dense corpus. In each run, we split 30 documents for training and validation to predict the other 6 test documents.

We define two evaluation metrics, i.e. *Exact Match* accuracy and *Partial Match* accuracy to measure the performance in this task as follows:

$$exact\ match = \frac{\#Number\ of\ the\ exact\ match\ predictions}{\#Total\ number\ of\ the\ test\ samples}$$

$$partial\ match = \frac{\#Number\ of\ the\ partial\ match\ predictions}{\#Total\ number\ of\ the\ test\ samples}$$

We define two *partial match* cases: 1) a precise *(1998-02-06)* is *partial match* with an imprecise *(after 1998-02-06)*, if the date values are the same. 2) *(after 1998-02-06)* is *partial match* with *(after 1998-02-06, before 1998-02-21)*, if one is a part of the other.

5.4 Main Results

Table 5 summarizes the main results of the two-step time anchor prediction system trained with TORDER and TLINK data. 'Precise', 'Imprecise' and 'Overall' denote the results of predicting time anchors of precise events, imprecise events, and overall performance. 'Event-DCT' or 'Event-Time' denotes the second-step selection takes only Event-DCT or Event-Time pairs as input, which helps us to investigate how much information is provided by the different types of pairs for predicting the final time anchors. The new TORDERs show significantly superior out-performance in all three settings (i.e. only Event-DCT pairs, only Event-Time pairs, or Event-DCT + Event-Time), compared to the TLINKs. With both Event-DCT and Event-Time temporal order information, the system achieves the highest overall *exact match* and *partial match* accuracy.

The Event-DCT, Event-Time pairs are the source of temporal information for predicting *time anchors*. The system only using the Event-DCT achieves surprisingly high accuracy, particularly on the TORDER *partial match* accuracy of the

	Exact	Partial
CAEVO	0.442	0.553
Bi-LSTM TLINK	0.437	0.550
Bi-LSTM TORDER	0.586	0.811

Table 6: The comparison to the state-of-the-art dense TLINK classifier

precise events. The reason is that most events reported in news articles usually occur in precisely the same day as DCT. Therefore, the TORDER Event-DCT is benefited from the low proportion of vague relations, which sharply outperforms the TLINK Event-DCT by 16.3% overall *exact match*. However, the contribution of the Event-Time to the overall might be underestimated in this task somehow. The TORDER Event-Time still beats the TLINKs by 11% overall *exact match* and 16.4% overall partial match. Furthermore, the Event-Time encoding the temporal information within 1-sentence window in our experiments can be easily strengthen by our TORDER proposal to introduce more inter-sentence pairs.

5.5 Comparison to a state-of-the-art dense TLINK classifier

In this section, we perform an additional experiment to make a comparison to a system with the first-step replaced by a state-of-the-art dense TLINK classifier CAEVO (Chambers et al., 2014). We adopt the data split setting in Section 4.3 for three classifiers: CAEVO, Bi-LSTM classifier trained with TLINKs and Bi-LSTM classifier trained with TORDERs.

The results are summarized in Table 6. CAEVO achieves the *exact match* accuracy slightly better than the Bi-LSTM model trained with the TLINKs. The Bi-LSTM model trained with the TORDERs sharply outperforms the other two systems by approximate 14% *exact match* accuracy and approximate 26% in *partial match* accuracy.

6 Conclusion

In this paper, we propose a new approach to obtain temporal relations based on time anchors (i.e. absolute time value) of mentions in news articles. Our pre-defined generation rules can automatically induce TORDER relations by comparing the temporal order of two time anchors in a timeline. The requirement of our proposal for annotating time anchors is much easier compared to conventional methods, as the annotation effort is linear with the number of mentions. The TORDER data used in this paper are publicly available. The analysis, empirical comparison and classification results of the new TORDERs and the TimeBank-Dense TLINKs show our new data achieve the low VAGUE proportion, the informative relation types and the balanced label distribution. We perform the second evaluation of using the temporal relation classifier to complete the downstream task of time anchor prediction in news articles. The main results show our TORDERs significantly outperform the TLINKs in this task, which suggests our proposal has the capability to encode informative temporal order information with less annotation effort.

The main limitation of TORDER is that events are required to be anchored in a timeline. Strötgen and Gertz (2016) introduce the highly different characteristics of time expressions in four domains of text. It suggests that our proposal is difficult to be applied in some domains. One possible solution is to adopt a hybrid annotation method to annotate a target event towards the most relevant event (TLINK-style), when temporal information is absent in its context. Although this work is motivated for contributing to timeline applications, evaluating this proposal in the temporal question answering is also valuable. **SAME_DAY** could be harmful because this task possibly requires to know the exact order between two events occurring on the same day. It is worth conceiving a more general solution to improve the limitations of TORDER in the future work.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments and thank Jason Bennett for useful discussions and proofreading.

References

James F Allen. 1990. Maintaining knowledge about temporal intervals. In *Readings in qualitative reasoning about physical systems*, Elsevier, pages 361–372.

Klaus Berberich, Srikanta Bedathur, Omar Alonso, and Gerhard Weikum. 2010. A language modeling approach for temporal information needs. In *European Conference on Information Retrieval*. Springer, pages 13–25.

Philip Bramsen, Pawan Deshpande, Yoong Keok Lee, and Regina Barzilay. 2006. Inducing temporal

graphs. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 189–198.

Taylor Cassidy, Bill McDowell, Nathanael Chambers, and Steven Bethard. 2014. An annotation framework for dense event ordering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 501–506. http://www.aclweb.org/anthology/P14-2082.

Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard. 2014. Dense event ordering with a multi-pass architecture. *Transactions of the Association for Computational Linguistics* 2:273–284. http://aclweb.org/anthology/Q/Q14/Q14-1022.pdf.

Angel X. Chang and Christopher Manning. 2012. Sutime: A library for recognizing and normalizing time expressions. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*. European Language Resources Association (ELRA), Istanbul, Turkey, pages 3735–3740. ACL Anthology Identifier: L12-1122. http://www.lrec-conf.org/proceedings/lrec2012/pdf/284_Paper.pdf.

Fei Cheng and Yusuke Miyao. 2017. Classifying temporal relations by bidirectional lstm over dependency paths. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1–6. http://aclweb.org/anthology/P17-2001.

Quang Xuan Do, Wei Lu, and Dan Roth. 2012. Joint inference for event timeline construction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, pages 677–687.

Jennifer D'Souza and Vincent Ng. 2013. Classifying temporal relations with rich linguistic knowledge. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Atlanta, Georgia, pages 918–927. http://www.aclweb.org/anthology/N13-1112.

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

Oleksandr Kolomiyets, Steven Bethard, and Marie-Francine Moens. 2012. Extracting narrative timelines as temporal dependency structures. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics, pages 88–97.

Natsuda Laokulrat, Makoto Miwa, Yoshimasa Tsuruoka, and Takashi Chikayama. 2013. Uttime: Temporal relation classification using deep syntactic features. In *Second Joint Conference on Lexical and Computational Semantics (* SEM)*. volume 2, pages 88–92. http://aclweb.org/anthology/S/S13/S13-2015.pdf.

Hector Llorens, Nathanael Chambers, Naushad UzZaman, Nasrin Mostafazadeh, James Allen, and James Pustejovsky. 2015. Semeval-2015 task 5: Qa tempeval - evaluating temporal information understanding with question answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Association for Computational Linguistics, Denver, Colorado, pages 792–800. http://www.aclweb.org/anthology/S15-2134.

Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. 2006. Machine learning of temporal relations. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Sydney, Australia, pages 753–760. https://doi.org/10.3115/1220175.1220270.

Yuanliang Meng, Anna Rumshisky, and Alexey Romanov. 2017. Temporal information extraction for question answering using syntactic dependencies in an lstm-based architecture. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 887–896. https://www.aclweb.org/anthology/D17-1092.

Anne-Lyse Minard, Manuela Speranza, Eneko Agirre, Itziar Aldabe, Marieke van Erp, Bernardo Magnini, German Rigau, and Ruben Urizar. 2015. Semeval-2015 task 4: Timeline: Cross-document event ordering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Association for Computational Linguistics, Denver, Colorado, pages 778–786. http://www.aclweb.org/anthology/S15-2132.

Paramita Mirza and Sara Tonelli. 2016. On the contribution of word embeddings to temporal relation classification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 2818–2828. http://aclweb.org/anthology/C16-1265.

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, et al. 2003. The timebank corpus. In *Corpus linguistics*. volume 2003, page 40. https://catalog.ldc.upenn.edu/LDC2006T08.

Nils Reimers, Nazanin Dehghani, and Iryna Gurevych. 2016. Temporal anchoring of events for the timebank corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 2195–2204. http://www.aclweb.org/anthology/P16-1207.

Roser Saurı, Jessica Littman, Bob Knippen, Robert Gaizauskas, Andrea Setzer, and James Pustejovsky. 2006. Timeml annotation guidelines version 1.2. 1.

Andrea Setzer. 2002. *Temporal information in newswire articles: an annotation scheme and corpus study.*. Ph.D. thesis, University of Sheffield.

Jannik Strötgen and Michael Gertz. 2015. A baseline temporal tagger for all languages. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 541–547. http://aclweb.org/anthology/D15-1063.

Jannik Strötgen and Michael Gertz. 2016. Domain-sensitive temporal tagging. *Synthesis Lectures on Human Language Technologies* 9(3):1–151.

Naushad UzZaman, Hector Llorens, James Allen, Leon Derczynski, Marc Verhagen, and James Pustejovsky. 2012. Tempeval-3: Evaluating events, time expressions, and temporal relations. *arXiv preprint arXiv:1206.5333* http://aclweb.org/anthology/S/S13/S13-2001.pdf.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Jessica Moszkowicz, and James Pustejovsky. 2009. The tempeval challenge: identifying temporal relations in text. *Language Resources and Evaluation* 43(2):161–179. https://doi.org/10.1007/s10579-009-9086-z.

Marc Verhagen, Roser Sauri, Tommaso Caselli, and James Pustejovsky. 2010. Semeval-2010 task 13: Tempeval-2. In *Proceedings of the 5th international workshop on semantic evaluation*. Association for Computational Linguistics, pages 57–62. http://www.aclweb.org/anthology/S10-1010.

ELDEN: Improved Entity Linking using Densified Knowledge Graphs

Priya Radhakrishnan[+], Partha Talukdar[*] and Vasudeva Varma[+]

[+]IIIT Hyderabad, India
[*]Indian Institute of Science, Bangalore, India
priya.r@research.iiit.ac.in, ppt@iisc.ac.in, vv@iiit.ac.in

Abstract

Entity Linking (EL) systems aim to automatically map mentions of an entity in text to the corresponding entity in a Knowledge Graph (KG). Degree of connectivity of an entity in the KG directly affects an EL system's ability to correctly link mentions in text to the entity in KG. This causes many EL systems to perform well for entities well connected to other entities in KG, bringing into focus the role of KG density in EL. In this paper, we propose Entity Linking using Densified Knowledge Graphs (ELDEN). ELDEN is an EL system which first densifies the KG with co-occurrence statistics from a large text corpus, and then uses the densified KG to train entity embeddings. Entity similarity measured using these trained entity embeddings result in improved EL. ELDEN outperforms state-of-the-art EL system on benchmark datasets. Due to such densification, ELDEN performs well for sparsely connected entities in the KG too. ELDEN's approach is simple, yet effective. We have made ELDEN's code and data publicly available.

1 Introduction

Entity Linking (EL) is the task of mapping *mentions* of an *entity* in text to the corresponding entity in Knowledge Graph (KG) (Hoffart et al., 2011; Dong et al., 2014; Chisholm and Hachey, 2015). EL systems primarily exploit two types of information: (1) similarity of the mention to the candidate entity string, and (2) coherence between the candidate entity and other entities mentioned in the vicinity of the mention in text. Coherence essentially measures how well the candidate entity is connected, either directly or indirectly, with other KG entities mentioned in the vicinity (Milne and Witten, 2008; Globerson et al., 2016). In the state-of-the-art EL system by (Yamada et al., 2016), coherence is measured as distance between embeddings of entities. This system performs well on entities which are densely-connected in KG, but not so well on sparsely-connected entities in the KG.

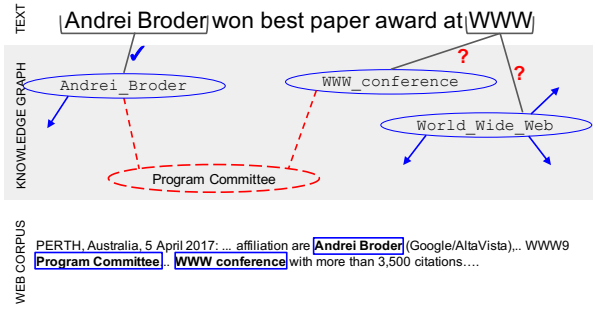


Figure 1: Improving entity disambiguation performance using KG densification: edges to `WWW_conference`, a sparsely-connected entity in the KG, is increased by adding edges from *pseudo entity* `Program Committee` whose mention co-occurs with it in web corpus. ELDEN, the system proposed in this paper, uses such densified KG to successfully link ambiguous mention *WWW* to the correct entity `WWW_conference`, instead of the more popular entity `World Wide Web`.

We demonstrate this problem using the example sentence in Figure 1. This sentence has two mentions: *Andrei Broder* and *WWW*. The figure also shows mention-entity linkages, i.e., mentions and their candidate entities in KG. Using a conventional EL system, the first mention *Andrei Broder*[1] can be easily linked to `Andrei_Broder` using string similarity between the mention and candidate entity strings. String similarity works well in this case as this mention is unambiguous in the given setting. However, the second mention

[1]We use *italics* to denote textual mentions and `typewriter` to indicate an entity in KG.

Proceedings of NAACL-HLT 2018, pages 1844–1853
New Orleans, Louisiana, June 1 - 6, 2018.

WWW has two candidates, World_Wide_Web and WWW_conference, and hence is ambiguous. In such cases, coherence measure between the candidate entity and other unambiguously linked entity(ies) is used for disambiguation.

State-of-the-art EL systems measure coherence as similarity between embeddings of entities. The entity embeddings are trained based on the number of common edges in KG[2]. In our example, common edges are edges World_Wide_Web shares with Andrei_Broder and edges WWW_conference shares with Andrei_Broder. But WWW_conference has less number of edges (it is a sparsely-connected entity) compared to World_Wide_Web. This leads to poor performance[3] whereby *WWW* is erroneously linked to World_Wide_Web instead of linking to WWW_conference.

In this paper, we propose ELDEN, an EL system which increases nodes and edges of the KG by using information available on the web about entities and *pseudo entities*. Pseudo Entities are words and phrases that frequently occur in Wikipedia, and co-occur with mentions of KG entities in the web corpus. Thus ELDEN uses a web corpus to find pseudo entities and refines the co-occurrences with Pointwise Mutual Information (PMI) (Church and Hanks, 1989) measure. ELDEN then adds edges to the entity from pseudo entities. In Figure 1, pseudo entity *Program Committee* co-occurs with mentions of Andrei_Broder and WWW_conference in web corpus and has a positive PMI value with both. So ELDEN adds edges from *Program Committee* to Andrei_Broder and WWW_conference, densifying neighborhood of the entities. Coherence, now measured as similarity between entity embeddings where embeddings are trained on densified KG, leads to improved EL performance.

Density (number of KG edges) of candidate entity affects EL performance. In our analysis of density and number of entities having that density in the Wikipedia KG, we find that entities with 500 edges or less make up more than 90%. Thus, creating an EL system that performs well on densely as well as sparsely-connected entities is a challenging, yet unavoidable problem.

We make the following contributions:

- ELDEN presents a simple yet effective graph densification method which may be applied to improve EL involving any KG.
- By using pseudo entities and unambiguous mentions of entity in a corpus, we demonstrate how non-entity-linked corpus can be used to improve EL performance.
- We have made ELDEN's code and data publicly available[4].

2 Related Work

Entity linking: Most EL systems use coherence among entities (Cheng and Roth, 2013) to link mentions. We studied coherence measures and datasets used in six recent[5] EL systems (He et al., 2013; Huang et al., 2015; Sun et al., 2015; Yamada et al., 2016; Globerson et al., 2016; Barrena et al., 2016). We see that the two popular datasets used for evaluating EL (Chisholm and Hachey, 2015) on documents are CoNLL (Hoffart et al., 2011) and TAC2010 (Ji et al., 2010), here after TAC. The popular coherence measures used are (1) WLM, (2) Entity Embedding Similarity and (3) Jaccard Similarity (Chisholm and Hachey, 2015; Guo et al., 2013). WLM is widely acknowledged as most popular (Hoffart et al., 2012) with almost all the six approaches analyzed above using WLM or its variants. Entity embedding similarity (Yamada et al., 2016) is reported to give highest EL[6] performance and is the baseline of ELDEN.

Enhancing entity disambiguation: Among methods proposed in literature to enhance entity disambiguation utilizing KG (Bhattacharya and Getoor) uses additional relational information between database references; (Han and Zhao, 2010) uses semantic relatedness between entities in other KGs; and (Shen et al., 2018) uses paths consisting of defined relations between entities in the KG (IMDB and DBLP). All these methods utilize structured information, while our method shows how unstructured data (web corpus about the entity to be linked) can be effectively used for entity disambiguation.

Entity Embeddings: ELDEN presents a method to enhance embedding of entities and words in a

[2]Wikipedia Link based Measure (WLM) (Milne and Witten, 2008) used in Yamada et al.'s system is based on number of common edges in KG.

[3]This paper focuses on mention disambiguation. We assume mention and candidate entities are detected already.

[4]https://github.com/priyaradhakrishnan0/ELDEN

[5] (Shen et al., 2015) presents a survey of EL systems.

[6]Named Entity Disambiguation (NED) and EL are synonymous terms in research (Hoffart et al., 2011)

common vector space. Word embedding methods like word2vec (Mikolov et al., 2013) and Glove (Pennington et al., 2014) have been extended to entities in EL (Yamada et al., 2016; Fang et al., 2016; Zwicklbauer et al., 2016; Huang et al., 2015). These methods use data about entity-entity co-occurrences to improve the entity embeddings. In ELDEN, we improve it with web corpus co-occurrence statistics. Ganea and Hofmann (2017) present a very interesting neural model for jointly learning entity embedding along with mentions and contexts.

KG densification with pseudo entities: KG densification using external corpus has been studied by Kotnis *et al.* (2015) and Hegde and Talukdar (2015). Densifying edge graph is also studied as 'link prediction' in literature (Martínez et al., 2016). Kotnis *et al.* augment paths between KG nodes using 'bridging entities' which are noun phrases mined from an external corpus. ELDEN has a similar approach as it proposes densifying the KG edges of entities by adding edges from pseudo entities. However, densification is used for relation inference in the former methods whereas it is used for entity coherence measurement in ELDEN.

Word co-occurrence measures: Chaudhari *et al.* (2011) survey several co-occurrence measures for word association including PMI, Jaccard (Dice, 1945) and Co-occurrence Significance Ratio (CSR). Damani (2013) proves that considering corpus level significant co-occurrences, PMI is better than others. Budiu *et al.* (2007) compare Latent Semantic Analysis (LSA), PMI and Generalized Latent Semantic Analysis (GLSA) and conclude that for large corpora like web corpus, PMI works best on word similarity tests. Hence, we chose PMI to refine co-occurring mentions of entities in web corpus.

3 Definitions and Problem Formulation

In this section, we present a few definitions and formulate the EL problem.

Knowledge Graph (KG): A Knowledge Graph is defined as $G = (E, F)$ with entities E as nodes and F as edges. In allegiance to EL literature and baselines (Milne and Witten, 2008; Globerson et al., 2016), we use the Wikipedia hyperlink graph as the KG in this paper, where nodes correspond to Wikipedia articles and edges are incoming links from one Wikipedia article to another. ELDEN ultimately uses a densified version of this Wikipedia KG, as described in Section 4.

Sparsely connected entities: Following Hoffart *et al.* (2012), we define an entity to be a sparsely connected entity, if the number of edges incident on the entity node in the KG is less than threshold η[7]. Otherwise, the entity is called a densely-connected entity.

Entity Linking (EL): *Given a set of mentions $M_D = \{m_1, ..., m_n\}$ in a document D, and a knowledge graph $G = (E, F)$, the problem of entity linking is to find the assignment $\Lambda : M_D \rightarrow E_D$, where E_D is the set of entities linked to mentions in document D such that $E_D \subseteq E$.*

For mention $m_i \in M_D$, let the set of possible entities it can link to (candidate entities) be $\mathbb{C}_i$. Then, the solution to the EL problem is an assignment Λ where,

$$\Lambda(m_i) = \arg\max_{e \in \mathbb{C}_i} [\phi(m_i, e) + \beta \cdot \psi(e, E_D)] \quad (1)$$

Here, $\phi(m_i, e) \in [0, 1]$ measures the contextual compatibility of mention m_i and entity e. $\phi(m_i, e)$ is obtained by combining prior probability and context similarity. $\psi(e, E_D)$ measures the coherence of e with entities in E_D. β is a variable controlling inclusion of ψ in the assignment Λ.

Problem Formulation : (Yamada et al., 2016) is a recently proposed state-of-the-art EL system. We consider it as a representative EL system and use it as the main baseline for the experiments in this paper. In this section, we briefly describe Yamada *et al.* (2016)'s two-step approach that solves the EL problem presented above.

Step 1: A mention $m_i \in M_D$ is defined to be *unambiguous* if $\exists e \in \mathbb{C}_i$ such that $\phi(m_i, e) \geq \gamma$. Let $M_D^{(u)} \subseteq M_D$ be the set of such unambiguous mentions in document D. For all unambiguous mentions $m \in M_D^{(u)}$, Yamada *et al.* freeze the assignment by solving Equation 1 after setting $\beta = 0$. In other words, $\psi(e, E_D)$ is not used while assigning entities to unambiguous mentions. Assigning entities first to unambiguous mentions has also been found to be helpful in prior research (Milne and Witten, 2008; Guo and Barbosa, 2014). Let A_D be the set of entities linked to in this step. In Figure 1, mention *Andrei Broder* is unambiguous[8].

[7] Like (Hoffart et al., 2012) we set $\eta = 500$ for the experiments in this paper.

[8] Between mention *Andrei Broder* and entity `Andrei_Broder`, string similarity and prior probability are 1.0. In the experiments we use a γ value of 0.95.

Notation	Definition
v_e	entity embedding of entity e with dimension $1 * d$
E	set of all titles in Wikipedia
S	set of all pseudo entities considered in ELDEN
E_+	set of all entities considered in ELDEN
$\mathbf{V}$	word vectors of E_+ where $\mathbf{V} \in \mathbb{R}^{k*d}$
ψ_{ELDEN}	Embedding similarity measured using $\mathbf{V}$ trained on G_{dense}
ψ_{Yamada}	Embedding similarity measured using $\mathbf{V}$ trained on input KG G

Table 1: Notation used in KG densification and learning entity embeddings (Please see Sec 4 for more details).

Step 2: In this step, Yamada *et al.* links all ambiguous mentions by solving Equation 2.

$$\Lambda(m_i) = \arg\max_{e \in \mathbb{C}_i} \left(\phi(m_i, e) + \frac{1}{|A_D|} \sum_{e_j \in A_D} v_e \cdot v_{e_j} \right) \quad \forall m_i \in M_D \setminus M_D^{(u)} \quad (2)$$

where $v_e, v_{e_j} \in \mathbb{R}^d$ are d-dimensional embeddings of entities e and e_j respectively. Please note that the equation above is a reformulation of Equation 1 with $\beta = 1$ and $\psi(e, E_D) = \frac{1}{|A_D|} \sum_{e_j \in A_D} v_e \cdot v_{e_j}$, where A_D is derived from E_D as described in Step 1. As coherence $\psi(e, E_D)$ is applied only in disambiguation of ambiguous mentions, we apply densification to only selective nodes of KG.

Embeddings of entities are generated using word2vec model and trained using WLM (Details in Section 5). Given a graph $G = (E, F)$, the WLM coherence measure $\psi_{wlm}(e_i, e_j)$ between two entities e_i and e_j is defined in Equation 3 where C_e is the set of entities with edge to entity e.

$$\psi_{\text{wlm}}(e_i, e_j) = 1 - \frac{log(max(|C_{e_i}|, |C_{e_j}|)) - log(|C_{e_i} \cap C_{e_j}|)|}{log(|E|) - log(min(|C_{e_i}|, |C_{e_j}|))} \quad (3)$$

4 Our Approach: ELDEN

In this section, we present ELDEN, our proposed approach. ELDEN extends (Yamada et al., 2016), with one important difference: rather than working with the input KG directly, ELDEN works with the densified KG, created by selective densification of the KG with co-occurrence statistics extracted from a large corpus. Even though this is a simple change, this results in improved EL performance. The method performs well even for sparsely connected entities.

Overview : Overview of the ELDEN system is shown in Figure 2. ELDEN starts off with densification of the input KG, using statistics from web corpus. Embeddings of entities are then learned utilizing the densified KG in the next step. Embedding similarity estimated using the learned entity embeddings is used in calculating coherence measure in subsequent EL. Notation used is summarized in Table 1.

***(i)*KG Densification** Figure 2 depicts densification of KG in 'Input KG' and 'Densified KG'. It shows two Wikipedia titles `Andrei_Broder` and `WWW_conference` from our running example (Figure 1). There are no edges common between `Andrei_Broder` and `WWW_conference`. In a web corpus, mentions of `Andrei_Broder` and `WWW_conference` co-occur with *Program committee* and it has a positive PMI value with both the entities. So ELDEN adds an edge from *Program committee* to both the entities. Here *Program committee* is a pseudo entity. Thus, ELDEN densifies the KG by adding edges from pseudo entities when the mentions of Wikipedia entity and pseudo entity co-occur in a web corpus and the pseudo entity has a positive PMI value with given entity.

Taking a closer look, KG densification process starts from 'input KG' which is Wikipedia hyperlink graph $G = (E, F)$, where the nodes are Wikipedia titles (E) and edges are hyperlinks (F). ELDEN processes Wikipedia text corpus and identifies phrases (unigrams and bi-grams) that occur frequently, i.e. more than 10 times in it. We denote these phrases as *pseudo entities* (S) and add them as nodes to the KG. Let $E_+ = E \cup S$ be the resulting set of nodes.

ELDEN then adds edges connecting entities in E_+ to entities in E. This is done by processing a web text corpus looking for mentions of entities in E_+, and linking the mentions to entities in KG $G' = (E_+, F)$. ELDEN uses Equation 1 with $\beta = 0$ for this entity linking, i.e. only mention-entity similarity $\phi(m, e)$ is used during this linking[9]. Based on this entity linked corpus, a co-occurrence matrix M of size $|E_+| \times |E_+|$ is constructed. Each cell $M_{i,j}$ is set to the PMI between

[9]Since prior probabilities of pseudo entities are not available, only mention-entity similarity component of $\phi(m, s)$ is used while linking a mention m to a pseudo entity $s \in S$.

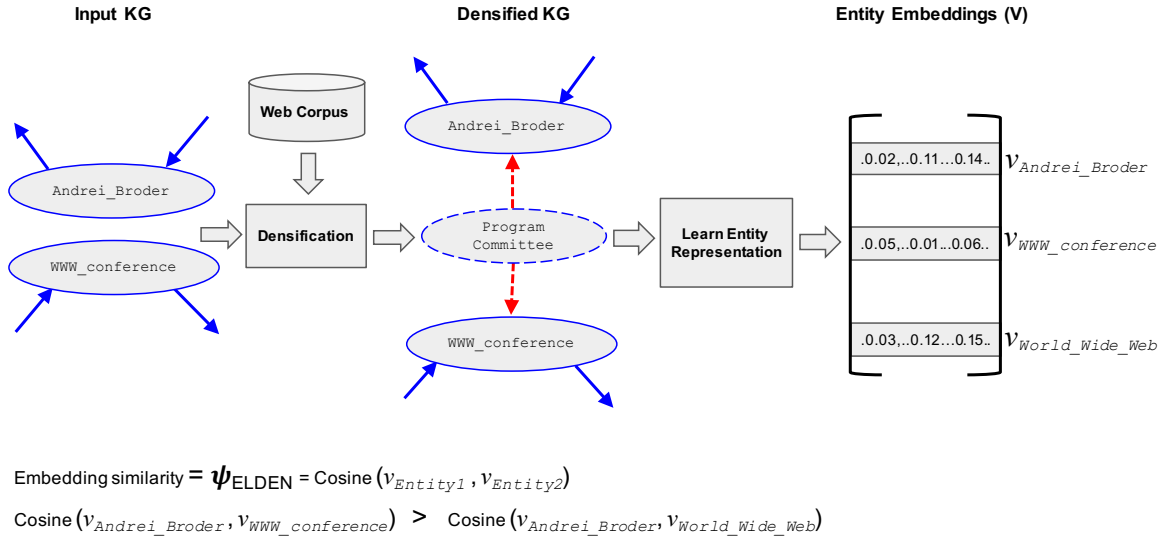


Figure 2: ELDEN consists of KG densification, training entity embeddings on densified KG and building EL system that uses similarity between the trained embeddings as coherence measure. ELDEN's difference from baseline method is that while baseline method uses input KG for training v_e, ELDEN uses densified KG for training v_e. Hence, the improved performance of ELDEN is solely attributed to densification.

Objective function component	Feature Group	Feature	Explanation
Contextual compatibility $\phi(m_i, e_i)$	Base	Entity Prior	Fraction of edges that links to the entity
		Prior Probability	Of all possible pages a mention can link to, the probability that it links to a given page
		Max Prior Probability	Maximum value of prior probabilities for a mention
		Max Prior Probability in document	Maximum value of prior probabilities for a document
		Num of candidates	Candidate count of the mention
	String Similarity	Exact match	Whether mention is an exact match of candidate
		Partial match	Whether mention is part of candidate
Coherence $\psi(e_i, e_j)$	Coherence measures	ψ_{wlm}	WLM (Please see Equation 3)
		ψ_{Yamada}	Similarity between entity embeddings that are learned using ψ_{wlm}
		ψ_{dense}	ψ_{wlm} calculated on densified KB
		ψ_{ELDEN}	Similarity between entity embeddings that are trained using ψ_{dense}
		$\psi_{\text{ELDEN++}}$	Combination of ψ_{wlm}, ψ_{Yamada}, ψ_{dense}, and ψ_{ELDEN}. ELDEN uses this coherence feature.

Table 2: Features used by various EL systems. Context compatibility $\phi(m_i, e_i)$ and coherence $\psi(e_i, e_j)$ of Equation 1 are parameterized using features as shown above. Context compatibility features are used by all systems. Coherence features used by Yamada16 are ψ_{wlm} and ψ_{Yamada}. ELDEN uses $\psi_{\text{ELDEN++}}$ as coherence feature.

the entities e and e'.

$$M_{e,e'} = \text{PMI}(e, e') = \log \frac{f(e, e') \times N}{f(e) \times f(e')}$$

where $f(e)$ is the frequency of entity e in web corpus, $f(e, e')$ is the sentence-constrained pair frequency of the entity pair (e, e') in web corpus, and $N = \sum_{e,e' \in E_+} f(e, e')$. Please note that PMI, and there by M, are symmetric. The expanded set of edges, F_+, is now defined as

$$F_+ = F \cup \{(e, e'), (e', e) \mid e' \in E_+, e \in E, M_{e,e'} > 0\}$$

In other words, we augment the set of initial edges F with additional edges connecting entities in E_+ with entities in E such that PMI between the entities is positive.

ELDEN now constructs the KG $G_{\text{dense}} = (E_+, F_+)$, which is a densified version of the input KG $G = (E, F)$. ELDEN uses this densified KG G_{dense} for subsequent processing and entity linking.

***(ii)*Learning Embeddings of Densified KG Entities** ELDEN derives entity embeddings using the same setup, corpus and Word2vec skip-gram with negative sampling model as in Yamada *et al.*, However, instead of training embeddings over the input KG, ELDEN trains embeddings of entities in the densified KG G_{dense}. Let $\mathbf{V}$ be the word2vec matrix containing embeddings of entities in E_+ where $\mathbf{V} \in \mathbb{R}^{k*d}$. v_e is the embedding of entity e in E_+ with dimension $1 * d$.

In word2vec model, entities in context are used

to predict the target entity. ELDEN maximizes the objective function (Goldberg and Levy, 2014) of word2vec skip-gram model with negative sampling, $L = \sum_{(t,c)\in\mathbb{P}} L_{t,c}$ where

$$L_{t,c} = log\ \theta(v_c \cdot v_t) + \sum_{n\in\mathbb{N}_{(t,c)}} log\ \theta(-v_n \cdot v_t)$$

Here v_t and v_c are the entity embeddings of target entity t and context entity c. $\mathbb{P}$ is the set of target-context entity pairs considered by the model. $\mathbb{N}_{(t,c)}$ is a set of randomly sampled entities used as negative samples with pair (t, c). This objective is maximized with respect to variables v_t's and v_c's, where $\theta(x) = \frac{1}{1+e^{-x}}$.

$\mathbb{P}$ and $\mathbb{N}$ are derived using G_{dense}. t and c are entities in E_+ such that c shares a common edge with t. v_n is randomly sampled from $\mathbf{V}$, for entities that do not share a common edge with t. Entity embedding similarity measured using $\mathbf{V}$ trained this way on G_{dense} is ψ_{ELDEN}. Embedding similarity is measured as cosine distance between v_es. Embeddings of S are trained using positive and negative word contexts derived using context length.

***(iii)* Bringing it All Together: ELDEN**

ELDEN is a supervised EL system which uses two sets of features: (1) contextual compatibility $\phi(m, e)$; and (2) coherence $\psi(e_i, e_j)$. These features are summarized in Table 2. Similarity between entity embeddings is measured as cosine similarity between v_es.

5 Experiments

In this section, we evaluate the following:

- Is ELDEN's corpus co-occurrence statistics-based densification helpful in disambiguating entities better? (Sec. 6.1)
- Where does ELDEN's selective densification of KG nodes link entities better? (Sec. 6.3)

Setup : ELDEN is implemented using Random Forest ensemble [10] (Breiman, 1998). Parameter values were set using CoNLL development set. Feature limit of 3 with number of estimators as 100 yielded best performance.

Knowledge Graph: Wikipedia Following prior EL literature, we use Wikipedia hypergraph as our KG (Milne and Witten, 2008; Globerson et al., 2016). This KG is enhanced with pseudo entities as explained in Section 4. We process the Wikipedia corpus following the same procedure as in Yamada *et al.* (2016). We cleaned [11] Wikipedia dump dated Nov 2015. We then parsed the Wikipedia article text to identify pseudo entities. More details on KG and parameters used for training embeddings are in Table 3. Training took 4 days on gpu with 2 cores.

Specifics	Value
Number of titles after cleaning ($\|E\|$)	4.6 M
Number of pseudo entities ($\|S\|$)	1.3 M
Total number of entities ($\|E_+\|$)	5.9 M
Number of epochs	3
Learning rate	0.25 linearly reducing to 0.01
Number of negative samples ($\|\mathbb{N}\|$)	5
Context window size	3
Dimensions of embedding (d)	100

Table 3: Parameters used in Wikipedia processing and training KG.

Preprocessing: Web corpus and Densified KG For our experiments, we created a web corpus by querying Google[12]. Candidate entities of all mentions in the dataset are queried in Google and top ten search results are considered for unigram and bigram frequencies. This corpus occupied 6.8GB for candidate entities of TAC and CoNLL dataset mentions (54336 entities). Even for sparsely connected entities, an average corpus size of 670 lines or more[13] was collected. We note that though some of the entities mentioned in this dataset are ten or more years old, we are able to collect, on an average more than 670 lines of web content. Thus corpus proves to be a good source of additional links for densification, for both common and rare entities. As Taneva and Weikum (2013) also note, it is not hard to find content about sparsely connected entities on the web.

The web corpus is analyzed for mentions and pseudo entities. Co-occurrence matrix M is created[14] for mention and pseudo entities occurring within window of size 10 for PMI calculation[15]. Edges are added from pseudo entities with positive PMI to mention of given entity. In experiments we add edges from top 10 pseudo entities ordered by

[10] http://scikit-learn.org

[11] by removing disambiguation, navigation, maintenance and discussion pages.

[12] https://www.google.com/

[13] A detailed analysis of knowledge gained from crawling for common versus less common entities is present in Figure 1 of supplementary material.

[14] This co-occurrence matrix is downloadable with source code.

[15] We experimented with window sizes 10, 25 and 50. We chose 10 that gave best results

PMI values[16].

Evaluation Dataset: In line with prior work on EL, we test the performance of ELDEN on CoNLL and TAC datasets. As this paper focuses on entity disambiguation, we tested ELDEN against datasets and baseline methods for disambiguation. We note that the entity disambiguation evaluation part of other recent datasets like ERD 2014 and TAC 2015 is exactly same as the TAC 2010 evaluation (Ellis et al., 2014) [17].

Training: ELDEN's parameters were tuned using training (development) sets of CoNLL and TAC datasets. CoNLL and TAC datasets consist of documents where mentions are marked and entity to which the mention links to, is specified. We use only mentions that link to a valid Wikipedia title (non NIL entities) and report performance on test set. Some aspects of these datasets relevant to our experiments are provided below.

CoNLL: In CoNLL test set (5267 mentions), we report Precision of topmost candidate entity, aggregated over all mentions (P-micro) and aggregated over all documents (P-macro), i.e., if tp, fp and p are the individual true positives, false positives and precision for each document in a dataset of δ documents, then

$$P_{\text{micro}} = \frac{\sum_{i=1}^{\delta} \text{tp}_i}{\sum_{i=1}^{\delta} \text{tp}_i + \sum_{i=1}^{\delta} \text{fp}_i} \text{ and } P_{\text{macro}} = \frac{\sum_{i=1}^{\delta} \text{p}_i}{\delta}.$$

For CoNLL candidate entities, we use (Pershina et al., 2015) dataset[18].

TAC: In TAC dataset, we report P-micro of top-ranked candidate entity on 1,020 mentions. P-macro is not applicable to TAC as most documents have only one mention as query mention (or 'mention to be linked'). For TAC candidate entities, we index the Wikipedia word tokens and titles using solr[19]. We index terms in (1) title of the entity, (2) title of another entity redirecting to the entity, and (3) names of anchors that point to the entity, in line with baselines. We are making this TAC candidate set publicly available.

Baseline: Yamada16 Our baseline is the Yamada *et al.* system explained in Section 3. Entity embedding distance measured using v_e trained on the input KG G is ψ_{Yamada}.

[16]This is a tunable parameter.

[17]These recent datasets consist of other evaluations, e.g., mention detection, multilinguality etc. which is beyond the scope of the paper and hence we didnt focus on them in the paper.

[18]https://github.com/masha-p/PPRforNED

[19]http://lucene.apache.org/solr/

Method	CONLL (P-micro)	CONLL (P-macro)	TAC (P-micro)
(Hoffart et al., 2011)	82.5	81.7	-
(He et al., 2013)	85.6	84.0	81.0
(Ling et al., 2015)	67.5	-	86.8
(Barrena et al., 2016)	88.32	-	-
(Chisholm and Hachey, 2015)	88.7	-	80.7
(Pershina et al., 2015)	91.8	89.9	-
(Globerson et al., 2016)	91.7	-	87.2
(Yamada et al., 2016)	**93.1**	92.6	85.2
ELDEN	93.0	**93.7**	**89.6**

Table 4: Performance comparison with other recent EL approaches. ELDEN matches best results in CoNLL and outperforms the state-of-the-art in TAC dataset. (Please see Section 6.1 for details and $\psi_{\text{ELDEN++}}$ row of Table 5 for ELDEN results.)

6 Results

6.1 Does ELDEN's selective densification help in disambiguation in EL?

In Table 4, we compare ELDEN's EL performance with results of other recently proposed state-of-the-art EL methods that use coherence models. We see that ELDEN results matches best results on CoNLL and outperforms state-of-the-art in TAC dataset. In the table, the last four rows uses the Pershina *et al.* (2015) candidate set and hence, we provide a comparison of their disambiguation performance. Improved results of ELDEN over baseline is attributed to the improved disambiguation due to KG densification.

6.2 Why does ELDEN's selective densification work?

We conduct ablation analysis using various feature and feature combinations and present performance of ELDEN and baseline in Table 5. Starting with base features, we add various features to ELDEN incrementally and report their impact on performance. The results when using base feature group alone, and base and string similarity groups together (ϕ) are presented in first and second rows for each dataset. We compare ψ_{ELDEN} to three coherence measures: ψ_{wlm}, ψ_{Yamada} and ψ_{dense}, details of which are provided in Table 2. The performance improvement from each of the four coherence measures are in the next four rows. Performance of ELDEN from using all four coherence features is given in $\psi_{\text{ELDEN++}}$ row.

On CoNLL dataset, ψ_{dense} combined with ϕ,

Dataset	Features	P-micro	P-macro
CoNLL	Base	87.0	88.3
	ϕ	90.0	91.2
	$\phi + \psi_{\text{wlm}}$	91.0*	91.8
	$\phi + \psi_{\text{Yamada}}$	90.0*	91.1
	$\phi + \psi_{\text{dense}}$	92.0*	93.0
	$\phi + \psi_{\text{ELDEN}}$	91.0*	91.2
	$\phi + \psi_{\text{wlm}} + \psi_{\text{Yamada}}$	91.0	91.8
	$\phi + \psi_{\text{dense}} + \psi_{\text{ELDEN}}$	91.0	92.6
	$\phi + \psi_{\text{ELDEN++}}$	**93.0**	**93.7**
TAC	Base	78.2	
	ϕ	80.2	
	$\phi + \psi_{\text{wlm}}$	82.5*	
	$\phi + \psi_{\text{Yamada}}$	83.1*	
	$\phi + \psi_{\text{dense}}$	85.5*	
	$\phi + \psi_{\text{ELDEN}}$	87.3*	
	$\phi + \psi_{\text{wlm}} + \psi_{\text{Yamada}}$	84.4	
	$\phi + \psi_{\text{dense}} + \psi_{\text{ELDEN}}$	88.7	
	$\phi + \psi_{\text{ELDEN++}}$	**89.6**	

Table 5: Ablation analysis involving various coherence measures (see Table 2 for definitions of these measures). Statistically significant improvements over ϕ are marked with an asterisk. ELDEN's coherence measure, $\psi_{\text{ELDEN++}}$, achieves the best overall performance. P-macro is not applicable to TAC as most documents have only one mention marked as query. (Please see Section 6.2).

gave an improvement of 2.0 and 1.9 (P-micro and P-macro) over Yamada16 results. We note that Yamada16 results are from our re-implementation of (Yamada et al., 2016) system [20] and we are able to almost reproduce the baseline results. We also present the results combining baseline's ψ_{Yamada} and ψ_{wlm} versus ELDEN's ψ_{ELDEN} and ψ_{dense} in next two rows. We find the ELDEN's KG densification features perform better than baselines.

On TAC dataset also, combined with ϕ, ψ_{dense} is found to do better than ψ_{wlm} and ψ_{ELDEN} gives a significant P-micro improvement of 4.2 over ψ_{Yamada}. The $\psi_{\text{ELDEN++}}$ P-micro in TAC dataset is statistically significant[21]. In short, we find the KG densification features, ψ_{dense} and ψ_{ELDEN}, as the features causing better performance of ELDEN on both datasets.

6.3 Where does ELDEN's selective densification work better?

While most EL systems give higher precision on CoNLL dataset than TAC dataset, ELDEN performs with high precision on TAC dataset too.

[20]We have re-implemented the Yamada *et al* system using hyper-parameters specified in the paper and these are our best-effort results.

[21]We performed two tailed t-test, with 2-tail 95% value of 1.96.

	% sparsely connected Entities	
Dataset	**Train**	**Test**
TAC	78.8	63.6
CoNLL	48.4	48.2

Table 6: Percentage of sparsely connected entities in evaluation datasets. TAC has higher composition of sparsely connected entities than CoNLL. Hence, ELDEN results are better in TAC over CoNLL (Please see Table 4).

This is explained by analyzing distribution of densely-connected and sparsely connected entities in TAC and CoNLL datasets as presented in Table 6. We see that CoNLL test set has almost half as densely-connected and half as sparsely connected entities, whereas in TAC test set, 63.6% are sparsely connected entities. This higher constitution of sparsely connected entities in TAC, explains ELDEN's better results in TAC relative to CoNLL dataset. As the number of sparsely connected entities is more than the number of densely-connected entities in most KGs (Reinanda et al., 2016), our method is expected to be of significance for most KGs.

6.4 What type of EL errors are best fixed with ELDEN's selective densification ?

We analyzed errors fixed by ELDEN on TAC dataset. We categorize the errors into four classes in line with error classes of Ling *et al.* (2015). We

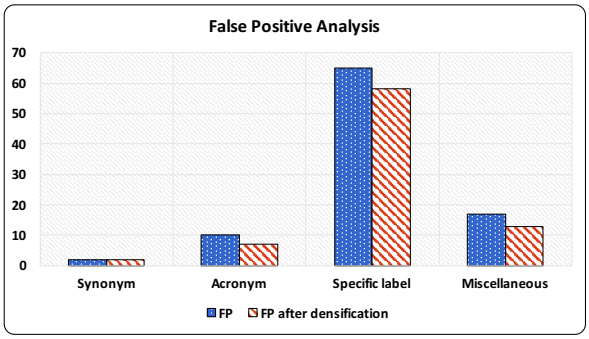


Figure 3: False positives of ELDEN before and after KG densification. Errors reduce with use of ψ_{dense} and ψ_{ELDEN} measures. (Please see Sec 6.4)

manually analyzed 240 wrong predictions of Yamada16 and compared it with that of ELDEN, and the results are presented in Figure 3. We found errors to reduce with use of KG densification features and most of the errors eliminated were in "Specific label" class. Errors in this class called for better modeling of mention's context and link-based similarity (Ling et al., 2015).(More details of this analysis in the supplementary document.)

7 Conclusion

We started this study by analyzing the performance of a state-of-the-art Entity Linking (EL) system and found that its performance was low when linking entities sparsely-connected in the KG. We saw that this can be addressed by densifying the KG with respect to the given entity. We proposed ELDEN, which densifies edge graph of entities using pseudo entities and mentions of entities in a large web corpus. Through our experiments, we find that ELDEN outperforms state-of-the-art baseline on benchmark datasets.

We believe that ELDENs combination of KG densification and entity embeddings is novel. Poor performance of EL systems on sparsely connected entities has been recognized as one of the open challenges by prior research. ELDEN performs well on sparsely connected entities too, as a validation of our method of combining KG densification followed by embedding. Our approach may be applied to any KG as the densification is performed with the help of *unstructured data*, and not any specific KG. We hope the simple graph densification method utilized in ELDEN will be of much interest to the research community.

Pseudo entities can be looked at as entity candidates for KG expansion, as also noted by Farid *et al.* (2016). In future, we plan to enhance ELDEN using EL of pseudo entities to estimate *entity prior* of entities not present in KG. We also plan to explore entity embeddings obtained using other graph densifying methods.

8 Acknowledgments

We thank the Microsoft Research India Travel Grants for generous travel funds to attend and present this paper at NAACL.

References

Ander Barrena, Aitor Soroa, and Eneko Agirre. 2016. Alleviating poor context with background knowledge for named entity disambiguation. In *ACL (1)*. The Association for Computer Linguistics.

Indrajit Bhattacharya and Lise Getoor. ???? Online collective entity resolution.

Leo Breiman. 1998. Arcing classifier (with discussion and a rejoinder by the author). *Ann. Statist.* 26(3):801–849. https://doi.org/10.1214/aos/1024691079.

Raluca Budiu, Christiaan Royer, and Peter Pirolli. 2007. Modeling information scent: A comparison of lsa, pmi and glsa similarity measures on common tests and corpora. In *Large Scale Semantic Access to Content (Text, Image, Video, and Sound)*. Paris, France, France, RIAO '07, pages 314–332. http://dl.acm.org/citation.cfm?id=1931390.1931422.

Dipak Chaudhari, Om P. Damani, and Srivatsan Laxman. 2011. Lexical co-occurrence, statistical significance, and word association. *CoRR* abs/1008.5287.

Xiao Cheng and Dan Roth. 2013. Relational inference for wikification. In *EMNLP*. ACL, pages 1787–1796.

Andrew Chisholm and Ben Hachey. 2015. Entity disambiguation with web links. *Transactions of the Association for Computational Linguistics* 3:145–156. https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/494.

Kenneth Ward Church and Patrick Hanks. 1989. Word association norms, mutual information, and lexicography. ACL '89, pages 76–83. https://doi.org/10.3115/981623.981633.

Om P. Damani. 2013. Improving pointwise mutual information (PMI) by incorporating significant co-occurrence. *CoRR* abs/1307.0596.

Lee R. Dice. 1945. Measures of the amount of ecologic association between species. *Ecology* 26(3):297–302. https://doi.org/10.2307/1932409.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. ACM, New York, NY, USA, KDD '14, pages 601–610. https://doi.org/10.1145/2623330.2623623.

Joe Ellis, Jeremy Getman, and Stephanie Strassel. 2014. Overview of linguistic resource for the tac kbp 2014 evaluations: Planning, execution, and results. Linguistic Data Consortium, University of Pennsylvania.

Weiyi Fang, Jianwen Zhang, Dilin Wang, Zheng Chen, and Ming Li. 2016. Entity disambiguation by knowledge and text jointly embedding. In *CoNLL*.

Mina Farid, Ihab F. Ilyas, Steven Euijong Whang, and Cong Yu. 2016. Lonlies: Estimating property values for long tail entities. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, USA, SIGIR '16, pages 1125–1128. https://doi.org/10.1145/2911451.2911466.

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep joint entity disambiguation with local neural attention. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 2609–2619. http://www.aclweb.org/anthology/D17-1276.

Amir Globerson, Nevena Lazic, Soumen Chakrabarti, Amarnag Subramanya, Michael Ringaard, and Fernando Pereira. 2016. Collective entity resolution with multi-focal attention. In *ACL*.

Yoav Goldberg and Omer Levy. 2014. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *CoRR* abs/1402.3722. http://arxiv.org/abs/1402.3722.

Stephen Guo, Ming-Wei Chang, and Emre Kiciman. 2013. To link or not to link? a study on end-to-end tweet entity linking. In *NAACL-HLT 2013*. http://research.microsoft.com/apps/pubs/default.aspx?id=183909.

Zhaochen Guo and Denilson Barbosa. 2014. Robust entity linking via random walks. ACM, New York, NY, USA, CIKM '14, pages 499–508. https://doi.org/10.1145/2661829.2661887.

Xianpei Han and Jun Zhao. 2010. Structural semantic relatedness: A knowledge-based method to named entity disambiguation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '10, pages 50–59. http://dl.acm.org/citation.cfm?id=1858681.1858687.

Zhengyan He, Shujie Liu, Yang Song, Mu Li, Ming Zhou, and Houfeng Wang. 2013. Efficient collective entity linking with stacking. EMNLP. http://research.microsoft.com/apps/pubs/default.aspx?id=202249.

Manjunath Hegde and Partha P. Talukdar. 2015. An entity-centric approach for overcoming knowledge graph sparsity. Association for Computational Linguistics, Lisbon, Portugal, EMNLP '15, pages 530–535. http://aclweb.org/anthology/D15-1061.

Johannes Hoffart, Stephan Seufert, Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. 2012. Kore: Keyphrase overlap relatedness for entity disambiguation. ACM, New York, NY, USA, CIKM '12, pages 545–554. https://doi.org/10.1145/2396761.2396832.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP '11, pages 782–792. http://dl.acm.org/citation.cfm?id=2145432.2145521.

Hongzhao Huang, Larry Heck, and Heng Ji. 2015. Leveraging deep neural networks and knowledge graphs for entity disambiguation. *CoRR* abs/1504.07678. http://arxiv.org/abs/1504.07678.

Heng Ji, Ralph Grishman, Hoa Trang Dang, Kira Griffitt, and Joe Ellis. 2010. Overview of the tac 2010 knowledge base population track. In *In Third Text Analysis Conference (TAC)*.

Bhushan Kotnis, Pradeep Bansal, and Partha P. Talukdar. 2015. Knowledge base inference using bridging entities. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. pages 2038–2043. http://aclweb.org/anthology/D/D15/D15-1241.pdf.

Xiao Ling, Sameer Singh, and Dan Weld. 2015. Design challenges for entity linking. *Transactions of the Association for Computational Linguistics (TACL)* 3.

Víctor Martínez, Fernando Berzal, and Juan-Carlos Cubero. 2016. A survey of link prediction in complex networks. *ACM Comput. Surv.* 49(4):69:1–69:33. https://doi.org/10.1145/3012704.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS 2013*. pages 3111–3119.

David Milne and Ian H. Witten. 2008. Learning to Link with Wikipedia. ACM, CIKM'08, pages 509–518.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Maria Pershina, Yifan He, and Ralph Grishman. 2015. *Personalized page rank for named entity disambiguation*, Association for Computational Linguistics (ACL), pages 238–243. NAACL HLT 2015.

Ridho Reinanda, Edgar Meij, and Maarten de Rijke. 2016. Document filtering for long-tail entities. ACM, CIKM'16.

W. Shen, J. Han, J. Wang, X. Yuan, and Z. Yang. 2018. Shine+: A general framework for domain-specific entity linking with heterogeneous information networks. *IEEE Transactions on Knowledge and Data Engineering* 30(2):353–366. https://doi.org/10.1109/TKDE.2017.2730862.

W. Shen, J. Wang, and J. Han. 2015. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering* 27(2):443–460. https://doi.org/10.1109/TKDE.2014.2327028.

Yaming Sun, Lei Lin, Duyu Tang, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2015. Modeling mention, context and entity with neural networks for entity disambiguation. In Qiang Yang and Michael Wooldridge, editors, *IJCAI*. AAAI Press, pages 1333–1339. http://dblp.uni-trier.de/db/conf/ijcai/ijcai2015.html#SunLTYJW15.

Bilyana Taneva and Gerhard Weikum. 2013. Gem-based entity-knowledge maintenance. ACM, New York, NY, USA, CIKM '13, pages 149–158. https://doi.org/10.1145/2505515.2505715.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. CoNLL 2016, pages 250–259. http://aclweb.org/anthology/K/K16/K16-1025.pdf.

Stefan Zwicklbauer, Christin Seifert, and Michael Granitzer. 2016. Robust and collective entity disambiguation through semantic embeddings. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, USA, SIGIR '16, pages 425–434. https://doi.org/10.1145/2911451.2911535.

Interpretable Charge Predictions for Criminal Cases: Learning to Generate Court Views from Fact Descriptions

[1]Hai Ye*, [1]Xin Jiang*, [2]Zhunchen Luo*, [1]Wenhan Chao†
[1] School of Computer Science and Engineering, Beihang University
[2] Information Research Center of Military Science
PLA Academy of Military Science
[1,2] Beijing, China
[1] {yehai, xinjiang, chaowenhan}@buaa.edu.cn
[2] zhunchenluo@gmail.com

Abstract

In this paper, we propose to study the problem of COURT VIEW GENeration from the fact description in a criminal case. The task aims to improve the interpretability of charge prediction systems and help automatic legal document generation. We formulate this task as a text-to-text natural language generation (NLG) problem. Sequence-to-sequence model has achieved cutting-edge performances in many NLG tasks. However, due to the non-distinctions of fact descriptions, it is hard for Seq2Seq model to generate charge-discriminative court views. In this work, we explore charge labels to tackle this issue. We propose a label-conditioned Seq2Seq model with attention for this problem, to decode court views conditioned on encoded charge labels. Experimental results show the effectiveness of our method.[1]

1 Introduction

Previous work has brought up multiple legal assistant systems with various functions, such as finding relevant cases given the query (Chen et al., 2013), providing applicable law articles for a given case (Liu and Liao, 2005) and etc., which have substantially improved the working efficiency. As legal assistant systems, charge prediction systems aim to determine appropriate charges such as *homicide* and *assault* for varied criminal cases by analyzing textual fact descriptions from cases (Luo et al., 2017), but ignore to give out the interpretations for the charge determination.

Court view is the written explanation from judges to interprete the charge decision for certain criminal case and is also the core part in a legal document, which consists of *rationales* and a *charge* where the charge is supported by the rationales as shown in Fig. 1. In this work, we propose to study the problem of COURT VIEW GENeration from fact descriptions in cases, and we formulate it as a text-to-text natural language generation (NLG) problem (Gatt and Krahmer, 2017). The input is the fact description in a case and the output is the corresponding court view. We only focus on generating rationales because charges can be decided by judges or charge prediction systems by also analyzing the fact descriptions (Luo et al., 2017; Lin et al., 2012). COURT-VIEW-GEN has beneficial functions, in that: (1) improve the interpretability of charge prediction systems by generating rationales in court views to support the predicted charges. The justification for charge decision is as important as deciding the charge itself (Hendricks et al., 2016; Lei et al., 2016). (2) benefit the automatic legal document generation as legal assistant systems, by automatically generating court views from fact descriptions, to release much human labor especially for simple cases but in large amount, where fact descriptions can be obtained from legal professionals or techniques such as information extraction (Cowie and Lehnert, 1996).

COURT-VIEW-GEN is not a trivial task. High-quality rationales in court views should contain the important fact details such as the degree of injury for charge of *intentional injury*, as they are important basis for charge determination. Fact details are like the summary for the fact description similar to the task of DOCument SUMmarization (Yao et al., 2017). However, rationales are not the simple summary with only fact details, to support charges, they should be charge-discriminative with *deduced information* which does not appear in fact descriptions. The fact descriptions for charge of *negligent homicide* usually only describe someone being killed without direct statement about

* indicates equal contribution.
† Corresponding author.
[1]Data and codes are available at https://github.com/oceanypt/Court-View-Gen.

Proceedings of NAACL-HLT 2018, pages 1854–1864
New Orleans, Louisiana, June 1 - 6, 2018.

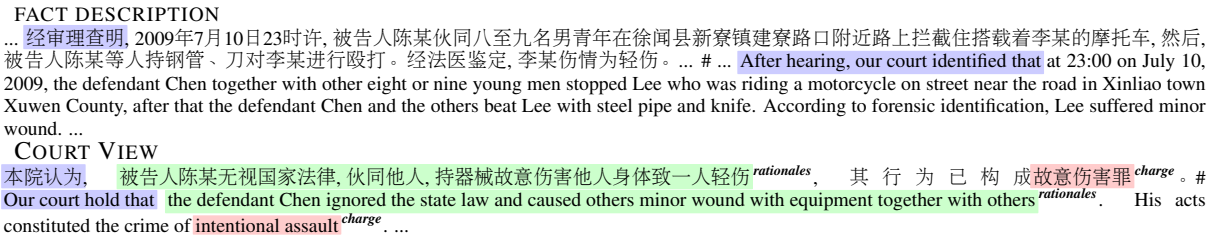


Figure 1: An example of fact description and court view from a legal document in a case.

the motive for killing, DOC-SUM will only summarize the fact of someone being killed, but rationales have to further contain the killing intention, aiming to be discriminative from those rationales for other charges like *intentional homicide*. However, it is hard to generate charge-discriminative rationales when input fact descriptions are not distinct among other facts with different charges. The fact descriptions for charge of *intentional homicide* are similar to those for charge of *negligent homicide* and also describe someone being killed but without clear motive, making it hard to generate charge-discriminative court views with accurate killing motives among the two charges.

Recently, sequence-to-sequence model with encoder-decoder paradigm (Sutskever et al., 2014) has achieved cutting-edge results in many NLG tasks, such as paraphrase (Mallinson et al., 2017), code generation (Ling et al., 2016) and question generation (Du et al., 2017). Seq2Seq model has also exhibited state-of-the-art performances on task of DOC-SUM (Chopra et al., 2016; Tan et al., 2017). However, non-distinctions of fact descriptions render Seq2Seq model hard to generate charge-discriminative rationales. In this paper, we explore charge labels of the corresponding fact descriptions, to benefit generating charge-discriminative rationales, where charge labels can be easily decided by human or charge prediction systems. Charge labels will provide with extra information to classify the non-discriminative fact descriptions. We propose a *label-conditioned* Seq2Seq model with attention for our task, in which fact descriptions are encoded into context vectors by an encoder and a decoder generates rationales with these vectors. We further encode charges as the labels and decode the rationales conditioned on the labels, to entail the decoder to learn to select gold-charge-related words to decode. Widely used attention mechanism (Luong et al., 2015) is fused into the Seq2Seq model, to learn to align target words to fact details in fact descriptions. Similar to Luo et al. (2017), we evaluate our model on Chinese criminal cases by constructing dataset from Chinese government website.

Our contributions in this paper can be summarized as follows:

- We propose the task of *court view generation* and release a real-world dataset for this task.
- We formulate the task as a text-to-text NLG problem. We utilize charge labels to benefit charge-discriminative court views generation, and propose a label-conditioned sequence-to-sequence model with attention for this task.
- Extensive experiments are conducted on a real-world dataset. The results show the efficiency of our model and exploiting charge labels for charge-discriminations improvement.

2 Related Work

Our work is firstly related to previous studies on legal assistant systems. Previous work considers the task of charge prediction as a text classification problem (Luo et al., 2017; Liu et al., 2004; Liu and Hsieh, 2006; Lin et al., 2012). Recently, Luo et al. (2017) investigate deep learning methods for this task. Besides, there are also works on identifying applicable articles for a given case (Liu and Liao, 2005; Liu and Hsieh, 2006; Liu et al., 2015), answering legal questions as a consulting system (Kim et al., 2014; Carvalho et al., 2015) and searching relevant cases for a given query (Raghav et al., 2016; Chen et al., 2013). As a legal assistant system, COURT-VIEW-GEN can benefit automatic legal document generation by generating court views from fact descriptions obtained from the last phase, through legal professionals or other technics like information extraction (Cowie and Lehnert, 1996) from raw documents in a case, if we generate legal documents step by step.

Our work is also related to recent studies on model interpretation (Ribeiro et al., 2016; Lipton, 2016; Ling et al., 2017). Recently, much work has

paid attention to giving textual explanations for classifications. Hendricks et al. (2016) generate visual explanations for image classification. Lei et al. (2016) propose to learn to select most supportive snippets from raw texts for text classification. COURT-VIEW-GEN can improve the interpretability of charge prediction systems by generating textual court views when predict the charges.

Our label-conditioned Seq2Seq model steams from widely used encoder-decoder paradigm (Sutskever et al., 2014) which has been widely used in machine translation (Bahdanau et al., 2014; Luong et al., 2015), summarization (Tan et al., 2017; Nallapati et al., 2016; Chopra et al., 2016; Cheng and Lapata, 2016), semantic parsing (Dong and Lapata, 2016) and paraphrase (Mallinson et al., 2017) or other NLG problems such as product review generation (Dong et al., 2017) and code generation (Yin and Neubig, 2017; Ling et al., 2016). Hendricks et al. (2016) propose to encode image labels for visual-language models to generate justification texts for image classification. We also introduce charge labels into Seq2Seq model to improve the charge-discriminations of generated rationales. Widely used attention mechanism (Luong et al., 2015; Xu et al., 2015) is applied to generate fact details more accurately.

3 COURT-VIEW-GEN Problem

Court View is the judicial explanation to interpret the reasons for the court making such charge for a case, consisting of the rationales and the charge supported by the rationales as shown in Fig. 1. In this work, we only focus on generating the part of rationales in court views. Charge prediction can be achieved by human or charge prediction systems (Luo et al., 2017). Final court views can be easily constructed by combining the generated rationales and the pre-decided charges.

Fact Description is the identified facts in a case (relevant events that have happened) such as the criminal acts (e.g. *degree of injury*).

The input of our model is the word sequential fact description in a case and the output is a word sequential court view (rationales part). We define the fact description as $\mathrm{x} = (x_1, x_2, \cdots, x_{|\mathrm{x}|})$ and the corresponding rationales as $\mathrm{y} = (y_1, y_2, \cdots, y_{|\mathrm{y}|})$. The charge for the case is denoted as v and will be exploited for COURT-VIEW-GEN. The task of COURT-VIEW-GEN is to find $\hat{\mathrm{y}}$ given x conditioned on the charge label v:

$$\hat{\mathrm{y}} = \arg\max_{\mathrm{y}} P(\mathrm{y}|\mathrm{x}, \mathrm{v}) \tag{1}$$

where $P(\mathrm{y}|\mathrm{x}, \mathrm{v})$ is the likelihood of the predicted rationales in the court view.

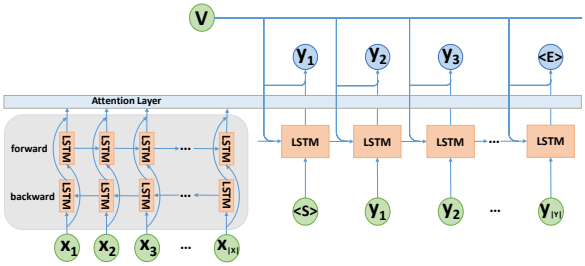


Figure 2: Label-conditioned Seq2Seq model with attention.

4 Our Model

4.1 Sequence-to-Sequence Model with Attention

Similar to Luong et al. (2015), our Seq2Seq model consists of an encoder and a decoder as shown in Fig. 2. Given the pair of fact description and rationales in court view (x, y), the encoder reads the word sequence of x and then the decoder will learn to predict the rationales in court view y. The probability of predicted y is given as follows:

$$P(\mathrm{y}) = \prod_{i=1}^{|\mathrm{y}|} P(y_i|y_{<i}, \mathrm{x}) \tag{2}$$

where $y_{<i} = y_1, y_2, \cdots, y_{i-1}$. We use a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) as encoder and use another LSTM as decoder similar to Du et al. (2017).

Decoder. From the decoder side, at time t, the probability to predict y_t is computed as follows:

$$P(y_t|y_{<t}, \mathbf{c}_t) = \mathrm{softmax}(\mathbf{W}_1 \tanh(\mathbf{W}_0[\mathbf{s}_t; \mathbf{c}_t]))$$

where $\mathbf{W}_0$ and $\mathbf{W}_1$ are learnable parameters; $\mathbf{s}_t$ is the hidden state of decoder at time t; $\mathbf{c}_t$ is the context vector generated from the encoder side containing the information of x at time t; here the bias of model is omitted for simplification. The hidden state of $\mathbf{s}_t$ is computed as follows:

$$\mathbf{s}_t = \mathrm{LSTM}_d(y_{t-1}, \mathbf{s}_{t-1})$$

where y_{t-1} is the word embedding vector for pre-state target word at time $t-1$. The initial state for decoder is initialized by the last state of encoder.

Context vector of $\mathbf{c}_t$ is computed by summing up the hidden states of $\{\mathbf{h}_k\}_{k=1}^{|\mathrm{x}|}$ generated by the encoder with attention mechanism and we adopt global attention (Luong et al., 2015) in our work.

Encoder with Attention. We adopt a one-layer bidirectional LSTM to encoder the fact descriptions. The hidden state $\mathbf{h}_j$ at time j is computed as follows:

$$\mathbf{h}_j = [\overrightarrow{\mathbf{h}_j}; \overleftarrow{\mathbf{h}_j}]$$

where $\mathbf{h}_j$ is the concatenation of forward hidden state $\overrightarrow{\mathbf{h}_j}$ and backward hidden state $\overleftarrow{\mathbf{h}_j}$, specifically:

$$\overrightarrow{\mathbf{h}_j} = \overrightarrow{\mathrm{LSTM}}_e(x_j, \overrightarrow{\mathbf{h}}_{j-1})$$
$$\overleftarrow{\mathbf{h}_j} = \overleftarrow{\mathrm{LSTM}}_e(x_j, \overleftarrow{\mathbf{h}}_{j+1})$$

The hidden outputs $\{\mathbf{h}_k\}_{k=1}^{|\mathrm{x}|}$ will be used to compute the context vectors for decoder.

From the decoder side, by applying attention mechanism at time i, the context vector of $\mathbf{c}_i$ is generated as follows:

$$\mathbf{c}_i = \sum_{j=1}^{|\mathrm{x}|} \alpha_{ij} \mathbf{h}_j \tag{3}$$

where α_{ij} is the attention weight and is computed as follows:

$$\alpha_{ij} = \frac{\exp(\mathbf{s}_i^T \mathbf{W}_2 \mathbf{h}_j)}{\sum_{k=1}^{|\mathrm{x}|} \exp(\mathbf{s}_i^T \mathbf{W}_2 \mathbf{h}_k)} \tag{4}$$

where $\mathbf{s}_i$ is the hidden output state at time i in the decoder side.

4.2 Label-conditioned Sequence-to-Sequence Model with Attention

Given the tuple of fact description, rationales in court view and charge label $(\mathrm{x}, \mathrm{y}, \mathrm{v})$, the probability to predict y is computed as follows:

$$P(\mathrm{y}) = \prod_{i=1}^{|\mathrm{y}|} P(y_i | y_{<i}, \mathrm{x}, \mathrm{v}) \tag{5}$$

From this formula, encoding charge labels provides extra constrains comparing to Eq. (2), and restricts the target word searching space from the whole space to only gold-charge-related space for rationales generation, so model can generate more charge-distinct rationales. Charge labels are trainable parameters denoted by $\mathbf{E}^v$ where every charge will have a trainable vector from $\mathbf{E}^v$, which will be updated in the model training process.

As shown in Fig. 2, in the decoder side, at time t, y_t is predicted with the probability as follows:

$$P(y_t | y_{<t}, \mathbf{c}_t, \mathrm{v}) = \mathrm{softmax}(\mathbf{W}_1 \tanh(\mathbf{W}_0[\mathbf{s}_t; \mathbf{c}_t; \mathbf{E}^v_{[\mathrm{v}]}])) \tag{6}$$

where $\mathbf{E}^v_{[\mathrm{v}]}$ is the embedding vector of v obtained from $\mathbf{E}^v$. In this formula, we connect charge label v to $\mathbf{s}_t$ and $\mathbf{c}_t$ aiming to influence the word selection process. We hope that our model can learn the latent connections between the charge label v and the words of rationales in court views through this way, to decode out charge-discriminative words.

As shown in Fig. 2, we further embed the charge label v to highlight the computing of hidden state $\mathbf{s}_t$ at time t and $\mathbf{s}_t$ is merged as follows:

$$\begin{aligned} \mathbf{s}_t &= \mathrm{LSTM}_d(y_{t-1}, \mathbf{s}^v_{t-1}) \\ \mathbf{s}^v_{t-1} &= f_v(\mathbf{s}_{t-1}, \mathrm{v}) \\ f_v &= \tanh(\mathbf{W}^v[\mathbf{s}_{t-1}; \mathbf{E}^v_{[\mathrm{v}]}] + \mathbf{b}^v) \end{aligned} \tag{7}$$

where $\mathbf{W}^v$ and $\mathbf{b}^v$ are learnable parameters. In this way, the information of charge label can be embedded into $\mathbf{s}_t$. From Eq. (3) and Eq. (4), attention weights $\mathbf{c}_t$ are computed from $\mathbf{s}_t$, so encoding the charge label v to hidden states will make the model concentrate more on charge-related information from fact descriptions to help generate more accurate fact details.

4.3 Model Training and Inference

Suppose we are given the training data: $\{\mathrm{x}^{(i)}, \mathrm{y}^{(i)}, \mathrm{v}^{(i)}\}_{i=1}^N$, we aim to maximize the log-likelihood of generated rationales in court views given the fact descriptions and charge labels, so the loss function is computed as follows:

$$\begin{aligned} \mathcal{L}(\theta) &= -\sum_{i=1}^{N} \log P(\mathrm{y}^{(i)} | \mathrm{x}^{(i)}, \mathrm{v}^{(i)}; \theta) \\ &= -\sum_{i=1}^{N} \sum_{j=1}^{|\mathrm{y}^{(i)}|} \log P(y_j^{(i)} | y_{<j}^{(i)}, \mathrm{x}^{(i)}, \mathrm{v}^{(i)}; \theta) \end{aligned}$$

We split the training data into multiple batches with size of 64 and adopt adam learning (Kingma and Ba, 2014) to update the parameters in every batch data. At the inference time, we encode the fact descriptions and charge labels into vectors and use the decoder to generate rationales in

# Training set	153706
# Dev set	9152
# Test set	9123
Avg. # tokens in fact desc.	219.9
Avg. # tokens in rationales	30.6
Num. of # charge labels	51
# Dict. size in fact desc.	222482
# Dict. size in rationales	21305

Table 1: Statistics of our dataset.

court views based on Eq. (1). We adopt the algorithm of beam search to generate rationales. Beam search size is set to 5. To make generation process stoppable, an indicator tag "</s>" is added to the end of the rationales sequences, and when "</s>" is generated the inference process will be terminated. The generated word sequential paths will be ranked and the one with largest value is selected as the final rationales in court view.

5 Experiments

5.1 Data Preparation

Following Luo et al. (2017), we construct dataset from the published legal documents in China Judgements Online[2]. We extract the fact descriptions, rationales in court views and charge labels using regular expressions. The paragraph started with "经审理查明" ("our court identified that") is regarded as the fact description and the part between "本院认为" ("our court hold that") and the charge are regarded as the rationales. Nearly all the samples in dataset match this extraction pattern. Length threshold of 256 is set up, and fact description longer than that will be stripped, leaving too long facts for future study. We use the tokens of "<name>", "<num>" and "<date>" to replace the names, numbers and dates appearing in the corpus. We tokenize the Chinese texts with the open source tool of HanLP[3]. For charge labels, we select the top 50 charge labels ranked by occurrences and leave the left charges as others. Details about our dataset are shown in Table 1.

For cases with multiple charges and multiple defendants, we can separate the fact descriptions and the court views according to the charges or the defendants. In this work, we only focus on the cases with one defendant and one charge, leaving the complex cases for future study, so we can collect large enough data from the published legal documents without human to annotate the data.

5.2 Experimental Settings

Word embeddings are randomly initialized and updated in the training process, with the size of 512 tuned from $\{256, 512, 1024\}$. Charge label vectors are initialized randomly with size of 512. Maximal vocabulary size of encoder is set to 100K words and decoder is 50K by stripping words exceeding the bounds. Maximal source length is 256 and target is 50. The hidden size of LSTM is 1024 tuned from $\{256, 512, 1024\}$. We choose perplexity as the update metric. Early stopping mechanism is applied to train the model. The initial learning rate is set to 0.0003 and the reduce factor is 0.5. Model performance will be checked on the validation set after every 1000 batches training and keep the parameters with lowest perplexity. Training process will be terminated if model performance is not improved for successive 8 times.

5.3 Comparisons with Baselines

Evaluation Metrics. We adopt both automatic evaluation and human judgement for model evaluation. BLEU-4 score (Papineni et al., 2002) and variant Rouge scores (Lin, 2004) are adopted for automatic evaluation which have been widely used in many NLG tasks. We set up two evaluation dimensions for human judgement: 1) how *fluent* of the rationales in court view is; 2) how *accurate* of the rationales is, aiming to evaluate how many fact details have been accurately expressed in the generated rationales. We adopt 5 scales for both *fluent* and *accurate* evaluation (5 is for the best). We ask three annotators who knows well about our task to conduct the human judgement. We randomly select 100 generated rationales in court views for every evaluated method. The three raters are also asked to judge whether rationales can be adopted for use in comprehensive evaluation (*adoptable*) and record the number of adoptable rationales for every evaluated method.

Baselines.

• **Rand** is to randomly select rationales in court views from the training set (method of $\textbf{Rand}_{\textbf{all}}$). We also randomly choose rationales from pools with same charge labels ($\textbf{Rand}_{\textbf{charge}}$). Adopting Rand method is to indicate the low bound performance of COURT-VIEW-GEN.

• **BM25** is a retrieval baseline to index the fact description match to the input fact description with highest BM25 score (Robertson and Walker,

[2]http://wenshu.court.gov.cn

[3]https://github.com/hankcs/HanLP

	AUTOMATIC EVALUATION			
MODEL (%)	B-4	R-1	R-2	R-L
Rand_{all}	6.4	26.5	6.2	25.1
$\text{Rand}_{\text{charge}}$	24.9	53.6	29.1	49.3
BM25_{f2f}	40.1	63.5	43.7	60.3
$\text{BM25}_{\text{f2f+charge}}$	42.8	67.1	47.4	63.8
MOSES+	6.2	39.8	20.8	18.6
NN-S2S	38.4	65.5	45.1	62.2
RAS†	44.1**	69.1**	50.3**	65.9**
Ours	**45.8**	**70.9**	**52.5**	**67.7**

	HUMAN JUDGEMENT		
MODEL	FLUENT	ACC.	ADOPT.(%)
BM25_{f2f}	4.95	3.66**	0.47**
$\text{BM25}_{\text{f2f+charge}}$	4.94	3.90**	0.50**
MOSES+	1.39**	1.31**	0**
NN-S2S	4.97	4.07**	0.62*
RAS†	4.96	4.25*	0.64*
Ours	4.93	**4.54**	**0.72**

Table 2: Results of automatic evaluation and human judgement with BLEU-4 and full length of F1 scores of variant Rouges. Best results are labeled as boldface. Statistical significance is indicated with $**(p < 0.01)$ and $*$ $(p < 0.05)$ comparing to our full model.

1994) from the training set, and use its rationales as the result (**BM25_{f2f}**). Similar fact descriptions may have the similar rationales. Fact descriptions from pools with same charges are also retrieved (**$\text{BM25}_{\text{f2f+charge}}$**), to see how much improvement that adding charge labels can gender.

• **MOSES+** (Koehn et al., 2007) is a phrase-based statistical machine translation system mapping fact descriptions to rationales. KenLM (Heafield et al., 2013) is adopted to train a trigram language model on the target corpus of training set which is tuned on the validation set with MERT.

• **NN-S2S** is the basic Seq2Seq model without attention from Sutskever et al. (2014) for machine translation. We set one LSTM layer for encoding and another one LSTM layer for decoding. We adopt perplexity for training metric and select the model with lowest perplexity on validation set.

• **RAS†** is an attention based abstract summarization model from Chopra et al. (2016). To deal with the much longer fact descriptions, we exploit the more advanced bidirectional LSTM model for the encoder instead of the simple convolutional model. Another LSTM model is set as the decoder coherent to Chopra et al. (2016).

Experimental Results. In automatic evaluation from Table 2, the evaluation scores are relatively high even for method of $\text{Rand}_{\text{charge}}$, which indicates that the expressions of the rationales with same charge labels are similar with many overlapped n-grams, such that the rationales for *crime of theft* usually begin with "以非法占有为目的" ("in intention of illegal possession"). Accurately generating fact details like degree of injury or time of theft is more difficult. Retrieval method by adding charge labels is the strong baseline even better than basic Seq2Seq model. Adding attention mechanism will improve the performance indicated by the method of RAS† which is superior to retrieval methods. By exploiting charge labels, our full model achieves the best performance. The performances of statistical machine translation model are really poor, for it requiring the lengths of parallel corpus to be similar.

In human evaluation, we can see that retrieval methods can not accurately express fact details, for that it is hard to retrieve rationales containing details all matching the fact descriptions. However, our system can learn to generate fact details by analyzing fact descriptions. Dropping attention mechanism will have negative effects on model performance. RAS† has worse performance in ACC. whose main reason may lie in that RAS† can not generate charge-discriminative rationales with *deduced information*, which demonstrates that our task is not the simple DOC-SUM task. For the *fluent* evaluation, generation models are highly close to retrieval methods whose rationales are written by humans, which reflects that the generation models can generate highly natural rationales.

5.4 Further Analysis

Impact of Exploiting Charge Labels.

• **Charge2Charge Analysis.** We first analyze the effects of exploiting charge labels on model performance charge to charge, by dropping to encode charges based on our full model. From the results shown in Fig. 3, we can find that the results can be improved much by exploiting charge labels among nearly all charges. This result also indicates that the non-distinct fact descriptions are common among nearly all charges and reflects the difficulty of this task, but utilizing charge labels can release the seriousness of the problem.

• **Charge-discriminations Analysis.** We further evaluate the effects of charge labels for charge-discriminations improvement on specific charges with non-distinct fact descriptions: *intentional homicide*, *negligent homicide*, *duty embezzlement* and *corruption*. For every charge, two participants are asked to count the number of ra-

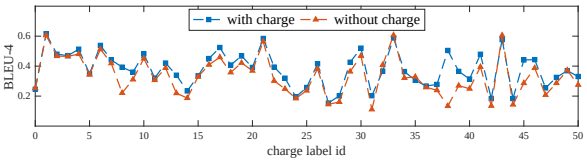


Figure 3: Results of impact of exploiting charge labels evaluated charge to charge in the metric of BLEU-4 (similar results can gender in other three metrics but are omitted for space saving).

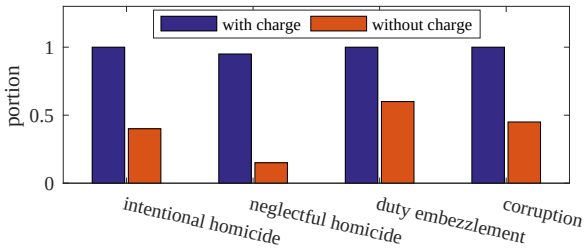


Figure 4: Portions of charge-discriminative rationales in court views for every charge with 20 candidates.

tionales that are relevant to the charge on 20 randomly selected candidates.

From Fig. 4, the number of charge discriminative rationales can be much improved among every charge by utilizing charge information, which demonstrates that charge labels can provide with much extra charge-related information to deal with latent information in fact descriptions. For crimes of *homicide*, the motives for killing are latent in the descriptions of killing without direct statement, but our system can learn to align the motives in rationales to the charge labels which are the strong distinct indicator for the two motives.

Ablation Study. We also ablate our full model to reveal different components of encoding charge labels for performance improvement. As shown in Table 3, " / softmax comp." is to remove the part in Eq. (6) and yields worse performance than our full model, but better than " / charge comp." that ignores to encode charge labels, which is same to the situation of " / hidden comp." that removes the part in Eq. (7). Our full model is still better than the ablated models. This finding shows that both of the methods of exploiting charge labels can improve model performance and stacking them will achieve better results.

Attention Mechanism Analysis. Heat map in Fig. 5 is used to illustrate the attention mechanism. The "slight injury" is aligned between the source and target. "responsibility" and "run" are well aligned to "away", which demonstrate the

	Ablation Study			
Model (%)	B-4	R-1	R-2	R-L
Our System	45.8**	70.9**	52.5**	67.7**
/ softmax comp.	45.7**	70.8**	52.3**	67.5**
/ hidden comp.	45.7**	70.2*	51.9*	67.0*
/ charge comp.	43.7	68.6	49.7	65.5

Table 3: Results of ablation study. Statistical significance is indicated with $**$ ($p < 0.01$) and $*$ ($p < 0.05$) comparing to the ablation of " / charge comp.".

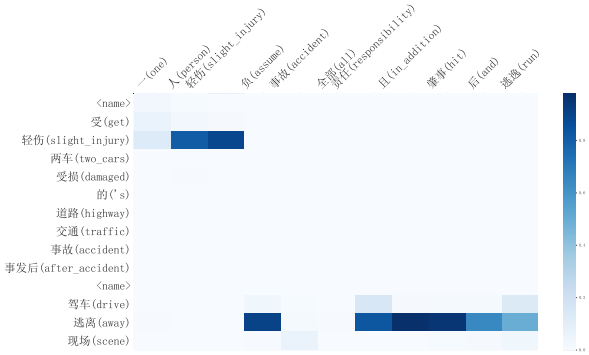


Figure 5: Heat map for attention mechanism analysis. The column is the source and the raw is the target.

efficiency of attention mechanism for generating fact details by forcing context vectors to focus more on fact details.

Performance by Reference Size. We further investigate the model performance by rationales length in court views. As shown in Fig. 6, not surprisingly the model performance drops when the length of reference rationales increases. Within the size of 30, BLEU-4 score can maintain around 0.4 and F1 score keeps around 0.5. Exceeding the length of 30, model performance decreases dramatically.

Human eval. vs. Automatic eval. *Are BLEU and Rouge suitable for* Court-View-Gen *evaluation*? Following the work of (Papineni et al., 2002; Liu et al., 2016), for the models evaluated in human judgemnet, we draw the linear regressions of their BLEU-4 and variant Rouge scores, as the function of Acc. and Adopt. from human judgement respectively as shown in Fig. 7. From

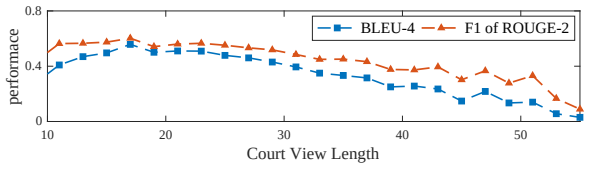


Figure 6: Model performance by rationales length with BLEU-4 and full length of F1 of Rouge-2.

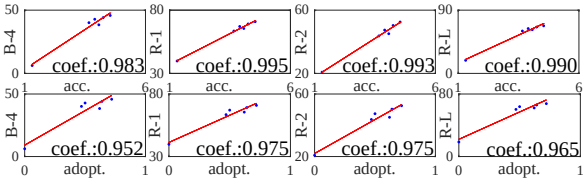


Figure 7: ACC. and ADOPT. of human judgement predict automatic evaluation scores.

the results, we can find that automatic evaluations track well with the human judgement with high correlation coefficients. This finding demonstrates that BLEU-4 and variant Rouges are adoptable for COURT-VIEW-GEN evaluation and provides the basis for future studies on this task.

Error Analysis. Our model has the drawback of *generating latent fact details*, which appear in rationales but are not clearly expressed in fact descriptions. For example, for the time of theft in charge of *larceny*, the term of "多次" ("several times") appears in rationales but may not be expressed in fact descriptions directly, only with descriptions of larceny but without exact term for this detail, so it will be hard for attention mechanism to learn to align "多次" in rationales to latent information in fact descriptions. In the generated rationales on test set, we find that only 42.4% samples can accurately extract out the term of "多次". It may need designed rules to deal with such details, like that count the time of theft from the descriptions, and if the time exceeds 1 then the term of "多次" can be generated in rationales.

5.5 Analysis through Cases

Fake Charge Label Conditioned Study. What generated rationales in court views will be if they are conditioned on fake charge labels? We select one fact description with gold charge of *intentional injury*, then generate rationales conditioned on fake charges of *defiance and affray crime*, *intentional homicide* and *neglectful homicide*.

From Fig. 8, the rationales conditioned on fake charges will be partly relevant to fake charge labels and also maintain fact details from the input fact description of gold charge. For the fake charge of *intentional homicide*, its fact details should be "caused someone dead", but instead express "causing someone slight injury" which is relevant to charge of *intentional injury*. For charge prediction systems, the discriminations between fact details and charges will help to remind people that the prediction results may be unreliable.

Case Study. Examples of generated rationales in court views are shown in Fig. 8. Generally speaking, our full label-conditioned model has high accuracy on generating fact details better than baseline models. For charges of *traffic accident crime* and *negligent homicide*, all fact details are generated. The extra information from charge labels helps the model to capture more important fact details, by forcing model to pay more attention to charge-related information in fact descriptions.

As for the charge-discrimination analysis, from the rationales of *negligent homicide*, we can infer that its fact description may relate to a traffic accident, which is non-distinct from that for *traffic accident crime*. Without encoding charge labels, Ours $_{/c}$ wrongly generates the rationales coherent to *traffic accident crime*, because traffic accidents are the strong indicator for traffic crimes, but the charge label will provide extra bias towards the *homicide crime*, so our full model can generate highly discriminative rationales. Utilizing charge labels, retrieval method can easily retrieve charge-related rationales, but hard to index rationales with accurate fact details. For charge of *larceny*, our full model extracts nearly all fact details but misses the fact of "多次"("several times"), reflecting the shortcoming of dealing with latent details.

6 Conclusion and Future Work

In this paper, we propose a novel task of court view generation and formulate it as a text-to-text NLG problem. We utilize charge labels to benefit the generation of charge-discriminative rationales in court views and propose a label-conditioned Seq2Seq model with attention for this task. Extensive experiments show the efficiency of our model and exploiting charge labels.

In the future: 1) More advanced technologies like reinforcement learning (Sutton and Barto, 1998) can be introduced to generate latent fact details such as the time of theft more accurately; 2) In this work, we only generate rationales in court views omitting charge prediction, it is interesting to see whether jointly generating the two parts will benefit both of the tasks; 3) Studying verification mechanism is meaningful to judge whether generated court views can really be adopted which is important for COURT-VIEW-GEN in practice; 4) More complex cases with multiple charges and multiple defendants will be considered in the future.

Model	[charge]	Generated Court Views Conditioned on Fake Charge Label
Gold	/故意伤害罪/ PP 故意 伤害 他人 身体 ， 致 一 人 轻伤 。# [intentional injury] PP intentionally injured others body , caused one people slight injury .	
Ours	/寻衅滋事罪/ PP 随意 殴打 他人 ， 致 一 人 轻伤 ， 情节 恶劣 。# [defiance and affray crime] PP beat others at will , caused one people slight injury . /故意杀人罪/ PP 故意 非法 剥夺 他人 生命 ， 致 一 人 轻伤 。# [intentional homicide] PP intentionally illegally deprived someone of life , caused one people slight injury . /过失致人死亡罪/ PP 过失 致 一 人 轻伤 。# [neglectful homicide] PP neglectfully caused one people slight injury .	
Model	[charge]	Generated Court Views
Gold	/交通肇事罪/ PP 违反 交通 运输 管理 法规 ， 造成 一 人 死亡 ， 二 人 受伤 的 交通 事故 ， 负 事故 的 全部 责任 。# [traffic accident crime] PP violated traffic transportation management regulations , caused one people dead , two people injured , take accident's full responsibility . /过失致人死亡罪/ PP 在 驾驶 机动车 过程 中 ， 疏忽大意 ， 致使 他人 被 碾压 致死 。# [negligent homicide] PP when driving car , being neglectful , caused people dead by rolling . /盗窃罪/ PP 以 非法 占有 为 目的 ， 伙同 他人 多次 秘密 窃取 公民 财物 ， 数额 较 大 。# [larceny] PP in intention of illegal possession , ganged up with others and stole goods secretly in relatively large amount for several times .	
Ours	PP 违反 交通 运输 管理 法规 ， 发生 交通 事故 ， 致 一 人 死亡 ， 二 人 受伤 ， 负 事故 的 全部 责任 。# PP violated traffic transportation management regulations , caused traffic accident , caused one people dead , two people injured , take accident's full responsibility . ✓ PP 因 疏忽大意 致 一 人 死亡 。# PP neglectfully caused one people dead . ✓ PP 以 非法 占有 为 目的 ， 结伙 他人 秘密 窃取 他人 财物 ， 数额 较 大 。# PP in intention of illegal possession , ganged up with others and stole goods secretly in relatively large amount . ✗	
Ours / c	PP 违反 交通 运输 管理 法规 ， 发生 重大 交通 事故 ， 致 一 人 死亡 ， 负 事故 的 全部 责任 。# PP violated traffic transportation management regulations , caused severe traffic accident , caused one people dead , took accident's full responsibility ✗ PP 违反 交通 运输 管理 法规 ， 发生 重大 交通 事故 ， 致 一 人 死亡 ， 负 事故 的 全部 责任 。# PP violated traffic transportation management regulations , caused severe traffic accident , caused one people dead , took accident's full responsibility . ✗ PP 以 非法 占有 为 目的 ， 秘密 窃取 他人 财物 ， 数额 较 大 。# PP in intention of illegal possession , stole goods secretly in relatively large amount . ✗	
BM25$_{f2f+c}$	PP 违反 道路 交通 运输 管理 法规 ， 致 一 人 死亡 且 负 事故 主要 责任 。# PP violated road traffic transportation management regulations , caused one people dead , took accident's main responsibility . ✗ PP 驾驶 车辆 过程 中 疏忽大意 ， 过失 致 一 人 死亡 。# PP when driving , neglectfully caused one people dead . ✓ PP 以 非法 占有 为 目的 ， 秘密 窃取 公民 财物 。# PP in intention of possession , stole goods secretly . ✗	

Figure 8: Fake charge label conditioned generated rationales in court views and examples of generated rationales.

Acknowledgments

Firstly, we would like to thank Yansong Feng, Yu Wu, Xiaojun Wan, Li Dong, Pengcheng Yin and Zichao Yan for their insightful comments and suggestions. We also very appreciate the comments from anonymous reviewers which will help further improve our work. This work is supported by National Natural Science Foundation of China (No. 61602490) and National Key R&D Plan (No. 2017YFB1402403). The work was done when Hai Ye interned in Beihang University from August, 2017 to January, 2018.

References

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

Danilo S. Carvalho, Minh-Tien Nguyen, Chien-Xuan Tran, and Minh-Le Nguyen. 2015. Lexical-morphological modeling for legal text analysis. In *New Frontiers in Artificial Intelligence - JSAI-isAI 2015 Workshops, LENLS, JURISIN, AAA, HAT-MASH, TSDAA, ASD-HR, and SKL, Kanagawa, Japan, November 16-18, 2015, Revised Selected Papers*. pages 295–311.

Yen-Liang Chen, Yi-Hung Liu, and Wu-Liang Ho. 2013. A text mining approach to assist the general public in the retrieval of legal documents. *JASIST* 64(2):280–290.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Volume 1: Long Papers*.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 93–98.

Jim Cowie and Wendy Lehnert. 1996. Information extraction. *Communications of the ACM* 39(1):80–91.

Li Dong, Shaohan Huang, Furu Wei, Mirella Lapata, Ming Zhou, and Ke Xu. 2017. Learning to generate product reviews from attributes. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, pages 623–632.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 33–43.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Volume 1: Long Papers*. pages 1342–1352.

Albert Gatt and Emiel Krahmer. 2017. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *CoRR* abs/1703.09902.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified

kneser-ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9, Volume 2: Short Papers*. pages 690–696.

Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. 2016. Generating visual explanations. In *Computer Vision - ECCV 2016 - 14th European Conference, Proceedings, Part IV*. pages 3–19.

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

Mi-Young Kim, Ying Xu, and Randy Goebel. 2014. Legal question answering using ranking SVM and syntactic/semantic similarity. In *New Frontiers in Artificial Intelligence - JSAI-isAI 2014 Workshops, LENLS, JURISIN, and GABA, Kanagawa, Japan, October 27-28, 2014, Revised Selected Papers*. pages 244–258.

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

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL 2007, Proceedings of the 45th Annual Meeting of the Association*.

Tao Lei, Regina Barzilay, and Tommi S. Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 107–117.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proceedings of the ACL-04 Workshop. Association for Computational Linguistics*. pages 74–81.

Wan-Chen Lin, Tsung-Ting Kuo, Tung-Jia Chang, Chueh-An Yen, Chao-Ju Chen, and Shou-de Lin. 2012. Exploiting machine learning models for chinese legal documents labeling, case classification, and sentencing prediction. *IJCLCLP* 17(4).

Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, Fumin Wang, and Andrew Senior. 2016. Latent predictor networks for code generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Volume 1: Long Papers*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Volume 1: Long Papers*. pages 158–167.

Zachary Chase Lipton. 2016. The mythos of model interpretability. *CoRR* abs/1606.03490.

Chao-Lin Liu, Cheng-Tsung Chang, and Jim-How Ho. 2004. Case instance generation and refinement for case-based criminal summary judgments in chinese. *J. Inf. Sci. Eng.* 20(4):783–800.

Chao-Lin Liu and Chwen-Dar Hsieh. 2006. Exploring phrase-based classification of judicial documents for criminal charges in chinese. In *Foundations of Intelligent Systems, 16th International Symposium, ISMIS 2006, Bari, Italy, September 27-29, 2006, Proceedings*. pages 681–690.

Chao-Lin Liu and Ting-Ming Liao. 2005. Classifying criminal charges in chinese for web-based legal services. In *Web Technologies Research and Development - APWeb 2005, 7th Asia-Pacific Web Conference Proceedings*. pages 64–75.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 2016*. pages 2122–2132.

Yi-Hung Liu, Yen-Liang Chen, and Wu-Liang Ho. 2015. Predicting associated statutes for legal problems. *Inf. Process. Manage.* 51(1):194–211.

Bingfeng Luo, Yansong Feng, Jianbo Xu, Xiang Zhang, and Dongyan Zhao. 2017. Learning to predict charges for criminal cases with legal basis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 2717–2726.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 1412–1421.

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2017. Paraphrasing revisited with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, Volume 1: Long Papers*. pages 881–893.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, August 11-12, 2016*. pages 280–290.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. pages 311–318.

K. Raghav, P. K. Reddy, and V. B. Reddy. 2016. Analyzing the extraction of relevant legal judgments using paragraph-level and citation information. In *AI4JArtificial Intelligence for Justice*.

Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. pages 1135–1144.

Stephen E. Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. (Special Issue of the SIGIR Forum)*. pages 232–241.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*. pages 3104–3112.

Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Volume 1: Long Papers*. pages 1171–1181.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning, ICML*. pages 2048–2057.

Jin-ge Yao, Xiaojun Wan, and Jianguo Xiao. 2017. Recent advances in document summarization. *Knowl. Inf. Syst.* 53(2):297–336.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Volume 1: Long Papers*. pages 440–450.

Delete, Retrieve, Generate: A Simple Approach to Sentiment and Style Transfer

Juncen Li[*1] **Robin Jia**[2] **He He**[2] **Percy Liang**[2]
[1] WeChat Search Application Department, Tencent
[2] Computer Science Department, Stanford University
juncenli@tencent.com
{robinjia,hehe,pliang}@cs.stanford.edu

Abstract

We consider the task of text attribute transfer: transforming a sentence to alter a specific attribute (e.g., sentiment) while preserving its attribute-independent content (e.g., changing *"screen is just the right size"* to *"screen is too small"*). Our training data includes only sentences labeled with their attribute (e.g., positive or negative), but not pairs of sentences that differ only in their attributes, so we must learn to disentangle attributes from attribute-independent content in an unsupervised way. Previous work using adversarial methods has struggled to produce high-quality outputs. In this paper, we propose simpler methods motivated by the observation that text attributes are often marked by distinctive phrases (e.g., *"too small"*). Our strongest method extracts content words by deleting phrases associated with the sentence's original attribute value, retrieves new phrases associated with the target attribute, and uses a neural model to fluently combine these into a final output. On human evaluation, our best method generates grammatical and appropriate responses on 22% more inputs than the best previous system, averaged over three attribute transfer datasets: altering sentiment of reviews on Yelp, altering sentiment of reviews on Amazon, and altering image captions to be more romantic or humorous.

1 Introduction

The success of natural language generation (NLG) systems depends on their ability to carefully control not only the topic of produced utterances, but also attributes such as sentiment and style. The desire for more sophisticated, controllable NLG has led to increased interest in text attribute transfer—the task of editing a sentence to alter specific attributes, such as style, sentiment, and tense (Hu et al., 2017; Shen et al., 2017; Fu et al., 2018). In each of these cases, the goal is to convert a sentence with one attribute (e.g., negative sentiment) to one with a different attribute (e.g., positive sentiment), while preserving all attribute-independent content[1] (e.g., what properties of a restaurant are being discussed). Typically, aligned sentences with the same content but different attributes are not available; systems must learn to disentangle attributes and content given only unaligned sentences labeled with attributes.

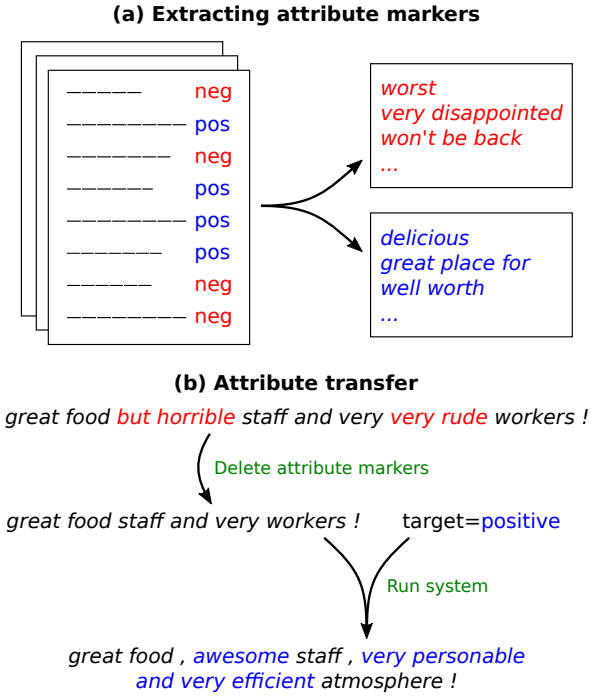


Figure 1: An overview of our approach. (a) We identify attribute markers from an unaligned corpus. (b) We transfer attributes by removing markers of the original attribute, then generating a new sentence conditioned on the remaining words and the target attribute.

Previous work has attempted to use adversarial

[*] Work done while the author was a visiting researcher at Stanford University.

[1] Henceforth, we refer to attribute-independent content as simply *content*, for simplicity.

Proceedings of NAACL-HLT 2018, pages 1865–1874
New Orleans, Louisiana, June 1 - 6, 2018.

networks (Shen et al., 2017; Fu et al., 2018) for this task, but—as we demonstrate—their outputs tend to be low-quality, as judged by human raters. These models are also difficult to train (Salimans et al., 2016; Arjovsky and Bottou, 2017; Bousmalis et al., 2017).

In this work, we propose a set of simpler, easier-to-train systems that leverage an important observation: attribute transfer can often be accomplished by changing a few *attribute markers*—words or phrases in the sentence that are indicative of a particular attribute—while leaving the rest of the sentence largely unchanged. Figure 1 shows an example in which the sentiment of a sentence can be altered by changing a few sentiment-specific phrases but keeping other words fixed.

With this intuition, we first propose a simple baseline that already outperforms prior adversarial approaches. Consider a sentiment transfer (negative to positive) task. First, from unaligned corpora of positive and negative sentences, we identify attribute markers by finding phrases that occur much more often within sentences of one attribute than the other (e.g., *"worst"* and *"very disppointed"* are negative markers). Second, given a sentence, we delete any negative markers in it, and regard the remaining words as its content. Third, we retrieve a sentence with similar content from the positive corpus.

We further improve upon this baseline by incorporating a neural generative model, as shown in Figure 1. Our neural system extracts content words in the same way as our baseline, then generates the final output with an RNN decoder that conditions on the extracted content and the target attribute. This approach has significant benefits at training time, compared to adversarial networks: having already separated content and attribute, we simply train our neural model to reconstruct sentences in the training data as an auto-encoder.

We test our methods on three text attribute transfer datasets: altering sentiment of Yelp reviews, altering sentiment of Amazon reviews, and altering image captions to be more romantic or humorous. Averaged across these three datasets, our simple baseline generated grammatical sentences with appropriate content and attribute 23% of the time, according to human raters; in contrast, the best adversarial method achieved only 12%. Our best neural system in turn outperformed our baseline, achieving an average success rate of 34%. Our code and data, including newly collected human reference outputs for the Yelp and Amazon domains, can be found at https://github.com/lijuncen/Sentiment-and-Style-Transfer.

2 Problem Statement

We assume access to a corpus of labeled sentences $\mathcal{D} = \{(x_1, v_1), \ldots, (x_m, v_m)\}$, where x_i is a sentence and $v_i \in \mathcal{V}$, the set of possible attributes (e.g., for sentiment, $\mathcal{V} = \{\text{"positive"}, \text{"negative"}\}$). We define $\mathcal{D}_v = \{x : (x, v) \in \mathcal{D}\}$, the set of sentences in the corpus with attribute v. Crucially, we do not assume access to a parallel corpus that pairs sentences with different attributes and the same content.

Our goal is to learn a model that takes as input (x, v^{tgt}) where x is a sentence exhibiting source (original) attribute v^{src}, and v^{tgt} is the target attribute, and outputs a sentence y that retains the content of x while exhibiting v^{tgt}.

3 Approach

As a motivating example, suppose we wanted to change the sentiment of *"The chicken was delicious."* from positive to negative. Here the word *"delicious"* is the only sentiment-bearing word, so we just need to replace it with an appropriate negative sentiment word. More generally, we find that the attribute is often *localized* to a small fraction of the words, an inductive bias not captured by previous work.

How do we know which negative sentiment word to insert? The key observation is that the remaining content words provide strong cues: given *"The chicken was ..."*, one can infer that a taste-related word like *"bland"* fits, but a word like *"rude"* does not, even though both have negative sentiment. In other words, while the deleted sentiment words do contain non-sentiment information too, this information can often be recovered using the other content words.

In the rest of this section, we describe our four systems: two baselines (RETRIEVEONLY and TEMPLATEBASED) and two neural models (DELETEONLY and DELETEANDRETRIEVE). An overview of all four systems is shown in Figure 2. Formally, the main components of these systems are as follows:

1. **Delete**: All 4 systems use the same procedure to separate the words in x into a set of

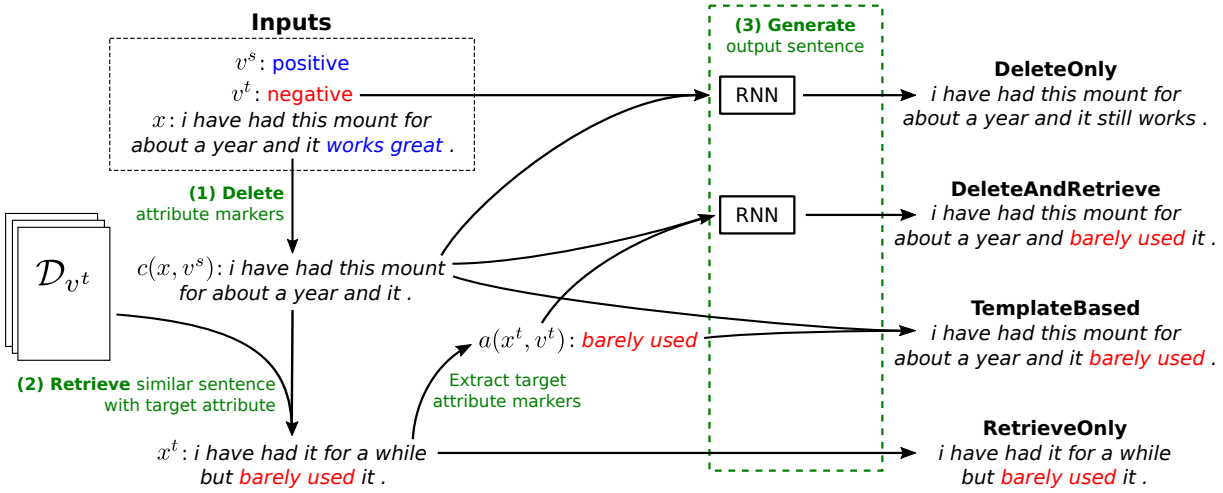


Figure 2: Our four proposed methods on the same sentence, taken from the AMAZON dataset. Every method uses the same procedure (1) to separate attribute and content by deleting attribute markers; they differ in the construction of the target sentence. RETRIEVEONLY directly returns the sentence retrieved in (2). TEMPLATEBASED combines the content with the target attribute markers in the retrieved sentence by slot filling. DELETEANDRETRIEVE generates the output from the content and the retrieved target attribute markers with an RNN. DELETEONLY generates the output from the content and the target attribute with an RNN.

attribute markers $a(x, v^{\text{src}})$ and a sequence of content words $c(x, v^{\text{src}})$.

2. **Retrieve**: 3 of the 4 systems look through the corpus and retrieve a sentence x^{tgt} that has the target attribute v^{tgt} and whose content is similar to that of x.
3. **Generate**: Given the content $c(x, v^{\text{src}})$, target attribute v^{tgt}, and (optionally) the retrieved sentence x^{tgt}, each system generates y, either in a rule-based fashion or with a neural sequence-to-sequence model.

We describe each component in detail below.

3.1 Delete

We propose a simple method to delete attribute markers (n-grams) that have the most discriminative power. Formally, for any $v \in \mathcal{V}$, we define the *salience* of an n-gram u with respect to v by its (smoothed) relative frequency in $\mathcal{D}_v$:

$$s(u, v) = \frac{\text{count}(u, \mathcal{D}_v) + \lambda}{\left(\sum_{v' \in \mathcal{V}, v' \neq v} \text{count}(u, \mathcal{D}_{v'})\right) + \lambda}, \quad (1)$$

where $\text{count}(u, \mathcal{D}_v)$ denotes the number of times an n-gram u appears in $\mathcal{D}_v$, and λ is the smoothing parameter. We declare u to be an attribute marker for v if $s(u, v)$ is larger than a specified threshold γ. The attributed markers can be viewed as discriminative features for a Naive Bayes classifier.

We define $a(x, v^{\text{src}})$ to be the set of all source attribute markers in x, and define $c(x, v^{\text{src}})$ as the sequence of words after deleting all markers in $a(x, v^{\text{src}})$ from x. For example, for *"The chicken was delicious,"* we would delete *"delicious"* and consider *"The chicken was..."* to be the content (Figure 2, Step 1).

3.2 Retrieve

To decide what words to insert into $c(x, v^{\text{src}})$, one useful strategy is to look at similar sentences with the target attribute. For example, negative sentences that use phrases similar to *"The chicken was..."* are more likely to contain *"bland"* than *"rude."* Therefore, we retrieve sentences of similar content and use target attribute markers in them for insertion.

Formally, we retrieve x^{tgt} according to:

$$x^{\text{tgt}} = \underset{x' \in \mathcal{D}_{v^{\text{tgt}}}}{\text{argmin}}\, d(c(x, v^{\text{src}}), c(x', v^{\text{tgt}})), \quad (2)$$

where d may be any distance metric comparing two sequences of words. We experiment with two options: (i) TF-IDF weighted word overlap and (ii) Euclidean distance using the content embeddings in Section 3.3 (Figure 2, Step 2).

3.3 Generate

Finally, we describe how each system generates y (Figure 2, Step 3).

RETRIEVEONLY returns the retrieved sentence x^{tgt} verbatim. This is guaranteed to produce a grammatical sentence with the target attribute, but its content might not be similar to x.

TEMPLATEBASED replaces the attribute markers deleted from the source sentence $a(x, v^{\text{src}})$ with those of the target sentence $a(x^{\text{tgt}}, v^{\text{tgt}})$.[2] This strategy relies on the assumption that if two attribute markers appear in similar contexts , they are roughly syntactically exchangeable. For example, *"love"* and *"don't like"* appear in similar contexts (e.g., *"i **love** this place."* and *"i **don't like** this place."*), and exchanging them is syntactically valid. However, this naive swapping of attribute markers can result in ungrammatical outputs.

DELETEONLY first embeds the content $c(x, v^{\text{src}})$ into a vector using an RNN. It then concatenates the final hidden state with a learned embedding for v^{tgt}, and feeds this into an RNN decoder to generate y. The decoder attempts to produce words indicative of the source content and target attribute, while remaining fluent.

DELETEANDRETRIEVE is similar to DELETEONLY, but uses the attribute markers of the retrieved sentence x^{tgt} rather than the target attribute v^{tgt}. Like DELETEONLY, it encodes $c(x, v^{\text{src}})$ with an RNN. It then encodes the sequence of attribute markers $a(x^{\text{tgt}}, v^{\text{tgt}})$ with another RNN. The RNN decoder uses the concatenation of this vector and the content embedding to generate y.

DELETEANDRETRIEVE combines the advantages of TEMPLATEBASED and DELETEONLY. Unlike TEMPLATEBASED, DELETEANDRETRIEVE can pick a better place to insert the given attribute markers, and can add or remove function words to ensure grammaticality. Compared to DELETEONLY, DELETEANDRETRIEVE has a stronger inductive bias towards using target attribute markers that are likely to fit in the current context. Guu et al. (2018) showed that retrieval strategies like ours can help neural generative models. Finally, DELETEANDRETRIEVE gives us finer control over the output; for example, we can control the degree of sentiment by deciding whether to add *"good"* or *"fantastic"* based on the retrieved sentence x^{tgt}.

[2] Markers are replaced from left to right, in order. If there are not enough markers in x^{tgt}, we use an empty string.

3.4 Training

We now describe how to train DELETEANDRETRIEVE and DELETEONLY. Recall that at training time, we do not have access to ground truth outputs that express the target attribute. Instead, we train DELETEONLY to reconstruct the sentences in the training corpus given their content and *original* attribute value by maximizing:

$$L(\theta) = \sum_{(x, v^{\text{src}}) \in \mathcal{D}} \log p(x \mid c(x, v^{\text{src}}), v^{\text{src}}); \theta). \tag{3}$$

For DELETEANDRETRIEVE, we could similarly learn an auto-encoder that reconstructs x from $c(x, v^{\text{src}})$ and $a(x, v^{\text{src}})$. However, this results in a trivial solution: because $a(x, v^{\text{src}})$ and $c(x, v^{\text{src}})$ were known to come from the same sentence, the model merely learns to stitch the two sequences together without any smoothing. Such a model would fare poorly at test time, when we may need to alter some words to fluently combine $a(x^{\text{tgt}}, v^{\text{tgt}})$ with $c(x, v^{\text{src}})$. To address this train/test mismatch, we adopt a denoising method similar to the denoising auto-encoder (Vincent et al., 2008). During training, we apply some noise to $a(x, v^{\text{src}})$ by randomly altering each attribute marker in it independently with probability 0.1. Specifically, we replace an attribute marker with another randomly selected attribute marker of the same attribute and word-level edit distance 1 if such a noising marker exists, e.g., *"was very rude"* to *"very rude"*, which produces $a'(x, v^{\text{src}})$.

Therefore, the training objective for DELETEANDRETRIEVE is to maximize:

$$L(\theta) = \sum_{(x, v^{\text{src}}) \in \mathcal{D}} \log p(x \mid c(x, v^{\text{src}}), a'(x, v^{\text{src}}); \theta). \tag{4}$$

4 Experiments

We evaluated our approach on three domains: flipping sentiment of Yelp reviews (YELP) and Amazon reviews (AMAZON), and changing image captions to be romantic or humorous (CAPTIONS). We compared our four systems to human references and three previously published adversarial approaches. As judged by human raters, both of our two baselines outperform all three adversarial methods. Moreover, DELETEANDRETRIEVE outperforms all other automatic approaches.

Dataset	Attributes	Train	Dev	Test
YELP	Positive	270K	2000	500
	Negative	180K	2000	500
CAPTIONS	Romantic	6000	300	0
	Humorous	6000	300	0
	Factual	0	0	300
AMAZON	Positive	277K	985	500
	Negative	278K	1015	500

Table 1: Dataset statistics.

4.1 Datasets

First, we describe the three datasets we use, which are commonly used in prior works too. All datasets are randomly split into train, development, and test sets (Table 1).

YELP Each example is a sentence from a business review on Yelp, and is labeled as having either positive or negative sentiment.

AMAZON Similar to YELP, each example is a sentence from a product review on Amazon, and is labeled as having either positive or negative sentiment (He and McAuley, 2016).

CAPTIONS In the CAPTIONS dataset (Gan et al., 2017), each example is a sentence that describes an image, and is labeled as either factual, romantic, or humorous. We focus on the task of converting factual sentences into romantic and humorous ones. Unlike YELP and AMAZON, CAPTIONS is actually an aligned corpus—it contains captions for the same image in different styles. Our systems do not use these alignments, but we use them as gold references for evaluation.

CAPTIONS is also unique in that we reconstruct romantic and humorous sentences during training, whereas at test time we are given factual captions. We assume these factual captions carry only content, and therefore do not look for and delete factual attribute markers; The model essentially only inserts romantic or humorous attribute markers as appropriate.

4.2 Human References

To supply human reference outputs to which we could compare the system outputs for YELP and AMAZON, we hired crowdworkers on Amazon Mechanical Turk to write gold outputs for all test sentences. Workers were instructed to edit a sentence to flip its sentiment while preserving its content.

Our delete-retrieve-generate approach relies on the prior knowledge that to accomplish attribute transfer, a small number of attribute markers should be changed, and most other words should be kept the same. We analyzed our human reference data to understand the extent to which humans follow this pattern. We measured whether humans preserved words our system marks as content, and changed words our system marks as attribute-related (Section 3.1). We define the *content word preservation rate* S_c as the average fraction of words our system marks as content that were preserved by humans, and the *attribute-related word change rate* S_a as the average fraction of words our system marks as attribute-related that were changed by humans:

$$S_c = \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{(x, v^{\text{src}}, y^*) \in \mathcal{D}_{\text{test}}} \frac{|c(x, v^{\text{src}}) \cap y^*|}{|c(x, v^{\text{src}})|}$$
$$S_a = 1 - \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{(x, v^{\text{src}}, y^*) \in \mathcal{D}_{\text{test}}} \frac{|a(x, v^{\text{src}}) \cap y^*|}{|a(x, v^{\text{src}})|}, \quad (5)$$

where $\mathcal{D}_{\text{test}}$ is the test set, y^* is the human reference sentence, and $|\cdot|$ denotes the number of non-stopwords. Higher values of S_c and S_a indicate that humans preserve content words and change attribute-related words, in line with the inductive bias of our model. S_c is 0.61, 0.71, and 0.50 on YELP, AMAZON, and CAPTIONS, respectively; S_a is 0.72 on YELP and 0.54 on AMAZON (not applicable on CAPTIONS).

To understand why humans sometimes deviated from the inductive bias of our model, we randomly sampled 50 cases from YELP where humans changed a content word or preserved an attribute-related word. 70% of changed content words were unimportant words (e.g., *"whole"* was deleted from *"whole experience"*), and another 18% were paraphrases (e.g., *"charge"* became *"price"*); the remaining 12% were errors where the system mislabeled an attribute-related word as a content word (e.g., *"old"* became *"new"*). 84% of preserved attribute-related words did pertain to sentiment but remained fixed due to changes in the surrounding context (e.g., *"don't **like**"* became *"**like**"*, and *"below **average**"* became *"above **average**"*); the remaining 16% were mistagged by our system as being attribute-related (e.g., *"walked out"*).

4.3 Previous Methods

We compare with three previous models, all of which use adversarial training. **STYLEEMBED-**

	Yelp				Amazon				Captions			
	Gra	Con	Att	Suc	Gra	Con	Att	Suc	Gra	Con	Att	Suc
CrossAligned	2.8	2.9	3.5	14%	3.2	2.5	2.9	7%	3.9	2.0	3.2	16%
StyleEmbedding	3.5	3.7	2.1	9%	3.2	2.9	2.8	11%	3.3	2.9	3.0	17%
MultiDecoder	2.8	3.1	3.0	8%	3.0	2.6	2.8	7%	3.4	2.8	3.2	18%
RetrieveOnly	**4.2**	2.7	**4.2**	25%	3.8	2.8	3.1	17%	**4.2**	2.6	3.8	27%
TemplateBased	3.0	**3.9**	3.9	21%	3.4	3.6	3.1	19%	3.3	**4.1**	3.5	33%
DeleteOnly	3.0	3.7	3.9	24%	3.7	**3.8**	3.2	24%	3.6	3.5	3.5	32%
DeleteAndRetrieve	3.3	3.7	4.0	**29%**	**3.9**	3.7	**3.4**	**29%**	3.8	3.5	**3.9**	**43%**
Human	4.6	4.5	4.5	75%	4.2	4.0	3.7	44%	4.3	3.9	4.0	56%

Table 2: Human evaluation results on all three datasets. We show average human ratings for grammaticality (Gra), content preservation (Con), and target attribute match (Att) on a 1 to 5 Likert scale, as well as overall success rate (Suc). On all three datasets, DeleteAndRetrieve is the best overall system, and all four of our methods outperform previous work.

ding (Fu et al., 2018) learns an vector encoding of the source sentence such that a decoder can use it to reconstruct the sentence, but a discriminator, which tries to identify the source attribute using this encoding, fails. They use a basic MLP discriminator and an LSTM decoder. **MultiDecoder** (Fu et al., 2018) is similar to StyleEmbedding, except that it uses a different decoder for each attribute value. **CrossAligned** (Shen et al., 2017) also encodes the source sentence into a vector, but the discriminator looks at the hidden states of the RNN decoder instead. The system is trained so that the discriminator cannot distinguish these hidden states from those obtained by forcing the decoder to output real sentences from the target domain; this objective encourages the real and generated target sentences to look similar at a population level.

4.4 Experimental Details

For our methods, we use 128-dimensional word vectors and a single-layer GRU with 512 hidden units for both encoders and the decoder. We use the maxout activation function (Goodfellow et al., 2013). All parameters are initialized by sampling from a uniform distribution between -0.1 and 0.1. For optimization, we use Adadelta (Zeiler, 2012) with a minibatch size of 256.

For attribute marker extraction, we consider spans up to 4 words, and the smoothing parameter λ is set to 1. We set the attribute marker threshold γ, which controls the precision and recall of our attribute markers, to 15, 5.5 and 5 for Yelp, Amazon, and Captions. These values were set by manual inspection of the resulting markers and tuning slightly on the dev set. For retrieval, we used the TF-IDF weighted word overlap score for DeleteAndRetrieve and TemplateBased, and the Euclidean distance of content embeddings for RetrieveOnly. We find the two scoring functions give similar results.

For all neural models, we do beam search with a beam size of 10. For DeleteAndRetrieve, similar to Guu et al. (2018), we retrieve the top-10 sentences and generate results using markers from each sentence. We then select the output with the lowest perplexity given by a separately-trained neural language model on the target-domain training data.

4.5 Human Evaluation

We hired workers on Amazon Mechanical Turk to rate the outputs of all systems. For each source sentence and target attribute, the same worker was shown the output of each tested system. Workers were asked to rate each output on three criteria on a Likert scale from 1 to 5: grammaticality, similarity to the target attribute, and preservation of the source content. Finally, we consider a generated output "successful" if it is rated 4 or 5 on all three criteria. For each dataset, we evaluated 400 randomly sampled examples (200 for each target attribute).

Table 2 shows the human evaluation results. On all three datasets, both of our baselines have a higher success rate than the previously published models, and DeleteAndRetrieve achieves the best performance among all systems. Additionally, we see that human raters strongly preferred the human references to all systems, suggesting there is still significant room for improvement on this task.

We find that a human evaluator's judgment of a sentence is largely relative to other sentences being evaluated together and examples given in the instruction (different for each dataset/task). There-

fore, evaluating all system outputs in one batch is important and results on different datasets are not directly comparable.

4.6 Analysis

We analyze the strengths and weaknesses of the different systems. Table 3 show typical outputs of each system on the YELP and CAPTIONS dataset.

We first analyze the adversarial methods. CROSSALIGNED and MULTIDECODER tend to lose the content of the source sentence, as seen in both the example outputs and the overall human ratings. The decoder tends to generate a frequent but only weakly related sentence with the target attribute. On the other hand, STYLEEMBEDDING almost always generates a paraphrase of the input sentence, implying that the encoder preserves some attribute information. We conclude that there is a delicate balance between preserving the original content and dropping the original attribute, and existing adversarial models tend to sacrifice one or the other.

Next, we analyze our baselines. RETRIEVEONLY scores well on grammaticality and having the target attribute, since it retrieves sentences with the desired attribute directly from the corpus. However, it is likely to change the content when there is no perfectly aligned sentence in the target domain. In contrast, TEMPLATEBASED is good at preserving the content because the content words are guaranteed to be kept. However, it makes grammatical mistakes due to the unsmoothed combination of content and attribute words.

DELETEANDRETRIEVE and DELETEONLY achieve a good balance among grammaticality, preserving content, and changing the attribute. Both have strong inductive bias on what words should be changed, but still have the flexibility to smooth out the sentence. The main difference is that DELETEONLY fills in attribute words based on only the target attribute, whereas DELETEANDRETRIEVE conditions on retrieved attribute words. When there is a diverse set of phrases to fill in—for example in CAPTIONS— conditioning on retrieved attribute words helps generate longer sentences with more specific attribute descriptions.

4.7 Automatic Evaluation

Following previous work (Hu et al., 2017; Shen et al., 2017), we also compute automatic evaluation metrics, and compare these numbers to our human evaluation results.

We use an attribute classifier to assess whether outputs have the desired attribute (Hu et al., 2017; Shen et al., 2017). We define the *classifier score* as the fraction of outputs classified as having the target attribute. For each dataset, we train an attribute classifier on the same training data. Specifically, we encode the sentence into a vector by a bidirectional LSTM with an average pooling layer over the outputs, and train the classifier by minimizing the logistic loss.

We also compute BLEU between the output and the human references, similar to Gan et al. (2017). A high BLEU score primarily indicates that the system can correctly preserve content by retaining the same words from the source sentence as the reference. One might also hope that it has some correlation with fluency, though we expect this correlation to be much weaker.

Table 4 shows the classifier and BLEU scores. In Table 5, we compute the system-level correlation between classifier score and human judgments of attribute transfer, and between BLEU and human judgments of content preservation and grammaticality. We also plot scores given by the automatic metrics and humans in Figure 4. While the scores are sometimes well-correlated, the results vary significantly between datasets; on AMAZON, there is no correlation between the classifier score and the human evaluation. Manual inspection shows that on AMAZON, some product genres are associated with either mostly positive or mostly negative reviews. However, our systems produce, for example, negative reviews about products that are mostly discussed positively in the training set. Therefore, the classifier often gives unreliable predictions on system outputs. As expected, BLEU does not correlate well with human grammaticality ratings. The lack of automatic fluency evaluation artificially favors systems like TEMPLATEBASED, which make more grammatical mistakes. We conclude that while these automatic evaluation methods are useful for model development, they cannot replace human evaluation.

4.8 Trading off Content versus Attribute

One advantage of our methods is that we can control the trade-off between matching the target attribute and preserving the source content. To achieve different points along this trade-off curve,

From negative to positive (YELP)	
SOURCE	we sit down and we got some really slow and lazy service .
CROSSALIGNED	we *went* down and we *were a good , friendly food* .
STYLEEMBEDDING	we sit down and we got some really *slow* and *prices suck* .
MULTIDECODER	we sit down and we got some really and *fast food* .
TEMPLATEBASED	we sit down and we got some *the service is always great* and *even better* service .
RETRIEVEONLY	*i* got *a veggie hoagie that was massive* and some *grade a customer* service .
DELETEONLY	we sit down and we got some *great* and *quick* service .
DELETEANDRETRIEVE	we got *very nice place to* sit down and we got some service .
From factual to romantic (CAPTIONS)	
SOURCE	two dogs play by a tree .
CROSSALIGNED	*a dog is running through the grass* .
STYLEEMBEDDING	two dogs play *against* a tree .
MULTIDECODER	two dogs play by a tree .
TEMPLATEBASED	two dogs play by a tree *loving* .
RETRIEVEONLY	two dogs *are playing in a pool as best friends* .
DELETEANDRETRIEVE	two dogs play by a tree , *enjoying the happiness of childhood* .
DELETEONLY	two dogs *in love* play *happily* by a tree .
From negative to positive (AMAZON)	
SOURCE	this is the worst game i have come across in a long time .
CROSSALIGNED	this is the *best thing* i *ve had for a few years* .
STYLEEMBEDDING	this is the worst game i have come across in a long time .
MULTIDECODER	this is the *best knife* i have *no room with* a long time .
TEMPLATEBASED	this is the *best* come across in a long time .
RETRIEVEONLY	*the customer support is some of the best* i have come across in a long time .
DELETEONLY	this is the *best* game i have come across in a long time .
DELETEANDRETRIEVE	this is the *best* game i have come across in a long time .

Table 3: Example outputs on YELP, CAPTIONS, and AMAZON. Additional examples for transfer from opposite directions are given in Table 6. Added or changed words are in *italic*. Attribute markers are colored.

	YELP		CAPTIONS		AMAZON	
	Classifier	BLEU	Classifier	BLEU	Classifier	BLEU
CROSSALIGNED	73.7%	3.1	74.3%	0.1	**74.1%**	0.4
STYLEEMBEDDING	8.7%	**11.8**	54.7%	6.7	43.3%	10.0
MULTIDECODER	47.6%	7.1	68.5%	4.6	68.3%	5.0
TEMPLATEBASED	81.7%	**11.8**	92.5%	**17.1**	68.7%	**27.1**
RETRIEVEONLY	**95.4%**	0.4	**95.5%**	0.7	**70.3%**	0.9
DELETEONLY	85.7%	7.5	83.0%	9.0	45.6%	24.6
DELETEANDRETRIEVE	88.7%	8.4	**96.8%**	7.3	48.0%	22.8

Table 4: Automatic evaluation results. "Classifier" shows the percentage of sentences labeled as the target attribute by the classifier. BLEU measures content similarity between the output and the human reference.

we simply vary the threshold γ (Section 3.1) *at test time* to control how many attribute markers we delete from the source sentence. In contrast, other methods (Shen et al., 2017; Fu et al., 2018) would require retraining the model with different hyperparameters to achieve this effect.

Figure 3 shows this trade-off curve for DELETEANDRETRIEVE, DELETEONLY, and TEMPLATEBASED on YELP, where target attribute match is measured by the classifier score and content preservation is measured by BLEU.[3] We see a clear trade-off between changing the attribute and retaining the content.

[3] RETRIEVEONLY is less affected by what content words are preserved, especially when no good output sentence exists in the target corpus. Therefore, we found that it did not exhibit a clear content-attribute trade-off.

5 Related Work and Discussion

Our work is closely related to the recent body of work on text attribute transfer with *unaligned* data, where the key challenge to disentangle attribute and content in an unsupervised way. Most existing work (Shen et al., 2017; Zhao et al., 2018; Fu et al., 2018; Melnyk et al., 2017) uses adversarial training to separate attribute and content: the content encoder aims to fool the attribute discriminator by removing attribute information from the content embedding. However, we find that empirically it is often easy to fool the discriminator without actually removing the attribute information. Therefore, we explicitly separate attribute and content by taking advantage of the prior knowledge that the attribute is localized to parts of the sentence.

To address the problem of unaligned data, Hu

	Classifier	BLEU	
	Attribute	Content	Grammaticality
All data	0.810 ($p < 0.01$)	0.876 ($p < 0.01$)	−0.127 ($p = 0.58$)
YELP	0.991 ($p < 0.01$)	0.935 ($p < 0.01$)	0.119 ($p = 0.80$)
CAPTIONS	0.982 ($p < 0.01$)	0.991 ($p < 0.01$)	−0.631 ($p = 0.13$)
AMAZON	−0.036 ($p = 0.94$)	0.857 ($p < 0.01$)	0.306 ($p = 0.50$)

Table 5: Spearman correlation between two automatic evaluation metrics and related human evaluation scores. While some correlations are strong, the classifier exhibits poor correlation on AMAZON, and BLEU only measures content, not grammaticality.

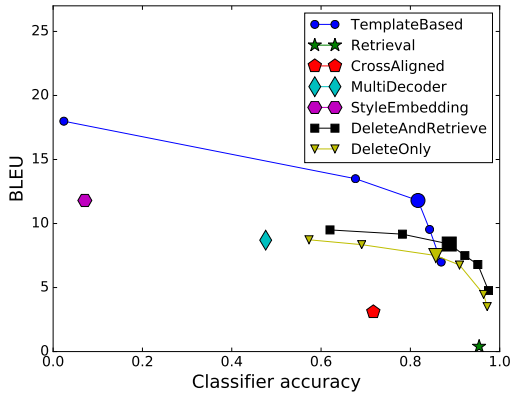


Figure 3: Trade-off curves between matching the target attribute (measured by classifier scores) and preserving the content (measured by BLEU). Bigger points on the curve correspond to settings used for both training and our official evaluation.

et al. (2017) relies on an attribute classifier to guide the generator to produce sentences with a desired attribute (e.g. sentiment, tense) in the Variational Autoencoder (VAE) framework. Similarly, Zhao et al. (2018) used a regularized autoencoder in the adversarial training framework; however, they also find that these models require extensive hyperparameter tuning and the content tends to be changed during the transfer. Shen et al. (2017) used a discriminator to align target sentences and sentences transfered to the target domain from the source domain. More recently, unsupervised machine translation models (Artetxe et al., 2017; Lample et al., 2017) used a cycle loss similar to Jun-Yan et al. (2017) to ensure that the content is preserved during the transformation. These methods often rely on bilinguial word vectors to provide word-for-word translations, which are then finetune by back-translation. Thus they can be used to further improve our results.

Our method of detecting attribute markers is reminiscent of Naive Bayes, which is a strong baseline for tasks like sentiment classification (Wang and Manning, 2012). Deleting these attribute markers can be viewed as attacking a Naive Bayes classifier by deleting the most informative features (Globerson and Roweis, 2006), similarly to how adversarial methods are trained to fool an attribute classifier. One difference is that our classifier is fixed, not jointly trained with the model.

To conclude, we have described a simple method for text attribute transfer that outperforms previous models based on adversarial training. The main leverage comes from the inductive bias that attributes are usually manifested in localized discriminative phrases. While many prior works on linguistic style analysis confirm our observation that attributes often manifest in idiosyncratic phrases (Recasens et al., 2013; Schwartz et al., 2017; Newman et al., 2003), we recognize the fact that in some problems (e.g., Pavlick and Tetreault (2017)), content and attribute cannot be so cleanly separated along phrase boundaries. Looking forward, a fruitful direction is to develop a notion of attributes more general than n-grams, but with more inductive bias than arbitrary latent vectors.

Reproducibility. All code, data, and experiments for this paper are available on the CodaLab platform at `https://worksheets.codalab.org/worksheets/0xe3eb416773ed4883bb737662b31b4948/`.

Acknowledgements. This work is supported by the DARPA Communicating with Computers (CwC) program under ARO prime contract no. W911NF- 15-1-0462. J.L. is supported by Tencent. R.J. is supported by an NSF Graduate Research Fellowship under Grant No. DGE-114747.

References

M. Arjovsky and L. Bottou. 2017. Towards principled methods for training generative adversarial networks. In *International Conference on Learning Representations (ICLR)*.

M. Artetxe, G. Labaka, E. Agirre, and K. Cho. 2017.

Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041* .

K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan. 2017. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *Computer Vision and Pattern Recognition (CVPR)*.

Z. Fu, X. Tan, N. Peng, D. Zhao, and R. Yan. 2018. Style transfer in text: Exploration and evaluation. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

C. Gan, Z. Gan, X. He, J. Gao, and L. Deng. 2017. Stylenet: Generating attractive visual captions with styles. In *Computer Vision and Pattern Recognition (CVPR)*.

A. Globerson and S. Roweis. 2006. Nightmare at test time: robust learning by feature deletion. In *International Conference on Machine Learning (ICML)*. pages 353–360.

I. Goodfellow, D. Warde-farley, M. Mirza, A. Courville, and Y. Bengio. 2013. Maxout networks. In *International Conference on Machine Learning (ICML)*. pages 1319–1327.

K. Guu, T. B. Hashimoto, Y. Oren, and P. Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association for Computational Linguistics (TACL)* 0.

R. He and J. McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *World Wide Web (WWW)*.

Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing. 2017. Toward controlled generation of text. In *International Conference on Machine Learning (ICML)*.

Z. Jun-Yan, P. Taesung, I. Phillip, and E. A. A. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *International Conference on Computer Vision (ICCV)*.

G. Lample, L. Denoyer, and M. Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043* .

I. Melnyk, C. N. dos Santos, K. Wadhawan, I. Padhi, and A. Kumar. 2017. Improved neural text attribute transfer with non-parallel data. *arXiv preprint arXiv:1711.09395* .

M. L. Newman, J. W. Pennebaker, D. S. Berry, and J. M. Richards. 2003. Lying words: Predicting deception from linguistic styles. *Personality and Social Psychology Bulletin* 29.

E. Pavlick and J. Tetreault. 2017. An empirical analysis of formality in online communication. *Transactions of the Association for Computational Linguistics (TACL)* 4.

M. Recasens, C. Danescu-Niculescu-Mizil, and D. Jurafsky. 2013. Linguistic models for analyzing and detecting biased language. In *Association for Computational Linguistics (ACL)*.

T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. 2016. Improved techniques for training gans. In *Advances in Neural Information Processing Systems (NIPS)*.

R. Schwartz, M. Sap, Y. Konstas, L. Zilles, Y. Choi, and N. A. Smith. 2017. The effect of different writing tasks on linguistic style: A case study of the ROC story cloze task. In *Computational Natural Language Learning (CoNLL)*.

T. Shen, T. Lei, R. Barzilay, and T. Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems (NIPS)*.

P. Vincent, H. Larochelle, Y. Bengio, , and P. Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning (ICML)*.

S. Wang and C. D. Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Association for Computational Linguistics (ACL)*.

M. D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

J. Zhao, Y. Kim, K. Zhang, A. M. Rush, and Y. LeCun. 2018. Adversarially regularized autoencoders. In *International Conference on Learning Representations (ICLR)*.

Adversarial Example Generation with Syntactically Controlled Paraphrase Networks

Mohit Iyyer[★†‡] **John Wieting**[★♣] **Kevin Gimpel**[♢] **Luke Zettlemoyer**[♠]

Allen Institute of Artificial Intelligence[†] UMass Amherst[‡] Carnegie Mellon University[♣]
Toyota Technological Institute at Chicago[♢] University of Washington[♠]
miyyer@cs.umass.edu jwieting@cs.cmu.edu
kgimpel@ttic.edu lsz@cs.washington.edu

Abstract

We propose *syntactically controlled paraphrase networks* (SCPNs) and use them to generate adversarial examples. Given a sentence and a target syntactic form (e.g., a constituency parse), SCPNs are trained to produce a paraphrase of the sentence with the desired syntax. We show it is possible to create training data for this task by first doing back-translation at a very large scale, and then using a parser to label the syntactic transformations that naturally occur during this process. Such data allows us to train a neural encoder-decoder model with extra inputs to specify the target syntax. A combination of automated and human evaluations show that SCPNs generate paraphrases that follow their target specifications without decreasing paraphrase quality when compared to baseline (uncontrolled) paraphrase systems. Furthermore, they are more capable of generating syntactically adversarial examples that both (1) "fool" pretrained models and (2) improve the robustness of these models to syntactic variation when used to augment their training data.

1 Introduction

Natural language processing datasets often suffer from a dearth of linguistic variation, which can hurt the generalization of models trained on them. Recent work has shown it is possible to easily "break" many learned models by evaluating them on *adversarial examples* (Goodfellow et al., 2015), which are generated by manually introducing lexical, pragmatic, and syntactic variation not seen in the training set (Ettinger et al., 2017). Robustness to such adversarial examples can potentially be improved by augmenting the training data, as shown by prior work that introduces rule-based lexical substitutions (Jia and Liang, 2017; Liang et al., 2017). However, more complex transformations, such as generating syntactically adversarial examples, remain an open challenge, as input semantics must be preserved in the face of potentially substantial structural modifications. In this paper, we introduce a new approach for learning to do *syntactically controlled paraphrase generation*: given a sentence and a target syntactic form (e.g., a constituency parse), a system must produce a paraphrase of the sentence whose syntax conforms to the target.

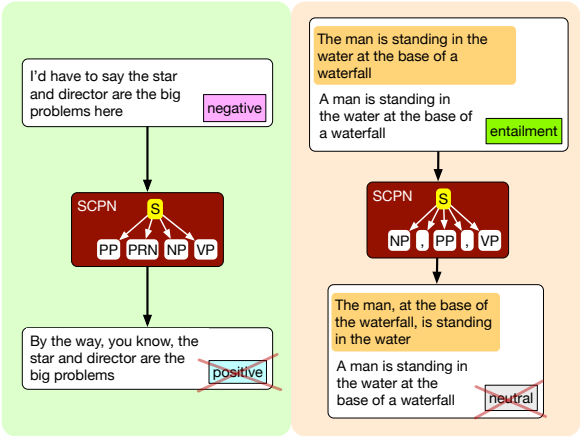


Figure 1: Adversarial examples for sentiment analysis (left) and textual entailment (right) generated by our syntactically controlled paraphrase network (SCPN) according to provided parse templates. In both cases, a pretrained classifier correctly predicts the label of the original sentence but not the corresponding paraphrase.

General purpose syntactically controlled paraphrase generation is a challenging task. Approaches that rely on handcrafted rules and grammars, such as the question generation system of McKeown (1983), support only a limited number of syntactic targets. We introduce the first learning approach for this problem, building on the generality of neural encoder-decoder models to support a wide range of transformations. In doing

★Authors contributed equally.

Proceedings of NAACL-HLT 2018, pages 1875–1885
New Orleans, Louisiana, June 1 - 6, 2018.

so, we face two new challenges: (1) obtaining a large amount of paraphrase pairs for training, and (2) defining syntactic transformations with which to label these pairs.

Since no large-scale dataset of sentential paraphrases exists publicly, we follow Wieting et al. (2017) and automatically generate millions of paraphrase pairs using neural *backtranslation*. Backtranslation naturally injects linguistic variation between the original sentence and its backtranslated counterpart. By running the process at a very large scale and testing for the specific variations we want to produce, we can gather ample input-output pairs for a wide range of phenomena. Our focus is on syntactic transformations, which we define using templates derived from linearized constituency parses (§2). Given such parallel data, we can easily train an encoder-decoder model that takes a sentence and target syntactic template as input, and produces the desired paraphrase.[1]

A combination of automated and human evaluations show that the generated paraphrases almost always follow their target specifications, while paraphrase quality does not significantly deteriorate compared to vanilla neural backtranslation (§4). Our model, the syntactically controlled paraphrase network (SCPN), is capable of generating adversarial examples for sentiment analysis and textual entailment datasets that significantly impact the performance of pretrained models (Figure 1). We also show that augmenting training sets with such examples improves robustness without harming accuracy on the original test sets (§5). Together these results not only establish the first general purpose syntactically controlled paraphrase approach, but also suggest that this general paradigm could be used for controlling many other aspects of the target text.

2 Collecting labeled paraphrase pairs

In this section, we describe a general purpose process for gathering and labeling training data for controlled paraphrase generation.

2.1 Paraphrase data via backtranslation

Inducing paraphrases from bilingual data has long been an effective method to overcome data limitations. In particular, bilingual pivoting (Bannard and Callison-Burch, 2005) finds quality paraphrases by pivoting through a different language. Mallinson et al. (2017) show that neural machine translation (NMT) systems outperform phrase-based MT on several paraphrase evaluation metrics.

In this paper, we use the PARANMT-50M corpus from Wieting and Gimpel (2017). This corpus consists of over 50 million paraphrases obtained by backtranslating the Czech side of the CzEng (Bojar et al., 2016) parallel corpus. The pretrained Czech-English model used for translation came from the Nematus NMT system (Sennrich et al., 2017). The training data of this system includes four sources: Common Crawl, CzEng 1.6, Europarl, and News Commentary. The CzEng corpus is the largest of these four and was found to have significantly more syntactic diversity than the other data sources (Wieting and Gimpel, 2017).[2]

2.2 Automatically labeling paraphrases with syntactic transformations

We need labeled transformations in addition to paraphrase pairs to train a controlled paraphrase model. Manually annotating each of the millions of paraphrase pairs is clearly infeasible. Our key insight is that target transformations can be detected (with some noise) simply by parsing these pairs.[3]

Specifically, we parse the backtranslated paraphrases using the Stanford parser (Manning et al., 2014),[4] which yields a pair of constituency parses $\langle p_1, p_2 \rangle$ for each sentence pair $\langle s_1, s_2 \rangle$, where s_1 is the reference English sentence in the CzEng corpus and s_2 is its backtranslated counterpart. For syntactically controlled paraphrasing, we assume s_1 and p_2 are inputs, and the model is trained to produce s_2. To overcome learned biases of the NMT system, we also include reversed pairs $\langle s_2, s_1 \rangle$ during training.

2.2.1 Syntactic templates

To provide syntactic control, we linearize the bracketed parse structure without leaf nodes (i.e., tokens). For example, the corresponding linearized parse

[1] Code, labeled data, and pretrained models available at https://github.com/miyyer/scpn.

[2] Syntactic diversity was measured by the entropy of the top two levels of parse trees in the corpora.

[3] Similar automated filtering could be used to produce data for many other transformations, such as tense changes, point-of-view shifts, and even stylometric pattern differences (Feng et al., 2012). This is an interesting area for future work.

[4] Because of the large dataset size, we use the faster but less accurate shift-reduce parser written by John Bauer.

tree for the sentence "*She drove home.*" is `(S(NP(PRP))(VP(VBD)(NP(NN)))(.))`. A system that requires a complete linearized target parse at test-time is unwieldy; how do we go about choosing the target parse? To simplify test-time usage, we relax the target syntactic form to a parse *template*, which we define as the top two levels of the linearized parse tree (the level immediately below the root along with the root); the prior example's template is $\texttt{S} \rightarrow \texttt{NP VP}$. In the next section, we design models such that users can feed in either parse templates or full parses depending on their desired level of control.

3 Syntactically Controlled Paraphrase Networks

The SCPN encoder-decoder architecture is built from standard neural modules, as we describe in this section.

3.1 Neural controlled paraphrase generation

Given a sentential paraphrase pair $\langle s_1, s_2 \rangle$ and a corresponding target syntax tree p_2 for s_2, we encode s_1 using a bidirectional LSTM (Hochreiter and Schmidhuber, 1997), and our decoder is a two-layer LSTM augmented with soft attention over the encoded states (Bahdanau et al., 2014) as well as a copy mechanism (See et al., 2017). Following existing work in NMT (Sennrich et al., 2015), we preprocess s_1 and s_2 into subword units using byte pair encoding, and we perform decoding using beam search. For all attention computations, we use a bilinear product with a learned parameter matrix $\mathbf{W}$: given vectors $\boldsymbol{u}$ and $\boldsymbol{v}$, we score them by $\boldsymbol{u}^T \mathbf{W} \boldsymbol{v}$.

We incorporate the target syntax p_2 into the generation process by modifying the inputs to the decoder. In particular, a standard decoder LSTM receives two inputs at every time step: (1) the embedding $\boldsymbol{w}_{t-1}$ of the ground-truth previous word in s_2, and (2) an attention-weighted average $\boldsymbol{a}_t$ of the encoder's hidden states. We additionally provide a representation $\boldsymbol{z}_t$ of the target p_2, so at every time step the decoder computes

$$\boldsymbol{h}_t = \text{LSTM}([\boldsymbol{w}_{t-1}; \boldsymbol{a}_t; \boldsymbol{z}_t]). \quad (1)$$

Since we preserve bracketed parse structure, our linearized parses can have hundreds of tokens. Forcing all of the relevant information contained by the parse tree into a single fixed representation (i.e., the last hidden state of an LSTM) is difficult with such large sequences. Intuitively, we want the decoder to focus on portions of the target parse tree that correspond with the current time step. As such, we encode p_2 using a (unidirectional) LSTM and compute $\boldsymbol{z}_t$ with an attention-weighted average of the LSTM's encoded states at every time step. This attention mechanism is conditioned on the decoder's previous hidden state $\boldsymbol{h}_{t-1}$.

3.2 From parse templates to full parses

As mentioned in Section 2.2.1, user-friendly systems should be able to accept high-level parse templates as input rather than full parses. Preliminary experiments show that SCPN struggles to maintain the semantics of the input sentence when we replace the full target parse with templates, and frequently generates short, formulaic sentences. The paraphrase generation model seems to rely heavily on the full syntactic parse to determine output length and clausal ordering, making it difficult to see how to modify the SCPN architecture for template-only target specification.

Instead, we train another model with exactly the same architecture as SCPN to generate complete parses from parse templates. This allows us to do the prediction in two steps: first predict the full syntactic tree and then use that tree to produce the paraphrase. Concretely, for the first step, assume t_2 is the parse template formed from the top two levels of the target parse p_2. The input to this *parse generator* is the input parse p_1 and t_2, and it is trained to produce p_2. We train the parse generator separately from SCPN (i.e., no joint optimization) for efficiency purposes. At test time, a user only has to specify an input sentence and target template; the template is fed through the parse generator, and its predicted target parse is in turn sent to SCPN for paraphrase generation (see Figure 2).

3.3 Template selection and post-processing

By switching from full parses to templates, we have reduced but not completely removed the burden of coming up with a target syntactic form. Certain templates may be not be appropriate for particular input sentences (e.g., turning a long sentence with multiple clauses into a noun phrase). However, others may be too similar to the input syntax, resulting in very little change. Since template selection is not a major focus of this paper, we use a relatively simple procedure, selecting the twenty most frequent templates in PARANMT-

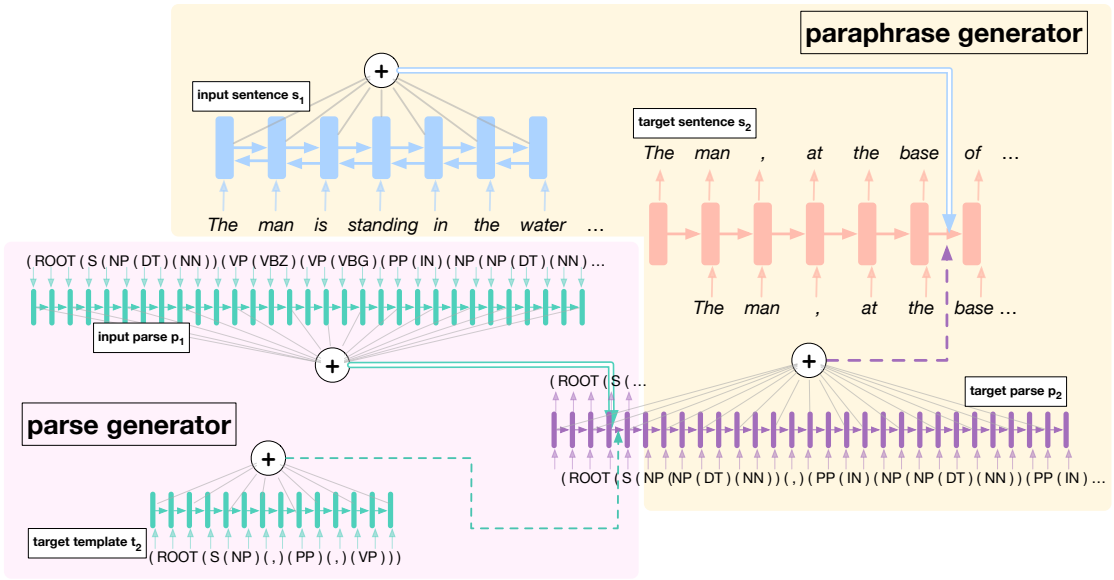


Figure 2: SCPN implements parse generation from templates as well as paraphrase generation from full parses as encoder-decoder architectures (attention depicted with dotted lines, copy mechanism with double stroked lines). While both components are trained separately, at test-time they form a pipelined approach to produce a controlled paraphrase from an input sentence s_1, its corresponding parse p_1, and a target template t_2.

50M.[5]

Since we cannot generate a valid paraphrase for every template, we postprocess to remove nonsensical outputs. In particular, we filter generated paraphrases using n-gram overlap and paraphrastic similarity, the latter of which is computed using the pretrained WORD,TRIAVG sentence embedding model from Wieting and Gimpel (2017).[6] These paraphrastic sentence embeddings significantly outperform prior work due to the PARANMT-50M data.

4 Intrinsic Experiments

Before using SCPN to generate adversarial examples on downstream datasets, we need to make sure that its output paraphrases are valid and grammatical and that its outputs follow the specified target syntax. In this section, we compare SCPN to a neural backtranslation baseline (NMT-BT) on the development set of our PARANMT-50M split using both human and automated experiments. NMT-BT is the same pretrained Czech-English model used to create PARANMT-50M; however, here we use it to generate in both directions (i.e., English-Czech and Czech-English).

[5]However, we do provide some qualitative examples of rare and medium-frequency templates in Table 3.

[6]After qualitatively analyzing the impact of different filtering choices, we set minimum n-gram overlap to 0.5 and minimum paraphrastic similarity to 0.7.

Model	**2**	**1**	**0**
SCPN w/ full parses	63.7	14.0	22.3
SCPN w/ templates	62.3	19.3	18.3
NMT-BT	65.0	17.3	17.7

Table 1: A crowdsourced paraphrase evaluation on a three-point scale (**0** = no paraphrase, **1** = ungrammatical paraphrase, **2** = grammatical paraphrase) shows both that NMT-BT and SCPN produce mostly grammatical paraphrases. Feeding parse templates to SCPN instead of full parses does not impact its quality.

4.1 Paraphrase quality & grammaticality

To measure paraphrase quality and grammaticality, we perform a crowdsourced experiment in which workers are asked to rate a paraphrase pair $\langle s, g \rangle$ on the three-point scale of Kok and Brockett (2010), where s is the source sentence and g is the generated sentence. A **0** on this scale indicates no paraphrase relationship, while **1** means that g is an ungrammatical paraphrase of s and **2** means that g is a grammatical paraphrase of s. We select 100 paraphrase pairs from the development set of our PARANMT-50M split (after the postprocessing steps detailed in Section 3.3) and have three workers rate each pair.[7] To focus the evaluation on the effect of syntactic manipulation on quality, we

[7]We use the Crowdflower platform for our experiments.

only select sentences whose top-level parse templates differ (i.e., $t_s \neq t_g$), ensuring that the output of both systems varies syntactically from the source sentences.

The results (Table 1) show that the uncontrolled NMT-BT model's outputs are comparable in quality and grammaticality to those of SCPN; neither system has a significant edge. More interestingly, we observe no quality drop when feeding templates to SCPN (via the parse generator as described in Section 3.2) instead of complete parse trees, which suggests that the parse generator is doing a good job of generating plausible parse trees; thus, for all of the adversarial evaluations that follow, we only use the templated variant of SCPN.

4.2 Do the paraphrases follow the target specification?

We next determine how often SCPN's generated paraphrases conform to the target syntax: if g is a generated paraphrase and p_g is its parse, how often does p_g match the ground-truth target parse p_2? We evaluate on our development set using *exact template match*: g is deemed a syntactic match to s_2 only if the top two levels of its parse p_g matches those of p_2. We evaluate two SCPN configurations, where one is given the full target parse p_2 and the other is given the result of running our parse generator on the target template t_2. As a sanity check, we also evaluate our parse generator using the same metric.

The results (Table 2) show that SCPN does indeed achieve syntactic control over the majority of its inputs. Our parse generator produces full parses that almost always match the target template; however, paraphrases generated using these parses are less syntactically accurate.[8] A qualitative inspection of the generated parses reveals that they can differ from the ground-truth target parse in terms of ordering or existence of lower-level constituents (Table 6); we theorize that these differences may throw off SCPN's decoder.

The NMT-BT system produces paraphrases that tend to be syntactically very similar to the input sentences: 28.7% of these paraphrases have the same template as that of the input sentence s_1, while only 11.1% have the same template as the ground-truth target s_2. Even though we train SCPN on data generated by NMT backtranslation, we avoid this issue by incorporating syntax into our learning process.

Model	Parse Acc.
SCPN w/ gold parse	64.5
SCPN w/ generated parse	51.6
Parse generator	99.9

Table 2: The majority of paraphrases generated by SCPN conform to the target syntax, but the level of syntactic control decreases when using generated target parses instead of gold parses. Accuracy is measured by exact template match (i.e., how often do the top two levels of the parses match).

5 Adversarial example generation

The intrinsic evaluations show that SCPN produces paraphrases of comparable quality to the uncontrolled NMT-BT system while also adhering to the specified target specifications. Next, we examine the utility of controlled paraphrases for adversarial example generation. To formalize the problem, assume a pretrained model for some downstream task produces prediction y_x given test-time instance x. An *adversarial* example x' can be formed by making label-preserving modifications to x such that $y_x \neq y_{x'}$. Our results demonstrate that controlled paraphrase generation with appropriate template selection produces far more valid adversarial examples than backtranslation on sentiment analysis and entailment tasks.

5.1 Experimental setup

We evaluate our syntactically adversarial paraphrases on the Stanford Sentiment Treebank (Socher et al., 2013, SST) and SICK entailment detection (Marelli et al., 2014). While both are relatively small datasets, we select them because they offer different experimental conditions: SST contains complicated sentences with high syntactic variance, while SICK almost exclusively consists of short, simple sentences. As a baseline, we compare the ten most probable beams from NMT-BT to controlled paraphrases generated by SCPN using ten templates randomly sampled from the template set described in Section 3.3.[9] We also need pretrained models

[8] With that said, exact match is a harsh metric; these paraphrases are more accurate than the table suggests, as often they differ by only a single constituent.

[9] We also experimented with the diverse beam search modification proposed by Li et al. (2016b) for NMT-BT but found that it dramatically warped the semantics of many beams; crowdsourced workers rated 49% of its outputs as **0**

template	paraphrase
original	with the help of captain picard , the borg will be prepared for everything .
(SBARQ(ADVP)(,)(S)(,)(SQ))	now , the borg will be prepared by picard , will it ?
(S(NP)(ADVP)(VP))	the borg here will be prepared for everything .
(S(S)(,)(CC)(S) (:)(FRAG))	with the help of captain picard , the borg will be prepared , and the borg will be prepared for everything ... for everything .
(FRAG(INTJ)(,)(S)(,)(NP))	oh , come on captain picard , the borg line for everything .
original	you seem to be an excellent burglar when the time comes .
(S(SBAR)(,)(NP)(VP))	when the time comes , you 'll be a great thief .
(S(``)(UCP)('')(NP)(VP))	" you seem to be a great burglar , when the time comes . " you said .
(SQ(MD)(SBARQ))	can i get a good burglar when the time comes ?
(S(NP)(IN)(NP)(NP)(VP)	look at the time the thief comes .

Table 3: Syntactically controlled paraphrases generated by SCPN for two examples from the PARANMT-50M development set. For each input sentence, we show the outputs of four different templates; the fourth template is a failure case (highlighted in green) exhibiting semantic divergence and/or ungrammaticality, which occurs when the target template is unsuited for the input.

for which to generate adversarial examples; we use the bidirectional LSTM baseline for both SST and SICK outlined in Tai et al. (2015) since it is a relatively simple architecture that has proven to work well for a variety of problems.[10] Since the SICK task involves characterizing the relationship between two sentences, for simplicity we only generate adversarial examples for the first sentence and keep the second sentence fixed to the ground truth.

5.2 Breaking pretrained models

For each dataset, we generate paraphrases for held-out examples and then run a pretrained model over them.[11] We consider a development example x *broken* if the original prediction y_x is correct, but the prediction $y_{x'}$ for at least one paraphrase x' is incorrect. For SST, we evaluate on the binary sentiment classification task and ignore all phrase-level labels (because our paraphrase models are trained on only sentences). Table 4 shows that for both datasets, SCPN breaks many more examples than NMT-BT. Moreover, as shown in Table 5, NMT-BT's paraphrases differ from the original example mainly by lexical substitutions, while SCPN often produces dramatically different syntactic structures.

5.3 Are the adversarial examples valid?

We have shown that we can break pretrained models with controlled paraphrases, but are these paraphrases actually valid adversarial examples? After all, it is possible that the syntactic modifications cause informative clauses or words (e.g., negations) to go missing. To measure the validity of our adversarial examples, we turn again to crowdsourced experiments. We ask workers to choose the appropriate label for a given sentence or sentence pair (e.g., positive or negative for SST), and then we compare the worker's judgment to the original development example's label. For both models, we randomly select 100 adversarial examples and have three workers annotate each one. The results (Table 4) show that on the more complex SST data, a higher percentage of SCPN's paraphrases are valid adversarial examples than those of NMT-BT, which is especially encouraging given our model also generates significantly *more* adversarial examples.

5.4 Increasing robustness to adversarial examples

If we additionally augment the *training* data of both tasks with controlled paraphrases, we can increase a downstream model's robustness to adversarial examples in the development set. To quantify this effect, we generate controlled paraphrases for the training sets of SST and SICK using the same templates as in the previous experiments. Then, we include these paraphrases as additional training examples and retrain our biLSTM task models.[12] As shown by Table 4, training on SCPN's paraphrases significantly improves robustness to syntactic adversaries without affecting accuracy on the original test sets. One im-

on the three-point scale.

[10] We initialize both models using pretrained GloVe embeddings (Pennington et al., 2014) and set the LSTM hidden dimensionality to 300.

[11] Since the SICK development dataset is tiny, we additionally generate adversarial examples on its test set.

[12] We did not experiment with more complex augmentation methods (e.g., downweighting the contribution of paraphrased training examples to the loss).

			No augmentation		With augmentation	
Model	Task	Validity	Test Acc	Dev Broken	Test Acc	Dev Broken
SCPN	SST	77.1	83.1	41.8	83.0	31.4
NMT-BT	SST	68.1	83.1	20.2	82.3	20.0
SCPN	SICK	77.7	82.1	33.8	82.7	19.8
NMT-BT	SICK	81.0	82.1	20.4	82.0	11.2

Table 4: SCPN generates more legitimate adversarial examples than NMT-BT, shown by the results of a crowd-sourced validity experiment and the percentage of held-out examples that are broken through paraphrasing. Furthermore, we show that by augmenting the training dataset with syntactically-diverse paraphrases, we can improve the robustness of downstream models to syntactic adversaries (see "Dev Broken" before and after augmentation) without harming accuracy on the original test set.

portant caveat is that this experiment only shows robustness to the set of templates used by SCPN; in real-world applications, careful template selection based on the downstream task, along with using a larger set of templates, is likely to increase robustness to less constrained syntactic adversaries. Augmentation with NMT-BT's paraphrases increases robustness on SICK, but on SST, it degrades test accuracy without any significant gain in robustness; this is likely due to its lack of syntactic variation compared to SCPN.

6 Qualitative Analysis

In the previous section, we quantitatively evaluated the SCPN's ability to produce valid paraphrases and adversarial examples. Here, we take a look at actual sentences generated by the model. In addition to analyzing SCPN's strengths and weaknesses compared to NMT-BT, we examine the differences between paraphrases generated by various configurations of the model to determine the impact of each major design decision (e.g., templates instead of full parses).

Syntactic manipulation: Table 3 demonstrates SCPN's ability to perform syntactic manipulation, showing paraphrases for two sentences generated using different templates. Many of the examples exhibit complex transformations while preserving both the input semantics and grammaticality, even when the target syntax is very different from that of the source (e.g., when converting a declarative to question). However, the failure cases demonstrate that not every template results in a valid paraphrase, as nonsensical outputs are sometimes generated when trying to squeeze the input semantics into an unsuitable target form.

Adversarial examples: Table 5 shows that SCPN and NMT-BT differ fundamentally in the type of adversaries they generate. While SCPN mostly avoids lexical substitution in favor of making syntactic changes, NMT-BT does the opposite. These examples reinforce the results of the experiment in Section 4.2, which demonstrates NMT-BT's tendency to stick to the input syntax. While SCPN is able to break more validation examples than NMT-BT, it is alarming that even simple lexical substitution can break such a high percentage of both datasets we tested.

Ebrahimi et al. (2017) observe a similar phenomenon with HotFlip, their gradient-based substitution method for generating adversarial examples. While NMT-BT does not receive signal from the downstream task like HotFlip, it also does not require external constraints to maintain grammaticality and limit semantic divergence. As future work, it would be interesting to provide this downstream signal to both NMT-BT and SCPN; for the latter, perhaps this signal could guide the template selection process, which is currently fixed to a small, finite set.

Templates vs. gold parses: Why does the level of syntactic control decrease when we feed SCPN parses generated from templates instead of gold parses (Table 2)? The first two examples in Table 6 demonstrate issues with the templated approach. In the first example, the template is not expressive enough for the parse generator to produce slots for the highlighted clause. A potential way to combat this type of issue is to dynamically define templates based on factors such as the length of the input sentence. In the second example, a parsing error results in an inaccurate template which in turn causes SCPN to generate a semantically-divergent paraphrase. The final two examples

template	original	paraphrase
(S(ADVP)(NP)(VP))	moody , heartbreaking , and filmed in a natural , unforced style that makes its characters seem entirely convincing even when its script is not .	so he 's filmed in a natural , unforced style that makes his characters seem convincing when his script is not .
(S(PP)(,)(NP)(VP))	there is no pleasure in watching a child suffer .	in watching the child suffer , there is no pleasure .
(S(S)(,)(CC)(S))	the characters are interesting and often very creatively constructed from figure to backstory .	the characters are interesting , and they are often built from memory to backstory .
	every nanosecond of the the new guy reminds you that you could be doing something else far more pleasurable .	each nanosecond from the new guy reminds you that you could do something else much more enjoyable .
	harris commands the screen , using his frailty to suggest the ravages of a life of corruption and ruthlessness .	harris commands the screen , using his weakness to suggest the ravages of life of corruption and recklessness .

Table 5: Adversarial sentiment examples generated by SCPN (top) and NMT-BT (bottom). The predictions of a pretrained model on the original sentences are correct (red is negative, blue is positive), while the predictions on the paraphrases are incorrect. The syntactically controlled paraphrases of SCPN feature more syntactic modification and less lexical substitution than NMT-BT's backtranslated outputs.

show instances where the templated model performs equally as well as the model with gold parses, displaying the capabilities of our parse generator.

Removing syntactic control: To examine the differences between syntactically controlled and uncontrolled paraphrase generation systems, we train an SCPN without including $\boldsymbol{z}_t$, the attention-weighted average of the encoded parse, in the decoder input. This uncontrolled configuration produces outputs that are very similar to its inputs, often identical syntactically with minor lexical substitution. Concretely, the uncontrolled SCPN produces a paraphrase with the same template as its input 38.6% of the time, compared to NMT-BT's 28.7% (Section 4.2).[13]

7 Related Work

Paraphrase generation (Androutsopoulos and Malakasiotis, 2010; Madnani and Dorr, 2010) has been tackled using many different methods, including those based on hand-crafted rules (McKeown, 1983), synonym substitution (Bolshakov and Gelbukh, 2004), machine translation (Quirk et al., 2004), and, most recently, deep learning (Prakash et al., 2016; Mallinson et al., 2017; Dong et al., 2017). Our syntactically controlled setting also relates to controlled language generation tasks in which one desires to generate or rewrite a sentence with particular characteristics. We review related work in both paraphrase generation and controlled language generation below.

7.1 Data-driven paraphrase generation

Madnani and Dorr (2010) review data-driven methods for paraphrase generation, noting two primary families: template-based and translation-based. The first family includes approaches that use hand-crafted rules (McKeown, 1983), thesaurus-based substitution (Bolshakov and Gelbukh, 2004; Zhang and LeCun, 2015), lattice matching (Barzilay and Lee, 2003), and template-based "shake & bake" paraphrasing (Carl et al., 2005). These methods often yield grammatical outputs but they can be limited in diversity.

The second family includes methods that rewrite the input using methods based on parallel text (Bannard and Callison-Burch, 2005), machine translation (Quirk et al., 2004; Napoles et al., 2016; Suzuki et al., 2017), or related statistical techniques (Zhao et al., 2009). Of particular relevance to our work are methods that incorporate syntax to improve fluency of paraphrase output. Callison-Burch (2008) constrains paraphrases to be the same syntactic type as the input, though he was focused on phrase-level, not sentential, paraphrasing. Pang et al. (2003) learn finite-state automata from translation pairs that generate syntactic paraphrases, though this requires multiple translations into the same language and cannot be used to generate paraphrases outside this dataset. Shen et al. (2006) extend this to deeper syntactic analysis. All of these approaches use syntax to

[13] A configuration without the copy mechanism copies input syntax even more, with a 47.7% exact template match.

template	(S(CC)(S)(,)(NP)(ADVP)(VP))
original	damian encouraged me , criticized , he ... he always made me go a little deeper .
SCPN parse	but damian , he supported me , he told me , he always made me go a little deeper .
SCPN template	but damian supported me , he always made me go a little deeper .
template	(S(S)(,)(NP)(VP))
original	zacharias did n't deserve to die , grishanov thought , and he was aware of the huge irony of his situation
SCPN parse	zacharias did not deserve to die , grishanov told himself , realizing the greatest irony of all .
SCPN template	zacharias did not deserve to die , he was aware of the great irony of his situation .
template	S(S)(,)(S))
original	give me some water , my lips are dry , and i shall try to tell you .
SCPN parse	give me some water , i have just a dry mouth .
SCPN template	give me some water , my lips are dry .
template	(S(NP)(,)(ADVP)(,)(VP))
original	in the meantime , the house is weakened , and all its old alliances and deals are thrown into doubt .
SCPN parse	the house , meanwhile , is weakening , which will be all of its old alliances and business .
SCPN template	the house , meanwhile , is weakened , and its old alliances and deals are thrown into doubt .

Table 6: Examples from PARANMT-50M comparing the output of two SCPN configurations, one with gold target parses (SCPN parse) and one with parses generated from templates (SCPN template), where templates are the top two levels of the gold parses. The first two examples demonstrate issues with missing information caused by inexpressive templates and parsing errors, respectively. The remaining examples, in which both configurations produce syntactically similar paraphrases, showcase the ability of the parse generator to produce viable full parses.

improve grammaticality, which is handled by our decoder language model.

Recent efforts involve neural methods. Iyyer et al. (2014) generate paraphrases with dependency tree recursive autoencoders by randomly selecting parse trees at test time. Li et al. (2017) generate paraphrases using deep reinforcement learning. Gupta et al. (2017) use variational autoencoders to generate multiple paraphrases. These methods differ from our approach in that none offer fine-grained control over the syntactic form of the paraphrase.

7.2 Controlled language generation

There is growing interest in generating language with the ability to influence the topic, style, or other properties of the output.

Most related to our methods are those based on syntactic transformations, like the tree-to-tree sentence simplification method of Woodsend and Lapata (2011) based on quasi-synchronous grammar (Smith and Eisner, 2006). Our method is more general since we do not require a grammar and there are only soft constraints. Perhaps the closest to the proposed method is the conditioned recurrent language model of Ficler and Goldberg (2017), which produces language with user-selected properties such as sentence length and formality but is incapable of generating paraphrases.

For machine translation output, Niu et al. (2017) control the level of formality while Sennrich et al. (2016) control the level of politeness. For dialogue, Li et al. (2016a) affect the output using speaker identity, while Wang et al. (2017) develop models to influence topic and style of the output. Shen et al. (2017) perform style transfer on non-parallel texts, while Guu et al. (2017) generate novel sentences from prototypes; again, these methods are not necessarily seeking to generate meaning-preserving paraphrases, merely transformed sentences that have an altered style.

8 Conclusion

We propose SCPN, an encoder-decoder model for syntactically controlled paraphrase generation, and show that it is an effective way of generating adversarial examples. Using a parser, we label syntactic variation in large backtranslated data, which provides training data for SCPN. The model exhibits far less lexical variation than existing uncontrolled paraphrase generation systems, instead preferring purely syntactic modifications. It is capable of generating adversarial examples that fool pretrained NLP models. Furthermore, by training on such examples, we increase the robustness of these models to syntactic variation.

Acknowledgments

We thank the reviewers for their insightful comments. We would also like to thank Mark Yatskar for many useful suggestions on our experiments.

References

Ion Androutsopoulos and Prodromos Malakasiotis. 2010. A survey of paraphrasing and textual entailment methods. *Journal of Artificial Intelligence Research* 38.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the Association for Computational Linguistics*.

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: an unsupervised approach using multiple-sequence alignment. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Ondřej Bojar, Ondřej Dušek, Tom Kocmi, Jindřich Libovický, Michal Novák, Martin Popel, Roman Sudarikov, and Dušan Variš. 2016. CzEng 1.6: Enlarged Czech-English Parallel Corpus with Processing Tools Dockered. In *Text, Speech, and Dialogue: 19th International Conference, TSD 2016*.

Igor Bolshakov and Alexander Gelbukh. 2004. Synonymous paraphrasing using WordNet and Internet. *Natural Language Processing and Information Systems* .

Chris Callison-Burch. 2008. Syntactic constraints on paraphrases extracted from parallel corpora. In *Proceedings of Empirical Methods in Natural Language Processing*.

Michel Carl, Paul Schmidt, and Jörg Schütz. 2005. Reversible template-based shake & bake generation. In *MT Summit X*.

Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. 2017. Learning to paraphrase for question answering. In *Proceedings of Empirical Methods in Natural Language Processing*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2017. Hotflip: White-box adversarial examples for nlp. *arXiv preprint arXiv:1712.06751* .

Allyson Ettinger, Sudha Rao, Hal Daumé III, and Emily M Bender. 2017. Towards linguistically generalizable nlp systems: A workshop and shared task. In *Proceedings of the First Workshop on Building Linguistically Generalizable NLP Systems*.

Song Feng, Ritwik Banerjee, and Yejin Choi. 2012. Characterizing stylistic elements in syntactic structure. In *Proceedings of Empirical Methods in Natural Language Processing*.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. *arXiv preprint arXiv:1707.02633* .

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *Proceedings of the International Conference on Learning Representations*.

Ankush Gupta, Arvind Agarwal, Prawaan Singh, and Piyush Rai. 2017. A deep generative framework for paraphrase generation. *arXiv preprint arXiv:1709.05074* .

Kelvin Guu, Tatsunori B Hashimoto, Yonatan Oren, and Percy Liang. 2017. Generating sentences by editing prototypes. *arXiv preprint arXiv:1709.08878* .

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* .

Mohit Iyyer, Jordan Boyd-Graber, and Hal Daumé III. 2014. Generating sentences from semantic vector space representations. In *NIPS Workshop on Learning Semantics*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of Empirical Methods in Natural Language Processing*.

Stanley Kok and Chris Brockett. 2010. Hitting the right paraphrases in good time. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016a. A persona-based neural conversation model. In *Proceedings of the Association for Computational Linguistics*.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016b. A simple, fast diverse decoding algorithm for neural generation. *arXiv preprint arXiv:1611.08562* .

Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. 2017. Paraphrase generation with deep reinforcement learning. *arXiv preprint arXiv:1711.00279* .

Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. 2017. Deep text classification can be fooled. *arXiv preprint arXiv:1704.08006* .

Nitin Madnani and Bonnie J Dorr. 2010. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics* 36(3).

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2017. Paraphrasing revisited with neural machine translation. In *Proceedings of the European Chapter of the Association for Computational Linguistics*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics System Demonstrations*.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. *SemEval 2014* .

Kathleen R McKeown. 1983. Paraphrasing questions using given and new information. *Computational Linguistics* 9(1).

Courtney Napoles, Chris Callison-Burch, and Matt Post. 2016. Sentential paraphrasing as black-box machine translation. In *Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*.

Xing Niu, Marianna Martindale, and Marine Carpuat. 2017. A study of style in machine translation: Controlling the formality of machine translation output. In *Proceedings of Empirical Methods in Natural Language Processing*.

Bo Pang, Kevin Knight, and Daniel Marcu. 2003. Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of Empirical Methods in Natural Language Processing*.

Aaditya Prakash, Sadid A Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2016. Neural paraphrase generation with stacked residual LSTM networks. In *Proceedings of International Conference on Computational Linguistics*.

Chris Quirk, Chris Brockett, and William Dolan. 2004. Monolingual machine translation for paraphrase generation. In *Proceedings of Empirical Methods in Natural Language Processing*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the Association for Computational Linguistics*.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, et al. 2017. Nematus: a toolkit for neural machine translation. *arXiv preprint arXiv:1703.04357* .

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. In *Proceedings of the Association for Computational Linguistics*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Controlling politeness in neural machine translation via side constraints. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Siwei Shen, Dragomir R Radev, Agam Patel, and Güneş Erkan. 2006. Adding syntax to dynamic programming for aligning comparable texts for the generation of paraphrases. In *Proceedings of International Conference on Computational Linguistics*.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Proceedings of Advances in Neural Information Processing Systems*.

David A Smith and Jason Eisner. 2006. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proceedings of the Workshop on Statistical Machine Translation*.

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

Yui Suzuki, Tomoyuki Kajiwara, and Mamoru Komachi. 2017. Building a non-trivial paraphrase corpus using multiple machine translation systems. In *Proceedings of ACL Student Research Workshop*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the Association for Computational Linguistics*.

Di Wang, Nebojsa Jojic, Chris Brockett, and Eric Nyberg. 2017. Steering output style and topic in neural response generation. In *Proceedings of Empirical Methods in Natural Language Processing*.

John Wieting and Kevin Gimpel. 2017. Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. *arXiv preprint arXiv:1711.05732* .

John Wieting, Jonathan Mallinson, and Kevin Gimpel. 2017. Learning paraphrastic sentence embeddings from back-translated bitext. In *Proceedings of Empirical Methods in Natural Language Processing*.

Kristian Woodsend and Mirella Lapata. 2011. Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of Empirical Methods in Natural Language Processing*.

Xiang Zhang and Yann LeCun. 2015. Text understanding from scratch. *arXiv preprint arXiv:1502.01710* .

Shiqi Zhao, Xiang Lan, Ting Liu, and Sheng Li. 2009. Application-driven statistical paraphrase generation. In *Proceedings of the Association for Computational Linguistics*.

Sentiment Analysis: It's Complicated!

[1*]Kian Kenyon-Dean, [1*]Eisha Ahmed, [1†]Scott Fujimoto, [1†]Jeremy Georges-Filteau, [1†]Christopher Glasz, [1†]Barleen Kaur, [1†]Auguste Lalande, [1#]Shruti Bhanderi, [1#]Robert Belfer, [1#]Nirmal Kanagasabai, [1#]Roman Sarrazingendron, [1#]Rohit Verma, and [2]Derek Ruths

[1,2]McGill University, Department of Computer Science

[1] {first.last}@mail.mcgill.ca

[2] derek.ruths@mcgill.ca

Abstract

Sentiment analysis is used as a proxy to measure human emotion, where the objective is to categorize text according to some predefined notion of sentiment. Sentiment analysis datasets are typically constructed with gold-standard sentiment labels, assigned based on the results of manual annotations. When working with such annotations, it is common for dataset constructors to discard "noisy" or "controversial" data where there is significant disagreement on the proper label. In datasets constructed for the purpose of Twitter sentiment analysis (TSA), these controversial examples can compose over 30% of the originally annotated data. We argue that the removal of such data is a problematic trend because, when performing real-time sentiment classification of short-text, an automated system cannot know *a priori* which samples would fall into this category of disputed sentiment. We therefore propose the notion of a "complicated" class of sentiment to categorize such text, and argue that its inclusion in the short-text sentiment analysis framework will improve the quality of automated sentiment analysis systems as they are implemented in real-world settings. We motivate this argument by building and analyzing a new publicly available TSA dataset of over 7,000 tweets annotated with 5x coverage, named MTSA. Our analysis of classifier performance over our dataset offers insights into sentiment analysis dataset and model design, how current techniques would perform in the real world, and how researchers should handle difficult data.

1 Introduction

The goal of sentiment analysis is to determine the attitude or emotional state held by the author of a piece of text. Automatic sentiment classification that can quickly garner user sentiment is useful for applications ranging from product marketing to measuring public opinion. The volume and availability of short-text user content makes automated sentiment analysis systems highly attractive for companies and organizations, despite potential complications arising from their short length and specialized use of language. The popularity of Twitter as a social media platform on which people can readily express their thoughts, feelings, and opinions, coupled with the openness of the platform, provides a large amount of publicly accessible data ripe for analysis, being a well established domain for sentiment analysis as reflecting real-world attitudes (Pak and Paroubek, 2010; Bollen et al., 2011). In this paper, we look into Twitter sentiment analysis (TSA) as a suitable, core instance of general short-text sentiment analysis (Thelwall et al., 2010, 2012; Kiritchenko et al., 2014; Dos Santos and Gatti, 2014), and encourage the methods and practices presented to be applied across other domains.

Tweet text	+	-	0
Members came in today for lunch to learn more about competitive events.	0	0	5
15 year old with an iPhone X, like DAMN girl, Whatcha gonna do with that much power in your hands? Facebook? Snapchat? That's it?	0	2	3
i am really missing the food my family makes rn	2	2	1

Table 1: Example tweets from our dataset over varying levels of annotator labellings; **+**, **-**, **0** stand for POSITIVE, NEGATIVE, OBJECTIVE.

Building a TSA model that can automatically

[*]These authors contributed equally to this work.
[†]These authors contributed equally to this work.
[#]These authors contributed equally to this work.

Proceedings of NAACL-HLT 2018, pages 1886–1895
New Orleans, Louisiana, June 1 - 6, 2018.

determine the sentiment of a tweet has received significant attention over the past several years. However, since most state-of-the-art TSA models use machine learning to tune their parameters, their performance – and relevance to a real-world implementation setting – is highly dependent on the dataset on which they are trained.

TSA dataset construction has, unfortunately, received less attention than TSA model design. Many commonly used TSA datasets make assumptions that do not hold in a real-world implementation setting. For example, it is a common practice for studies to discard tweets on which there is high annotator disagreement. While some argue that this is done to remove noise resulting from poor annotator quality, this argument does not hold when considering that these datasets present high rates of unanimous annotator agreement[1]. This suggests that the problem is not poor annotators, but, rather, difficult data that does not fall into the established categories of sentiment.

Consider the sample tweets in Table 1 drawn from our dataset, one with unanimous agreement on an OBJECTIVE label, one with 60% agreement, and one with complete disagreement. We observe that, as the amount of disagreement across annotations increases, so too does the clarity of what the tweet's gold standard label really should be. Though the issues we raise may seem obvious, the absence of their proper treatment in the existing literature suggests the need to systematically consider their implications in sentiment analysis.

In this paper, we propose the inclusion of a COMPLICATED class of sentiment to indicate that the text does not fall into the established categories of sentiment. We offer insights into the differences between tweets that receive different levels of inter-annotator-agreement, providing empirical evidence that tweets with differing levels of agreement are qualitatively different from each other.

Our claims are supported by empirical analysis of a new TSA dataset, the McGill Twitter Sentiment Analysis dataset (MTSA), which we release publicly with this work[2]. The dataset contains 7,026 tweets across five different topic-domains, annotated with 5x coverage. We release this dataset with the raw annotation results, and hope that researchers and organizations will be able to analyze our dataset and build models that can be applied in real-world sentiment analysis settings.

2 Current Problems in TSA

The field of Twitter Sentiment Analysis (TSA) has seen a considerable productive work over the past several years, and several large reviews and surveys have been written to highlight the trends and progress of the field, its datasets, and the methods used for building automatic TSA systems (Saif et al., 2013; Medhat et al., 2014; Martínez-Cámara et al., 2014; Giachanou and Crestani, 2016).

There are a variety of methods for constructing TSA datasets along a variety of domains, ranging from very specific (e.g., OMD (Shamma et al., 2009)) to general (e.g., SemEval 2013-2014 (Nakov et al., 2016)). While there is the popular Stanford Twitter corpus, constructed with noisy labellings (Go et al., 2009), the more common method of constructing TSA datasets relies on manual annotation (usually crowd-sourced) of tweet sentiment to establish gold-standard labellings according to a pre-defined set of possible label categories (often POSITIVE, NEGATIVE, and NEUTRAL) (Shamma et al., 2009; Speriosu et al., 2011; Thelwall et al., 2012; Saif et al., 2013; Nakov et al., 2016; Rosenthal et al., 2017).

One of the earliest manually annotated TSA datasets, the Obama-McCain Debate (OMD) (Shamma et al., 2009) was released with the specific annotator votes for each tweet, rather than a final specific label assignment. Nonetheless, most work on this dataset filters out tweets with less than two-thirds agreement (Speriosu et al., 2011; Saif et al., 2013) (Table 2). Unfortunately, many later dataset releases have not followed the example of the OMD; the designers of such datasets have opted instead to release only the resultant labelling according to a motivated (but constraining) label-assignment schema, often removing tweets with high inter-annotator disagreement from the final dataset release (Saif et al., 2013; Nakov et al., 2016; Rosenthal et al., 2017).

The assumptions and implications resulting from such design choices should be carefully considered by researchers before deciding on how to construct or analyze sentiment analysis datasets. Indeed, a current limitation in the field is the lack of attention paid to label-assignment schemes, which ultimately determine the gold-standard labellings of samples. We argue that researchers

[1] Annotator disagreement information has proven useful in other areas of sentiment analysis (Wilson et al., 2005).

[2] Download at https://github.com/networkdynamics/mcgill-tsa

Name	# Annotated	Discarded	Coverage	Labels	Ref.
OMD	3,238	1,087 (33.6%)[3]	3x	**+**, **-**, MIXED, OTHER	(Shamma et al., 2009)
STS-Gold	3,000	794 (26.5%)	3x	**+**, **-**, **0**, MIXED, OTHER	(Saif et al., 2013)
SemEval 2013-14 (B)	17,048	Unknown	5x	**+**, **-**, **0**	(Nakov et al., 2016)
SemEval 2017	12,379	None (0%)	5x	**s+**, **w+**, **w-**, **s-**, **0**	(Rosenthal et al., 2017)

Table 2: Summary of some of the major TSA datasets used in recent work. Symbols **+**, **-**, **0** stand for POSITIVE, NEGATIVE, and NEUTRAL, respectively; prefixes **s**, **w** stand for STRONGLY and WEAKLY.

should consider whether or not the choices made during dataset construction adequately reflect a situation in which automatic sentiment analysis systems would be used in real-world settings.

2.1 General Trends in TSA Datasets

In the SemEval 2017 Task 4 (Rosenthal et al., 2017), a thorough 5x coverage annotation scheme is used (each tweet is annotated by at least five people). Annotations were made on a five-point scale, with categories STRONGLYNEGATIVE, WEAKLYNEGATIVE, NEUTRAL, WEAKLYPOSITIVE, and STRONGLYPOSITIVE. If at least three out of five of the annotators gave the same labelling, that was accepted as the final annotation. Otherwise, the authors used an averaging scheme (mapping the labels to integers $-2, -1, 0, 1, 2$) to determine the final label, taking the average of the labellings and rounding according to a specific criterion. This is highly problematic. For example, if a controversial tweet receives two STRONGLYNEGATIVE, two STRONGLYPOSITIVE, and one NEUTRAL labelling, it will have a resultant label of NEUTRAL. Yet, the tweet would certainly not be "neutral", it would be qualitatively different from a tweet with unanimous agreement on a NEUTRAL labelling. In Section 5, we provide empirical results supporting this claim, discovering that high-disagreement data is qualitatively different from high-agreement data.

Nakov et al. (2016) provide a thorough exploration into the specific design decisions and considerations made during the construction of the 2013-2014 SemEval shared task for short-text sentiment analysis. In Subtask B, annotators determined the overall polarity of a piece of text, according to a ternary labelling scheme between POSITIVE, NEGATIVE, or NEUTRAL. The final label of the sentence was "determined based on the majority of the labels" according to 5x coverage. The designers thus discarded sentences where there was no majority annotator agreement, since such sentences "are likely to be controversial cases" (p. 40); they do not report how much data was discarded.

Saif et al. (2013) constructed a new dataset, the STS-Gold, by taking into account several limitations of the TSA datasets they reviewed. In their study, 3,000 tweets were labelled with 3x coverage. Any tweet without unanimous agreement on the label was discarded; this decision was justified by the argument that they did not want "noisy data" in their dataset. Thus, they discarded 794 tweets, or 26.5% of their originally annotated data. While we argue that this is a problematic design decision, we note that discarding data in this way successfully isolated unanimous-agreement from majority-agreement data, thus avoiding conflating tweets with different levels of agreement, unlike in the 2013-14 and 2017 SemEval tasks.

The annotation scheme for the STS-Gold resolves one of the problems in the SemEval 2017 Task, as it provides an option for labelling a MIXED category, capturing tweets bearing multiple conflicting sentiments. It also provides the OTHER category for tweets where it is "difficult to decide on a proper label". Interestingly, the dichotomy between the high frequency of high-disagreement tweets (794 total) compared to the low frequency of tweets unanimously labelled as OTHER (4 total) is consistent with our findings on the COMPLICATED label (Section 3.3).

[3]Note that the entire OMD dataset was released with annotator votes, but most studies remove that proportion tweets where there was not at least two-thirds agreement on label.

The challenges and possible approaches to manual sentiment annotation have been previously discussed by Mohammad (2016), who offers important insights into how questions and problem descriptions should be posed to annotators.

2.2 Summary of TSA Problems

Based on analysis of the design choices of the three datasets described above, and on the thorough overview of other datasets found in (Saif et al., 2013), we conclude that there are two primary limitations in the standard TSA datasets.

First, the lack of distinction between data with majority- vs. unanimous-agreement on the annotated label (Nakov et al., 2016; Rosenthal et al., 2017). In the analysis (Sections 3.3 and 6) of our TSA dataset, we observe a clear qualitative difference between majority-agreement and unanimous-agreement data, suggesting that these sets of data should not necessarily be treated in the same way.

Second, the systematic removal of controversial (or, high-disagreement) data (Saif et al., 2013; Nakov et al., 2016). We argue that this tendency is problematic because any automatic sentiment analysis system to be implemented in a real-world setting cannot know *a priori* which tweets will be "noisy" or "controversial". An automatic sentiment analysis system trained on such a dataset will inevitably mislabel such tweets as they appear in a real-world implementation setting.

We therefore suggest that the following paradigm become the norm in the field: in releasing sentiment analysis datasets, *researchers should provide the specific annotations obtained for each sample* (as was done by Shamma et al. (2009)), in addition to the resultant labelling based on the label-assignment scheme they decide upon. Additionally, *data with high levels of annotator disagreement should not be discarded*, rather, it should be included in dataset releases.

3 Building the MTSA Dataset

The absence of a TSA dataset containing raw annotations and sufficient coverage to identify sources of annotator disagreement necessitated the creation of a new annotated dataset. Here, we provide an overview of the development of a new McGill TSA (MTSA) dataset composed of 7,026 tweets annotated with 5x coverage.

Topic	Count	% of Total
Sports	1752	24.9
Food	1729	24.6
Media	1697	24.2
Commercial Tech.	1353	19.3
General	495	7.0
Total	**7026**	100

Table 3: Distribution of annotated tweets by topic.

3.1 Data Collection

Tweets were collected from Twitter's streaming API, filtered for English tweets that contained at least one English token, that were posted by users in North American time-zones. Each tweet had to contain at least one keyword from a topic cloud relating to *Food* (example keywords: "weight", "breakfast", "protein"), *Media* ("cinema", "gameofthrones", "reggae"), *Commercial Technology* ("microsoft", "laptop", "iphone"), or *Sports* ("spurs", "hockey", "habs"). Using this topic cloud and a diverse set of keywords per topic (average of 38 hand-selected keywords per topic), we collected tweets with the intent to represent the general sentiment surrounding a specific topic, while reducing the bias that would result by relying on a single topic or keyword. A further subset of tweets (categorized as *General*) was collected from the stream, without any keyword filters, in order to further broaden the representative scope of our dataset.

We additionally filtered out tweets containing external links or images, arguing that analysis of these multimodal tweets is a separate problem, belonging to the domain of Multimodal Sentiment Analysis (Poria et al., 2016; Soleymani et al., 2017). After the entire filtering process[4] was complete, we obtained 7,026 tweets across the different topics, which would be annotated with 5x coverage. The distribution of these tweets is seen below in Table 3.

3.2 Data Annotation

Data annotation was crowd-sourced using the CrowdFlower platform[5]. All qualified CrowdFlower contributors had the opportunity to complete the task, which was presented as: carefully

[4]See supplemental material for full enumeration of the specific filters used and the keywords used for each topic.

[5]Website: https://www.crowdflower.com

read the tweet, determine whether or not it expresses sentiment (e.g., OBJECTIVE or not), if it does, categorize the sentiment as being either POSITIVE, NEGATIVE, or COMPLICATED. In the instructions, COMPLICATED was presented as the preferable option when the sentiment expressed in the tweet was ambiguous, mixed or could be interpreted as both positive and/or negative. After a one-line description of the meaning of each category, the contributor was presented with examples of tweets belonging in each category before starting the task.

In order to be considered qualified to complete the task, the contributor had to correctly answer at least 8 of 10 test questions, which we manually selected and labelled. When a user failed a test question, they were presented with the correct answer and a corresponding justification to ensure that they understood the task.

We experimented with the inclusion of test questions from the COMPLICATED category during screening, and found that this was a major source of protest among high-quality annotators. Indeed, it may be paradoxical to expect annotators to agree on tweets that cause significant disagreement. Furthermore, due to the heterogeneous nature of this class, such test questions would risk biasing the annotators' notion of the category. As such, we limited our test questions to OBJECTIVE, POSITIVE, and NEGATIVE tweets.

Users who successfully passed the initial test questions annotated a maximum of 400 tweets. Of those tweets, 10% were additional hidden test questions to continuously assess the quality of the annotators; an accuracy of at least 80% on these test questions was the threshold for including their annotations in the dataset. In the end, a total of 35,926 tasks were completed by 181 trusted contributors, resulting in 7,026 annotated tweets.

3.3 Dataset Analysis

The annotated tweets are categorized by four agreement levels: *Unanimous* (5 out of 5 agreed on the label), *Consensus* (exactly 4 out of 5 agreed), *Majority* (exactly 3 out of 5 agreed), or *Disputed* (maximum 2 out of 5 agreed). The distribution of agreement rates was consistent across topics (see supplemental material), thus the entire dataset is merged for the remainder of the analysis.

Annotator agreement distribution. Tweets with at least *Consensus* agreement compose 64% of the dataset (4505 tweets), and tweets with at least *Majority* agreement compose 92% of the dataset (6473 tweets; see Table 4). The decision to discard tweets with significant annotator disagreement, as previously done in TSA research, would result in the loss of 8% to 34% of the annotated tweets in our dataset, depending on whether to filter to a minimum *Majority* or *Consensus* agreement, respectively. Interestingly, these numbers are consistent with the proportion of discarded tweets in previous literature (Table 2).

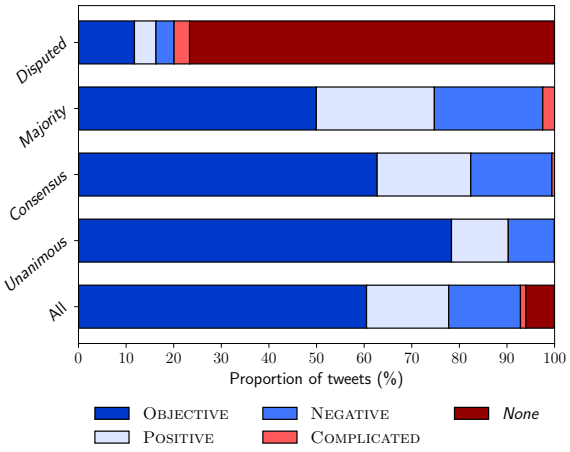


Figure 1: Most frequent annotation by annotator agreement rates. Of the 553 *Disputed* tweets, only 129 had a single most frequent annotation.

Sentiment and annotator agreement. Tweets that caused more disagreement among the human annotators were found to be more sentiment-laden (majority label of POSITIVE, NEGATIVE, or COMPLICATED; Figure 1). Objective tweets composed 78% (1892 tweets), 63% (1311), and 50% (983) of the *Unanimous*, *Consensus*, and *Majority* subsets of annotated tweets, respectively.

COMPLICATED label usage. Use of the COMPLICATED label by annotators was infrequent, and of those tweets with high inter-annotator agreement, almost exclusively limited to tweets that expressed clear, mixed sentiment. For example, the single tweet that received a unanimous COMPLICATED annotation had clear mixed sentiment: "the iPhone 6s is so big and hard to use but I still like it". There were a total of 13 tweets with at least *Consensus* agreement for the COMPLICATED label (see supplemental material). These specific tweets largely corresponded to the MIXED label used in previous TSA datasets (Shamma et al., 2009; Saif et al., 2013). Other types of ambiguous

Agreement	Count	% of Total
Unanimous	2415	34.4
Consensus	2090	29.7
Majority	1968	28.0
Disputed	553	7.9
Total	7026	100

Table 4: Annotator agreement rates. *Unanimous* stands for 100% annotator agreement, *Consensus* 80%, *Majority* 60%, and *Disputed* <60%.

Label	Count	% of Total
OBJECTIVE	4186	59.6
POSITIVE	1187	16.9
NEGATIVE	1038	14.8
COMPLICATED	62	0.9
Disputed	553	7.9
Total	7026	100

Table 5: Distribution of tweets across classes, where the label given is the result of majority vote.

tweets that did not clearly fall clearly within OBJECTIVE, POSITIVE, and NEGATIVE categories were not consistently identified as COMPLICATED by annotators. Rather, those tweets were a source of significant disagreement.

4 Classifying Tweet Sentiment

Here, we present the construction of shallow classifier and the experiments performed to study the phenomenon of annotator disagreement. Our objective was not to build a state-of-the-art classifier with optimal accuracy rates, rather, we sought to understand how the inclusion or exclusion of tweet subsets based on annotator disagreement impacts classification accuracy.

4.1 Preprocessing and Feature Extraction

To use machine learning methods with textual data, it is necessary to represent the data in a vector space such that each sample has the same dimensionality, despite varying sequence lengths. We concatenated three different standard feature extraction methods to build vector representations of tweets: N-Grams (unigrams and bigrams), mean word embedding (GLoVE embeddings built from twitter data (Pennington et al., 2014)[6]), and SentiWordNet scores (Baccianella et al., 2010).[7]

4.2 Experimental Design

As described in Section 2, most recent work in TSA has agglomerated tweets together based on the majority labelling. For example, a tweet annotated with a *Majority* agreement labelling (e.g., 3 OBJECTIVE and 2 NEGATIVE) would be given the label OBJECTIVE, just as one with *Unanimous* agreement on an OBJECTIVE labelling. In our experiments with our collected dataset (Section 3) we seek to determine whether or not there is a qualitative difference between high- versus low-agreement data.

Experiment I. In the first experiment setting, we agglomerate tweets according to the traditional practice for assigning labels based on annotations (Section 2); e.g., we remove tweets with at least a *Majority* voted label as COMPLICATED, and remove the *Disputed* tweets (that is, we remove 8.75% of our annotated data for these experiments), creating a 3-class classification problem. We experiment over four different sets of our data in this scenario: the full dataset (minus the COMPLICATED 8.75%); tweets with exactly *Majority* agreement; tweets with exactly *Consensus* agreement; and tweets with exactly *Unanimous* agreement on the label (see Figure 1 for the label distributions over each of these subsets). Additionally, when making predictions on a specific subset, we present results from training solely on the subset versus training on all of the data in this setting.

Experiment II. In the second experiment setting, we sought to determine the impact of including controversial samples, making a 4-class classification problem. Samples that were labelled with at least *Majority* agreement on a COMPLICATED label, and samples with *Disputed* agreement, were all assigned the label COMPLICATED. We thus used the entirety of our dataset for this experiment, where the COMPLICATED class accounted for 8.75% (615) of the samples, with the rest of the samples being given the majority-vote labelling.

Methods. For both experiments, we use a logistic regression classifier with balanced training set class weights, using the feature set described in Section 4.1. Preliminary experiments with fea-

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

[7] See supplemental material for full elaboration of the preprocessing decisions and features extracted.

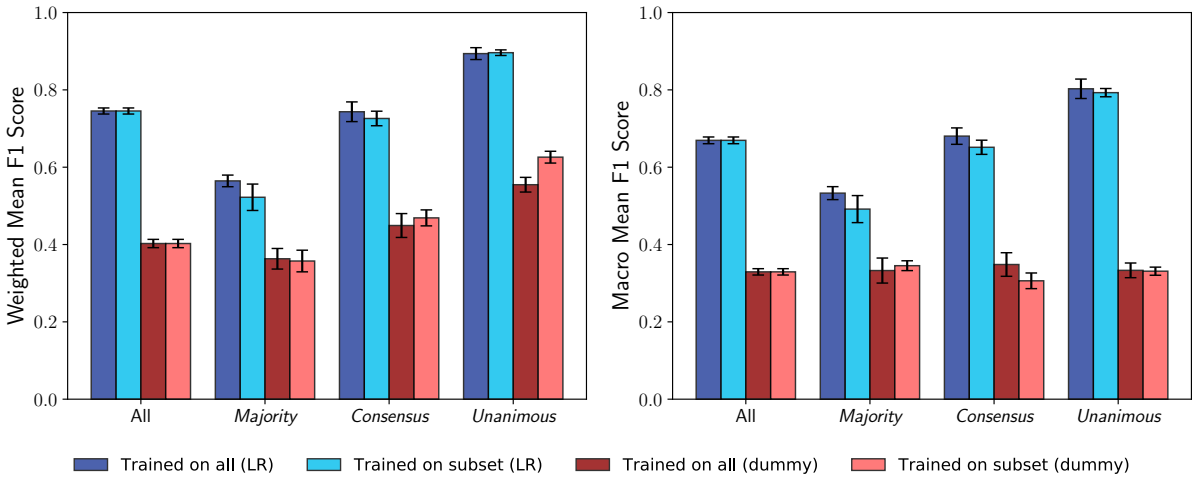


Figure 2: *Experiment I.* Weighted- and macro-F1-scores, obtained by testing logistic regression (LR) and a stratified random guesser (dummy) on the different agreement-level subsets, as described in Section 4.2. Black bars indicate plus/minus one standard deviation from the mean, as computed from the accuracies obtained across each of the 5 cross-validated folds.

ture ablation, whether or not to balance the train set classes, and different models (SVM with linear or RBF kernel, Random Forests, Naive Bayes, and K-Nearest-Neighbors), proved that this model variant was the best. We compare to a stratified random guesser, which predicts according to the distribution of classes in the training set (e.g., if 50% of the training set has samples labelled as OBJECTIVE, it will guess OBJECTIVE 50% of the time). To account for possible variance in the results, we use 5-fold cross validation over the full dataset, where the accuracy reported is the average over the specific scores obtained on each fold.

4.3 Evaluation

We evaluate with weighted- and macro-F1-scores to assess classifier performance. F1-score is a common way to measure classifier performance in sentiment analysis as it computes the harmonic mean between precision and recall. In multi-class classification, we obtain a one-versus-all F-score, F_c, for each class c in our set of possible classes, $\mathcal{C}$. Weighted F-score weights each F-score by its support in the test set; if there are n_c samples in the test set belonging to class c, then the weighted F-score is expressed by $\mathbf{F}_{weighted}$ in Equation 1.

$$\mathbf{F}_{weighted} = \frac{1}{(\sum n_c)} \sum_{c \in \mathcal{C}} n_c F_c \qquad (1)$$

Naturally, the weighted F-score is influenced by the frequency of samples in a class; so, in our case, it is biased toward the OBJECTIVE class due to its large frequency compared to the other classes (Table 5; Figure 1). Thus, we also report the macro F-score, which averages the F-scores for each class without considering their support, expressed by $\mathbf{F}_{macro}$ in Equation 2. This score evaluates model performance isolated from the class distribution, allowing us to determine if a change in accuracy is the result of simply a change in distribution of classes or a change in model generalization ability.

$$\mathbf{F}_{macro} = \frac{1}{|C|} \sum_{c \in \mathcal{C}} F_c \qquad (2)$$

5 Results

In Figure 2, we present the results for Experiment I (Section 4.2). We note that the presented accuracy is higher when evaluated with weighted F-score versus macro F-score. Since both weighted- and macro-F1-score increase as we move along to higher agreement subsets, we conclude that the accuracy improvement is not solely due to a change in distribution of classes. Rather, there must be a qualitative difference between high- vs. low-agreement tweets, otherwise the accuracy would have been the same across agreement levels.

In Figure 3, we present the normalized confusion matrix obtained from Experiment II. We observe that the model poorly classifies COMPLICATED tweets. Although the model uses balanced class weights for training, it predicts OBJECTIVE the majority of the time, where each other class is most frequently mistaken as OBJEC-

Test subset	Trained on all			Trained on subset		
	Precision	*Recall*	*F1-Score*	*Precision*	*Recall*	*F1-Score*
All	0.681	0.660	0.669	–	–	–
Majority	0.552	0.524	**0.533**	0.502	0.488	0.491
Consensus	0.689	0.674	**0.680**	0.680	0.633	0.652
Unanimous	0.789	0.821	**0.803**	0.843	0.761	0.793

Table 6: *Experiment I. Macro*-F1 score results for precision, recall, and F1-score, as shown visually in Figure 2. These are the mean scores across the 5 cross-validated folds. Bold numbers indicate the improvement of training on all data versus just the subset being tested upon.

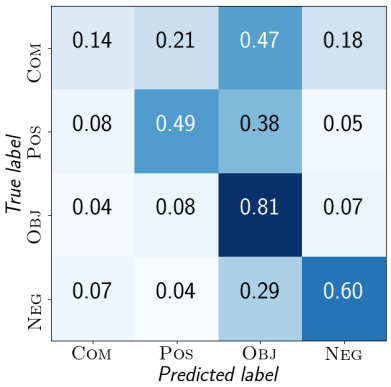


Figure 3: *Experiment II.* Normalized confusion matrix from using logistic regression.

Label	*Precision*	*Recall*	*F1*
COMPLICATED	0.199	0.138	0.163
POSITIVE	0.599	0.605	0.602
OBJECTIVE	0.766	0.806	0.786
NEGATIVE	0.510	0.487	0.498
Total – *weighted*	0.650	0.667	0.658
Total – *macro*	0.518	0.509	0.512

Table 7: *Experiment II.* Results across evaluation metrics, as shown visually in Figure 3. Results are computed by agglomerating all predictions made across each of the 5 cross-validated folds.

TIVE. The final weighted-F1-score and macro-F1-scores, were, respectively: 65.8% and 51.2% with logistic regression, and 41.1% and 24.7% with the stratified random guesser. This large difference between weighted and macro is largely due to the poor classifier performance on the COMPLICATED class.

6 Discussion

The interpretation of expressed sentiment is an inherently subjective exercise, the gold-standard of which is the sentiment perceived by other humans. Thus, it is crucial to better understand sentiment annotation itself to inform future classifier design.

Annotator disagreement is not human error. Our results show that annotator disagreements cannot simply be attributed to human error. There is a clear decrease in classifier performance when testing on subsets of tweets with lower annotator agreement (Figure 2), suggesting that tweets across these subsets are qualitatively different from each other. From a probabilistic perspective, this means that samples that obtain high annotator agreement are generated by a different real-world function than those that obtain low annotator agreement. This perspective is further justified by the fact that classifier performance is roughly the same when training on the full dataset versus when training just on the specific agreement level subsets. Future work should explore how to handle this data, and we recommend reporting results on the different subsets by agreement-level.

On defaulting to the majority label. When each tweet is assigned a gold-standard label according to the majority annotation, we demonstrated that there are qualitative differences between tweets with *Majority*, *Consensus*, and *Unanimous* agreement. As exemplified by the sample tweets in Table 1, the differences between the two tweets with a majority OBJECTIVE annotation is reflected in the inter-annotator disagreement. We have shown that the subtleties in sentiment expression are masked by simply taking the majority label, and future work would involve factoring in these varying levels of agreement on labels during the model design process.

Standards for sentiment analysis datasets. To advance the field of short-text sentiment analysis, it is necessary to change common practices

in dataset design and development. First, future datasets should be released with the raw annotator label assignments without discarding any annotated data. This would allow other researchers to experiment with different means of handling annotation-disagreement during the model design process. Secondly, we argue that sufficient resolution of short-text sentiment annotations requires at least 5x coverage. Our dataset, MTSA, of 7,026 tweets was constructed with 5x annotation coverage, a resolution at which we can just begin to distinguish these subsets of tweets. Higher coverage may be needed still to identify and understand these annotator disagreements. In contrast, the differences between these two subsets would be masked using the 3x coverage commonly found in other datasets.

Identifying ambiguous data. Results from Experiment II, and analysis of our COMPLICATED tweets, reveal that detecting high-disagreement tweets is a difficult task for both classifiers and humans. The poor performance of human annotators on identifying ambiguous tweets in our study, and the fact that high disagreement affected up to one third of the samples across TSA datasets, suggests that "complicatedness" is a real phenomenon. The optimal way to handle and identify this data requires further research. It is, however, an essential problem to solve, as real-world implementations of automated sentiment analysis systems will inevitably be confronted with such data. Such a system may be able to leverage the raw annotations during training, which is why we release the MTSA dataset with the raw annotation results included, and suggest all others do this as well.

7 Conclusion

In this paper, we highlight the need to better engage with how humans actually annotate data in short-text sentiment analysis dataset construction by constructing the new McGill Twitter Sentiment Analysis (MTSA) dataset. Future work involves leveraging raw human annotations to improve sentiment analysis classifiers, and finding ways to better detect and understand the "complicated" property in these samples that cause high annotator disagreement. Additionally, we encourage researchers to use MTSA in the development of other methods for short text sentiment analysis, including unsupervised, lexicon-based, and rule-based methods.

Acknowledgements

This work was the product of a class project pursued collectively by students in the COMP 767 graduate seminar in Social Media Analytics at McGill University, taught by Derek Ruths. This work was funded by the Discovery Grant Accelerator Supplement 2017-05165.

References

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*. volume 10, pages 2200–2204.

Johan Bollen, Huina Mao, and Alberto Pepe. 2011. Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena. *ICWSM* 11:450–453.

Cícero Nogueira Dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *COLING*. pages 69–78.

Anastasia Giachanou and Fabio Crestani. 2016. Like it or not: A survey of twitter sentiment analysis methods. *ACM Computing Surveys (CSUR)* 49(2):28.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford* 1(2009):12.

Svetlana Kiritchenko, Xiaodan Zhu, and Saif M Mohammad. 2014. Sentiment analysis of short informal texts. *Journal of Artificial Intelligence Research* 50:723–762.

Eugenio Martínez-Cámara, M Teresa Martín-Valdivia, L Alfonso Urena-López, and A Rturo Montejo-Ráez. 2014. Sentiment analysis in twitter. *Natural Language Engineering* 20(1):1–28.

Walaa Medhat, Ahmed Hassan, and Hoda Korashy. 2014. Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal* 5(4):1093–1113.

Saif Mohammad. 2016. A practical guide to sentiment annotation: Challenges and solutions. In *Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. pages 174–179.

Preslav Nakov, Sara Rosenthal, Svetlana Kiritchenko, Saif M Mohammad, Zornitsa Kozareva, Alan Ritter, Veselin Stoyanov, and Xiaodan Zhu. 2016. Developing a successful semeval task in sentiment analysis of twitter and other social media texts. *Language Resources and Evaluation* 50(1):35–65.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *LREc*. volume 10.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543.

Soujanya Poria, Erik Cambria, Newton Howard, Guang-Bin Huang, and Amir Hussain. 2016. Fusing audio, visual and textual clues for sentiment analysis from multimodal content. *Neurocomputing* 174:50–59.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. Semeval-2017 task 4: Sentiment analysis in twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. pages 502–518.

Hassan Saif, Miriam Fernandez, Yulan He, and Harith Alani. 2013. Evaluation datasets for Twitter sentiment analysis: a survey and a new dataset, the STS-Gold. In *CEUR Workshop Proceedings*. volume 1096, pages 9–21.

David A Shamma, Lyndon Kennedy, and Elizabeth F Churchill. 2009. Tweet the debates: understanding community annotation of uncollected sources. In *Proceedings of the first SIGMM workshop on Social media*. ACM, pages 3–10.

Mohammad Soleymani, David Garcia, Brendan Jou, Björn Schuller, Shih-Fu Chang, and Maja Pantic. 2017. A survey of multimodal sentiment analysis. *Image and Vision Computing* 65:3–14.

Michael Speriosu, Nikita Sudan, Sid Upadhyay, and Jason Baldridge. 2011. Twitter polarity classification with label propagation over lexical links and the follower graph. In *Proceedings of the First workshop on Unsupervised Learning in NLP*. Association for Computational Linguistics, pages 53–63.

Mike Thelwall, Kevan Buckley, and Georgios Paltoglou. 2012. Sentiment strength detection for the social web. *Journal of the Association for Information Science and Technology* 63(1):163–173.

Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. 2010. Sentiment strength detection in short informal text. *Journal of the Association for Information Science and Technology* 61(12):2544–2558.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*. Association for Computational Linguistics, pages 347–354.

Multi-task Learning of Pairwise Sequence Classification Tasks Over Disparate Label Spaces

Isabelle Augenstein[1]*, **Sebastian Ruder**[23]*, **Anders Søgaard**[1]
[1]Department of Computer Science, University of Copenhagen, Denmark
[2]Insight Research Centre, National University of Ireland, Galway
[3]Aylien Ltd., Dublin, Ireland
{augenstein|soegaard}@di.ku.dk, sebastian@ruder.io

Abstract

We combine multi-task learning and semi-supervised learning by inducing a joint embedding space between disparate label spaces and learning transfer functions between label embeddings, enabling us to jointly leverage unlabelled data and auxiliary, annotated datasets. We evaluate our approach on a variety of sequence classification tasks with disparate label spaces. We outperform strong single and multi-task baselines and achieve a new state-of-the-art for topic-based sentiment analysis.

1 Introduction

Multi-task learning (MTL) and semi-supervised learning are both successful paradigms for learning in scenarios with limited labelled data and have in recent years been applied to almost all areas of NLP. Applications of MTL in NLP, for example, include partial parsing (Søgaard and Goldberg, 2016), text normalisation (Bollman et al., 2017), neural machine translation (Luong et al., 2016), and keyphrase boundary classification (Augenstein and Søgaard, 2017).

Contemporary work in MTL for NLP typically focuses on learning representations that are useful across tasks, often through hard parameter sharing of hidden layers of neural networks (Collobert et al., 2011; Søgaard and Goldberg, 2016). If tasks share optimal hypothesis classes at the level of these representations, MTL leads to improvements (Baxter, 2000). However, while sharing hidden layers of neural networks is an effective regulariser (Søgaard and Goldberg, 2016), we potentially *loose synergies between the classification functions* trained to associate these representations with class labels. This paper sets out to build an architecture in which such synergies are exploited, with an application to pairwise sequence classification tasks. Doing so, we achieve a new state of the art on topic-based sentiment analysis.

For many NLP tasks, disparate label sets are weakly correlated, e.g. part-of-speech tags correlate with dependencies (Hashimoto et al., 2017), sentiment correlates with emotion (Felbo et al., 2017; Eisner et al., 2016), etc. We thus propose to induce a joint label embedding space (visualised in Figure 2) using a Label Embedding Layer that allows us to model these relationships, which we show helps with learning.

In addition, for tasks where labels are closely related, we should be able to not only model their relationship, but also to directly estimate the corresponding label of the target task based on auxiliary predictions. To this end, we propose to train a Label Transfer Network (LTN) jointly with the model to produce pseudo-labels across tasks.

The LTN can be used to label unlabelled and auxiliary task data by utilising the 'dark knowledge' (Hinton et al., 2015) contained in auxiliary model predictions. This pseudo-labelled data is then incorporated into the model via semi-supervised learning, leading to a natural combination of multi-task learning and semi-supervised learning. We additionally augment the LTN with data-specific diversity features (Ruder and Plank, 2017) that aid in learning.

Contributions Our contributions are: a) We model the relationships between labels by inducing a joint label space for multi-task learning. b) We propose a Label Transfer Network that learns to transfer labels between tasks and propose to use semi-supervised learning to leverage them for training. c) We evaluate MTL approaches on a variety of classification tasks and shed new light on settings where multi-task learning works. d) We perform an extensive ablation study of our model.

*The first two authors contributed equally.

Proceedings of NAACL-HLT 2018, pages 1896–1906
New Orleans, Louisiana, June 1 - 6, 2018.

e) We report state-of-the-art performance on topic-based sentiment analysis.

2 Related work

Learning task similarities Existing approaches for learning similarities between tasks enforce a clustering of tasks (Evgeniou et al., 2005; Jacob et al., 2009), induce a shared prior (Yu et al., 2005; Xue et al., 2007; Daumé III, 2009), or learn a grouping (Kang et al., 2011; Kumar and Daumé III, 2012). These approaches focus on homogeneous tasks and employ linear or Bayesian models. They can thus not be directly applied to our setting with tasks using disparate label sets.

Multi-task learning with neural networks Recent work in multi-task learning goes beyond hard parameter sharing (Caruana, 1993) and considers different sharing structures, e.g. only sharing at lower layers (Søgaard and Goldberg, 2016) and induces private and shared subspaces (Liu et al., 2017; Ruder et al., 2017). These approaches, however, are not able to take into account relationships between labels that may aid in learning. Another related direction is to train on disparate annotations of the same task (Chen et al., 2016; Peng et al., 2017). In contrast, the different nature of our tasks requires a modelling of their label spaces.

Semi-supervised learning There exists a wide range of semi-supervised learning algorithms, e.g., self-training, co-training, tri-training, EM, and combinations thereof, several of which have also been used in NLP. Our approach is probably most closely related to an algorithm called *co-forest* (Li and Zhou, 2007). In co-forest, like here, each learner is improved with unlabeled instances labeled by the ensemble consisting of all the other learners. Note also that several researchers have proposed using auxiliary tasks that are unsupervised (Plank et al., 2016; Rei, 2017), which also leads to a form of semi-supervised models.

Label transformations The idea of manually mapping between label sets or learning such a mapping to facilitate transfer is not new. Zhang et al. (2012) use distributional information to map from a language-specific tagset to a tagset used for other languages, in order to facilitate cross-lingual transfer. More related to this work, Kim et al. (2015) use canonical correlation analysis to transfer between tasks with disparate label spaces. There has also been work on label transformations in the context of multi-label classification problems (Yeh et al., 2017).

3 Multi-task learning with disparate label spaces

3.1 Problem definition

In our multi-task learning scenario, we have access to labelled datasets for T tasks $\mathcal{T}_1, \ldots, \mathcal{T}_T$ at training time with a target task $\mathcal{T}_T$ that we particularly care about. The training dataset for task $\mathcal{T}_i$ consists of N_k examples $X_{\mathcal{T}_i} = \{x_1^{\mathcal{T}_i}, \ldots, x_{N_k}^{\mathcal{T}_i}\}$ and their labels $Y_{\mathcal{T}_i} = \{\mathbf{y}_1^{\mathcal{T}_i}, \ldots, \mathbf{y}_{N_k}^{\mathcal{T}_i}\}$. Our base model is a deep neural network that performs classic hard parameter sharing (Caruana, 1993): It shares its parameters across tasks and has task-specific softmax output layers, which output a probability distribution $\mathbf{p}^{\mathcal{T}_i}$ for task $\mathcal{T}_i$ according to the following equation:

$$\mathbf{p}^{\mathcal{T}_i} = \mathrm{softmax}(\mathbf{W}^{\mathcal{T}_i}\mathbf{h} + \mathbf{b}^{\mathcal{T}_i}) \quad (1)$$

where $\mathrm{softmax}(\mathbf{x}) = e^{\mathbf{x}} / \sum_{i=1}^{\|\mathbf{x}\|} e^{\mathbf{x}_i}$, $\mathbf{W}^{\mathcal{T}_i} \in \mathbb{R}^{L_i \times h}$, $\mathbf{b}^{\mathcal{T}_i} \in \mathbb{R}^{L_i}$ is the weight matrix and bias term of the output layer of task $\mathcal{T}_i$ respectively, $\mathbf{h} \in \mathbb{R}^h$ is the jointly learned hidden representation, L_i is the number of labels for task $\mathcal{T}_i$, and h is the dimensionality of $\mathbf{h}$.

The MTL model is then trained to minimise the sum of the individual task losses:

$$\mathcal{L} = \lambda_1 \mathcal{L}_1 + \ldots + \lambda_T \mathcal{L}_T \quad (2)$$

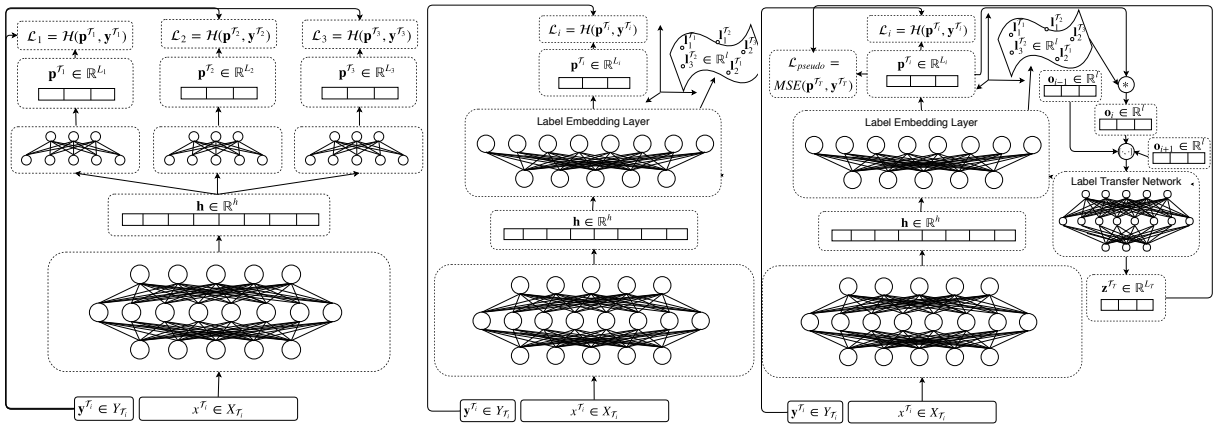
where $\mathcal{L}_i$ is the negative log-likelihood objective $\mathcal{L}_i = \mathcal{H}(\mathbf{p}^{\mathcal{T}_i}, \mathbf{y}^{\mathcal{T}_i}) = -\frac{1}{N} \sum_n \sum_j \log \mathbf{p}_j^{\mathcal{T}_i} \mathbf{y}_j^{\mathcal{T}_i}$ and λ_i is a parameter that determines the weight of task $\mathcal{T}_i$. In practice, we apply the same weight to all tasks. We show the full set-up in Figure 1a.

3.2 Label Embedding Layer

In order to learn the relationships between labels, we propose a Label Embedding Layer (LEL) that embeds the labels of all tasks in a joint space. Instead of training separate softmax output layers as above, we introduce a label compatibility function $c(\cdot, \cdot)$ that measures how similar a label with embedding $\mathbf{l}$ is to the hidden representation $\mathbf{h}$:

$$c(\mathbf{l}, \mathbf{h}) = \mathbf{l} \cdot \mathbf{h} \quad (3)$$

where $\cdot$ is the dot product. This is similar to the Universal Schema Latent Feature Model introduced by Riedel et al. (2013). In contrast to



(a) Multi-task learning (b) MTL with LEL (c) Semi-supervised MTL with LTN

Figure 1: a) Multi-task learning (MTL) with hard parameter sharing and 3 tasks $\mathcal{T}_{1-3}$ and L_{1-3} labels per task. A shared representation $\mathbf{h}$ is used as input to task-specific softmax layers, which optimise cross-entropy losses $\mathcal{L}_{1-3}$. b) MTL with the Label Embedding Layer (LEL) embeds task labels $\mathbf{l}_{1-L_i}^{\mathcal{T}_{1-3}}$ in a joint embedding space and uses these for prediction with a label compatibility function. c) Semi-supervised MTL with the Label Transfer Network (LTN) in addition optimises an unsupervised loss $\mathcal{L}_{pseudo}$ over pseudo-labels $\mathbf{z}^{\mathcal{T}_T}$ on auxiliary/unlabelled data. The pseudo-labels $\mathbf{z}^{\mathcal{T}_T}$ are produced by the LTN for the main task $\mathcal{T}_T$ using the concatenation of auxiliary task label output embeddings $[\mathbf{o}_{i-1}, \mathbf{o}_i, \mathbf{o}_{i+1}]$ as input.

other models that use the dot product in the objective function, we do not have to rely on negative sampling and a hinge loss (Collobert and Weston, 2008) as negative instances (labels) are known. For efficiency purposes, we use matrix multiplication instead of a single dot product and softmax instead of sigmoid activations:

$$\mathbf{p} = \text{softmax}(\mathbf{L}\mathbf{h}) \tag{4}$$

where $\mathbf{L} \in \mathbb{R}^{(\sum_i L_i) \times l}$ is the label embedding matrix for all tasks and l is the dimensionality of the label embeddings. In practice, we set l to the hidden dimensionality h. We use padding if $l < h$. We apply a task-specific mask to $\mathbf{L}$ in order to obtain a task-specific probability distribution $\mathbf{p}^{\mathcal{T}_i}$. The LEL is shared across all tasks, which allows us to learn the relationships between the labels in the joint embedding space. We show MTL with the LEL in Figure 1b.

3.3 Label Transfer Network

The LEL allows us to learn the relationships between labels. In order to make use of these relationships, we would like to leverage the predictions of our auxiliary tasks to estimate a label for the target task. To this end, we introduce the Label Transfer Network (LTN). This network takes the auxiliary task outputs as input. In particular, we define the output label embedding $\mathbf{o}_i$ of task $\mathcal{T}_i$ as the sum of the task's label embeddings $\mathbf{l}_j$ weighted with their probability $\mathbf{p}_j^{\mathcal{T}_i}$:

$$\mathbf{o}_i = \sum_{j=1}^{L_i} \mathbf{p}_j^{\mathcal{T}_i} \mathbf{l}_j \tag{5}$$

The label embeddings $\mathbf{l}$ encode general relationship between labels, while the model's probability distribution $\mathbf{p}^{\mathcal{T}_i}$ over its predictions encodes fine-grained information useful for learning (Hinton et al., 2015). The LTN is trained on labelled target task data. For each example, the corresponding label output embeddings of the auxiliary tasks are fed into a multi-layer perceptron (MLP), which is trained with a negative log-likelihood objective $\mathcal{L}_{\text{LTN}}$ to produce a pseudo-label $\mathbf{z}^{\mathcal{T}_T}$ for the target task $\mathcal{T}_T$:

$$\text{LTN}_T = \text{MLP}([\mathbf{o}_1, \ldots, \mathbf{o}_{T-1}]) \tag{6}$$

where $[\cdot, \cdot]$ designates concatenation. The mapping of the tasks in the LTN yields another signal that can be useful for optimisation and act as a regulariser. The LTN can also be seen as a mixture-of-experts layer (Jacobs et al., 1991) where the experts are the auxiliary task models. As the label embeddings are learned jointly with the main model, the LTN is more sensitive to the relationships between labels than a separately learned mixture-of-experts model that only relies on the experts' output distributions. As such, the LTN

can be directly used to produce predictions on unseen data.

3.4 Semi-supervised MTL

The downside of the LTN is that it requires additional parameters and relies on the predictions of the auxiliary models, which impacts the runtime during testing. Instead, of using the LTN for prediction directly, we can use it to provide pseudo-labels for unlabelled or auxiliary task data by utilising auxiliary predictions for semi-supervised learning.

We train the target task model on the pseudo-labelled data to minimise the squared error between the model predictions $\mathbf{p}^{\mathcal{T}_i}$ and the pseudo labels $\mathbf{z}^{\mathcal{T}_i}$ produced by the LTN:

$$\mathcal{L}_{pseudo} = MSE(\mathbf{p}^{\mathcal{T}_T}, \mathbf{z}^{\mathcal{T}_T}) = ||\mathbf{p}^{\mathcal{T}_T} - \mathbf{z}^{\mathcal{T}_T}||^2 \tag{7}$$

We add this loss term to the MTL loss in Equation 2. As the LTN is learned together with the MTL model, pseudo-labels produced early during training will likely not be helpful as they are based on unreliable auxiliary predictions. For this reason, we first train the base MTL model until convergence and then augment it with the LTN. We show the full semi-supervised learning procedure in Figure 1c.

3.5 Data-specific features

When there is a domain shift between the datasets of different tasks as is common for instance when learning NER models with different label sets, the output label embeddings might not contain sufficient information to bridge the domain gap.

To mitigate this discrepancy, we augment the LTN's input with features that have been found useful for transfer learning (Ruder and Plank, 2017). In particular, we use the number of word types, type-token ratio, entropy, Simpson's index, and Rényi entropy as diversity features. We calculate each feature for each example.[1] The features are then concatenated with the input of the LTN.

3.6 Other multi-task improvements

Hard parameter sharing can be overly restrictive and provide a regularisation that is too heavy when jointly learning many tasks. For this reason, we propose several additional improvements that seek to alleviate this burden: We use skip-connections, which have been shown to be useful for multi-task learning in recent work (Ruder et al., 2017). Furthermore, we add a task-specific layer before the output layer, which is useful for learning task-specific transformations of the shared representations (Søgaard and Goldberg, 2016; Ruder et al., 2017).

[1]For more information regarding the feature calculation, refer to Ruder and Plank (2017).

Task	Domain	N	L	Metric
`Topic-2`	Twitter	4,346	2	ρ^{PN}
`Topic-5`	Twitter	6,000	5	MAE^{M}
`Target`	Twitter	6,248	3	F_1^{M}
`Stance`	Twitter	2,914	3	F_1^{FA}
`ABSA-L`	Reviews	2,909	3	Acc
`ABSA-R`	Reviews	2,507	3	Acc
`FNC-1`	News	39,741	4	Acc
`MultiNLI`	Diverse	392,702	3	Acc

Table 1: Training set statistics and evaluation metrics of every task. N: # of examples. L: # of labels.

4 Experiments

For our experiments, we evaluate on a wide range of text classification tasks. In particular, we choose pairwise classification tasks—i.e. those that condition the reading of one sequence on another sequence—as we are interested in understanding if knowledge can be transferred even for these more complex interactions. To the best of our knowledge, this is the first work on transfer learning between such pairwise sequence classification tasks. We implement all our models in Tensorflow (Abadi et al., 2016) and release the code at `https://github.com/coastalcph/mtl-disparate`.

4.1 Tasks and datasets

We use the following tasks and datasets for our experiments, show task statistics in Table 1, and summarise examples in Table 2:

Topic-based sentiment analysis Topic-based sentiment analysis aims to estimate the sentiment of a tweet known to be about a given topic. We use the data from SemEval-2016 Task 4 Subtask B and C (Nakov et al., 2016) for predicting on a two-point scale of positive and negative (`Topic-2`) and five-point scale ranging from highly negative to highly positive (`Topic-5`) respectively. An example from this dataset would be to classify the

Topic-based sentiment analysis: *Tweet*: No power at home, sat in the dark listening to AC/DC in the hope it'll make the electricity come back again *Topic*: AC/DC *Label*: positive
Target-dependent sentiment analysis: *Text*: how do you like settlers of catan for the wii? *Target*: wii *Label*: neutral
Aspect-based sentiment analysis: *Text*: For the price, you cannot eat this well in Manhattan *Aspects*: restaurant prices, food quality *Label*: positive
Stance detection: *Tweet*: Be prepared - if we continue the policies of the liberal left, we will be #Greece *Target*: Donald Trump *Label*: favor
Fake news detection: *Document*: Dino Ferrari hooked the whopper wels catfish, (...), which could be the biggest in the world. *Headline*: Fisherman lands 19 STONE catfish which could be the biggest in the world to be hooked *Label*: agree
Natural language inference: *Premise*: Fun for only children *Hypothesis*: Fun for adults and children *Label*: contradiction

Table 2: Example instances from the datasets described in Section 4.1.

tweet "No power at home, sat in the dark listening to AC/DC in the hope it'll make the electricity come back again" known to be about the topic "AC/DC", which is labelled as a positive sentiment. The evaluation metrics for `Topic-2` and `Topic-5` are macro-averaged recall (ρ^{PN}) and macro-averaged mean absolute error (MAE^M) respectively, which are both averaged across topics.

Target-dependent sentiment analysis Target-dependent sentiment analysis (`Target`) seeks to classify the sentiment of a text's author towards an entity that occurs in the text as positive, negative, or neutral. We use the data from Dong et al. (2014). An example instance is the expression "how do you like settlers of catan for the wii?" which is labelled as neutral towards the target "wii'.' The evaluation metric is macro-averaged F_1 (F_1^M).

Aspect-based sentiment analysis Aspect-based sentiment analysis is the task of identifying whether an aspect, i.e. a particular property of an item is associated with a positive, negative, or neutral sentiment (Ruder et al., 2016). We use the data of SemEval-2016 Task 5 Subtask 1 Slot 3 (Pontiki et al., 2016) for the laptops (`ABSA-L`) and restaurants (`ABSA-R`) domains. An example is the sentence "For the price, you cannot eat this well in Manhattan", labelled as positive towards both the aspects "restaurant prices" and "food quality". The evaluation metric for both domains is accuracy (Acc).

Stance detection Stance detection (`Stance`) requires a model, given a text and a target entity, which might not appear in the text, to predict whether the author of the text is in favour or against the target or whether neither inference is likely (Augenstein et al., 2016). We use the data of SemEval-2016 Task 6 Subtask B (Mohammad et al., 2016). An example from this dataset would be to predict the stance of the tweet "Be prepared - if we continue the policies of the liberal left, we will be #Greece" towards the topic "Donald Trump", labelled as "favor". The evaluation metric is the macro-averaged F_1 score of the "favour" and "against" classes (F_1^{FA}).

Fake news detection The goal of fake news detection in the context of the Fake News Challenge[2] is to estimate whether the body of a news article agrees, disagrees, discusses, or is unrelated towards a headline. We use the data from the first stage of the Fake News Challenge (`FNC-1`). An example for this dataset is the document "Dino Ferrari hooked the whopper wels catfish, (...), which could be the biggest in the world." with the headline "Fisherman lands 19 STONE catfish which could be the biggest in the world to be hooked" labelled as "agree". The evaluation metric is accuracy (Acc)[3].

Natural language inference Natural language inference is the task of predicting whether one sentences entails, contradicts, or is neutral towards another one. We use the Multi-Genre NLI corpus (`MultiNLI`) from the RepEval 2017 shared task (Nangia et al., 2017). An example for an instance would be the sentence pair "Fun for only children", "Fun for adults and children", which are in a "contradiction" relationship. The evaluation metric is accuracy (Acc).

[2] `http://www.fakenewschallenge.org/`

[3] We use the same metric as Riedel et al. (2017).

	Stance	FNC	MultiNLI	Topic-2	Topic-5*	ABSA-L	ABSA-R	Target
Augenstein et al. (2016)	**49.01**	-	-	-	-	-	-	-
Riedel et al. (2017)	-	**88.46**	-	-	-	-	-	-
Chen et al. (2017)	-	-	**74.90**	-	-	-	-	-
Palogiannidi et al. (2016)	-	-	-	79.90	-	-	-	-
Balikas and Amini (2016)	-	-	-	-	**0.719**	-	-	-
Brun et al. (2016)	-	-	-	-	-	-	**88.13**	-
Kumar et al. (2016)	-	-	-	-	-	**82.77**	86.73	-
Vo and Zhang (2015)	-	-	-	-	-	-	-	**69.90**
STL	41.1	72.72	49.25	63.92	0.919	76.74	67.47	64.01
MTL + LEL	46.26	72.71	49.94	**80.52**	0.814	74.94	79.90	66.42
MTL + LEL + LTN, main model	43.16	72.73	48.75	73.90	0.810	75.06	83.71	66.10
MTL + LEL + LTN + semi, main model	43.56	72.72	48.00	72.35	0.821	75.42	83.26	63.00

Table 3: Comparison of our best performing models on the test set against a single task baseline and the state of the art, with task specific metrics. *: lower is better. Bold: best. Underlined: second-best.

4.2 Base model

Our base model is the Bidirectional Encoding model (Augenstein et al., 2016), a state-of-the-art model for stance detection that conditions a bidirectional LSTM (BiLSTM) encoding of a text on the BiLSTM encoding of the target. Unlike Augenstein et al. (2016), we do not pre-train word embeddings on a larger set of unlabelled in-domain text for each task as we are mainly interested in exploring the benefit of multi-task learning for generalisation.

4.3 Training settings

We use BiLSTMs with one hidden layer of 100 dimensions, 100-dimensional randomly initialised word embeddings, a label embedding size of 100. We train our models with RMSProp, a learning rate of 0.001, a batch size of 128, and early stopping on the validation set of the main task with a patience of 3.

5 Results

Our main results are shown in Table 3, with a comparison against the state of the art. We present the results of our multi-task learning network with label embeddings (MTL + LEL), multi-task learning with label transfer (MTL + LEL + LTN), and the semi-supervised extension of this model. On 7/8 tasks, at least one of our architectures is better than single-task learning; and in 4/8, all our architectures are much better than single-task learning.

The state-of-the-art systems we compare against are often highly specialised, task-dependent architectures. Our architectures, in contrast, have not been optimised to compare favourably against the state of the art, as our main objective is to develop a novel approach to multi-task learning leveraging synergies between label sets and knowledge of marginal distributions from unlabeled data. For example, we do not use pre-trained word embeddings (Augenstein et al., 2016; Palogiannidi et al., 2016; Vo and Zhang, 2015), class weighting to deal with label imbalance (Balikas and Amini, 2016), or domain-specific sentiment lexicons (Brun et al., 2016; Kumar et al., 2016). Nevertheless, our approach outperforms the state-of-the-art on two-way topic-based sentiment analysis (Topic-2).

The poor performance compared to the state-of-the-art on FNC and MultiNLI is expected; as we alternate among the tasks during training, our model only sees a comparatively small number of examples of both corpora, which are one and two orders of magnitude larger than the other datasets. For this reason, we do not achieve good performance on these tasks as main tasks, but they are still useful as auxiliary tasks as seen in Table 4.

6 Analysis

6.1 Label Embeddings

Our results above show that, indeed, modelling the similarity between tasks using label embeddings sometimes leads to much better performance. Figure 2 shows why. In Figure 2, we visualise the label embeddings of an MTL+LEL model trained on all tasks, using PCA. As we can see, similar labels are clustered together across tasks, e.g. there are two positive clusters (middle-right and top-right), two negative clusters (middle-left and bottom-left), and two neutral clusters (middle-top

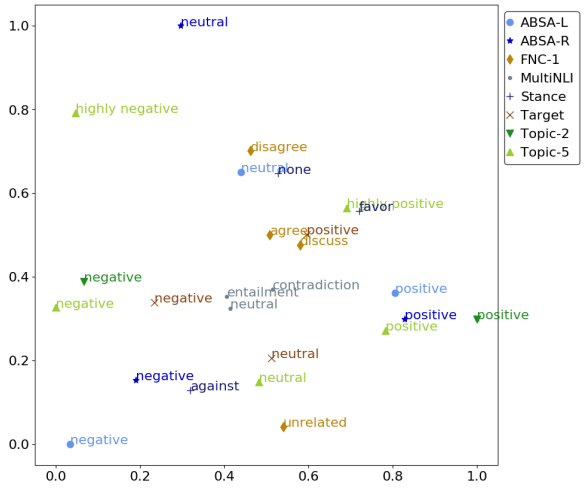


Figure 2: Label embeddings of all tasks. Positive, negative, and neutral labels are clustered together.

and middle-bottom).

Our visualisation also provides us with a picture of what auxilary tasks are beneficial, and to what extent we can expect synergies from multi-task learning. For instance, the notion of positive sentiment appears to be very similar across the topic-based and aspect-based tasks, while the conceptions of negative and neutral sentiment differ. In addition, we can see that the model has failed to learn a relationship between `MultiNLI` labels and those of other tasks, possibly accounting for its poor performance on the inference task. We did not evaluate the correlation between label embeddings and task performance, but Bjerva (2017) recently suggested that mutual information of target and auxiliary task label sets is a good predictor of gains from multi-task learning.

6.2 Auxilary Tasks

For each task, we show the auxiliary tasks that achieved the best performance on the development data in Table 4. In contrast to most existing work, we did not restrict ourselves to performing multi-task learning with only one auxiliary task (Søgaard and Goldberg, 2016; Bingel and Søgaard, 2017). Indeed we find that most often a combination of auxiliary tasks achieves the best performance. In-domain tasks are less used than we assumed; only `Target` is consistently used by all Twitter main tasks. In addition, tasks with a higher number of labels, e.g. `Topic-5` are used more often. Such tasks provide a more fine-grained reward signal, which may help in learning representations that generalise better. Finally, tasks with large amounts of training data such as `FNC-1` and `MultiNLI` are also used more often. Even if not directly related, the larger amount of training data that can be indirectly leveraged via multi-task learning may help the model focus on relevant parts of the representation space (Caruana, 1993). These observations shed additional light on when multi-task learning may be useful that go beyond existing studies (Bingel and Søgaard, 2017).

Main task	Auxiliary tasks
`Topic-2`	`FNC-1`, `MultiNLI`, `Target`
`Topic-5`	`FNC-1`, `MultiNLI`, `ABSA-L`, `Target`
`Target`	`FNC-1`, `MultiNLI`, `Topic-5`
`Stance`	`FNC-1`, `MultiNLI`, `Target`
`ABSA-L`	`Topic-5`
`ABSA-R`	`Topic-5`, `ABSA-L`, `Target`
`FNC-1`	`Stance`, `MultiNLI`, `Topic-5`, `ABSA-R`, `Target`
`MultiNLI`	`Topic-5`

Table 4: Best-performing auxiliary tasks for different main tasks.

6.3 Ablation analysis

We now perform a detailed ablation analysis of our model, the results of which are shown in Table 5. We ablate whether to use the LEL (+ *LEL*), whether to use the LTN (+ *LTN*), whether to use the LEL output or the main model output for prediction (main model output is indicated by , *main model*), and whether to use the LTN as a regulariser or for semi-supervised learning (semi-supervised learning is indicated by + *semi*). We further test whether to use diversity features (– *diversity feats*) and whether to use main model predictions for the LTN (+ *main model feats*).

Overall, the addition of the Label Embedding Layer improves the performance over regular MTL in almost all cases.

6.4 Label transfer network

To understand the performance of the LTN, we analyse learning curves of the relabelling function vs. the main model. Examples for all tasks without semi-supervised learning are shown in Figure 3. One can observe that the relabelling model does not take long to converge as it has fewer parameters than the main model. Once the relabelling model is learned alongside the main

	Stance	FNC	MultiNLI	Topic-2	Topic-5*	ABSA-L	ABSA-R	Target
MTL	44.12	72.75	49.39	**80.74**	0.859	74.94	82.25	65.73
MTL + LEL	**46.26**	72.71	**49.94**	80.52	0.814	74.94	79.90	**66.42**
MTL + LTN	40.95	72.72	44.14	78.31	0.851	73.98	82.37	63.71
MTL + LTN, main model	41.60	72.72	47.62	79.98	0.814	75.54	81.70	65.61
MTL + LEL + LTN	44.48	**72.76**	43.72	74.07	0.821	**75.66**	81.92	65.00
MTL + LEL + LTN, main model	43.16	72.73	48.75	73.90	0.810	75.06	**83.71**	66.10
MTL + LEL + LTN + main preds feats	42.78	72.72	45.41	66.30	0.835	73.86	81.81	65.08
MTL + LEL + LTN + main preds feats, main model	42.65	72.73	48.81	67.53	**0.803**	75.18	82.59	63.95
MTL + LEL + LTN + main preds feats – diversity feats	42.78	72.72	43.13	66.3	0.835	73.5	81.7	63.95
MTL + LEL + LTN + main preds feats – diversity feats, main model	42.47	72.74	47.84	67.53	0.807	74.82	82.14	65.11
MTL + LEL + LTN + semi	42.65	72.75	44.28	77.81	0.841	74.10	81.36	64.45
MTL + LEL + LTN + semi, main model	43.56	72.72	48.00	72.35	0.821	75.42	83.26	63.00

Table 5: Ablation results with task-specific evaluation metrics on test set with early stopping on dev set. *LTN* means the output of the relabelling function is shown, which does not use the task predictions, only predictions from other tasks. *LTN + main preds feats* means main model predictions are used as features for the relabelling function. *LTN, main model* means that the main model predictions of the model that trains a relabelling function are used. Note that for MultiNLI, we down-sample the training data. *: lower is better. Bold: best. Underlined: second-best.

Task	Main	LTN	Main (Semi)	LTN (Semi)
Stance	2.12	2.62	1.94	1.28
FNC	4.28	2.49	6.92	4.84
MultiNLI	1.5	1.95	1.94	1.28
Topic-2	6.45	4.44	5.87	5.59
Topic-5*	9.22	9.71	11.3	5.90
ABSA-L	3.79	2.52	9.06	6.63
ABSA-R	10.6	6.70	9.06	6.63
Target	26.3	14.6	20.1	15.7

Table 6: Error analysis of LTN with and without semi-supervised learning for all tasks. Metric shown: percentage of correct predictions only made by either the relabelling function or the main model, respectively, relative to the the number of all correct predictions.

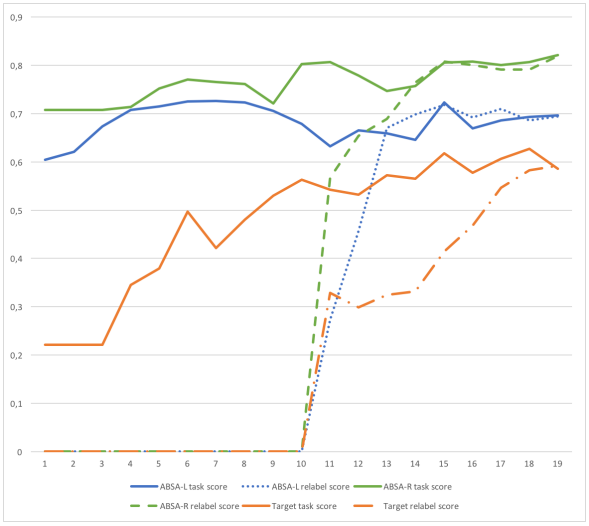


Figure 3: Learning curves with LTN for selected tasks, dev performances shown. The main model is pre-trained for 10 epochs, after which the relabelling function is trained.

model, the main model performance first stagnates, then starts to increase again. For some of the tasks, the main model ends up with a higher task score than the relabelling model. We hypothesise that the softmax predictions of other, even highly related tasks are less helpful for predicting main labels than the output layer of the main task model. At best, learning the relabelling model alongside the main model might act as a regulariser to the main model and thus improve the main model's performance over a baseline MTL model, as it is the case for TOPIC-5 (see Table 5).

To further analyse the performance of the LTN, we look into to what degree predictions of the main model and the relabelling model for individual instances are complementary to one another. Or, said differently, we measure the percentage of correct predictions made only by the relabelling model or made only by the main model, relative to the number of correct predictions overall. Results of this for each task are shown in Table 6 for the LTN with and without semi-supervised learning. One can observe that, even though the relabelling function overall contributes to the score to a lesser degree than the main model, a substantial number of correct predictions are made by the relabelling function that are missed by the main model. This is most prominently pronounced for ABSA-R, where the proportion is 14.6.

7 Conclusion

We have presented a multi-task learning architecture that (i) leverages potential synergies between classifier functions relating shared representations with disparate label spaces and (ii) enables learning from mixtures of labeled and unlabeled data. We have presented experiments with combinations of eight pairwise sequence classification tasks. Our results show that leveraging synergies between label spaces sometimes leads to big improvements, and we have presented a new state of the art for topic-based sentiment analysis. Our analysis further showed that (a) the learned label embeddings were indicative of gains from multi-task learning, (b) auxiliary tasks were often beneficial across domains, and (c) label embeddings almost always led to better performance. We also investigated the dynamics of the label transfer network we use for exploiting the synergies between disparate label spaces.

Acknowledgments

Sebastian Ruder is supported by the Irish Research Council Grant Number EBPPG/2014/30 and Science Foundation Ireland Grant Number SFI/12/RC/2289. Anders Søgaard is supported by the ERC Starting Grant Number 313695. Isabelle Augenstein is supported by Eurostars grant Number E10138. We further gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* .

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. 2016. Twitter Stance Detection with Bidirectional Conditional Encoding. In *Proceedings of EMNLP*.

Isabelle Augenstein and Anders Søgaard. 2017. Multi-task learning of keyphrase boundary detection. In *Proceedings of ACL*.

Georgios Balikas and Massih-Reza Amini. 2016. TwiSE at SemEval-2016 Task 4: Twitter Sentiment Classification. In *Proceedings of SemEval*.

Jonathan Baxter. 2000. A Model of Inductive Bias Learning. *JAIR* 12:149–198.

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. In *Proceedings of EACL*.

Johannes Bjerva. 2017. Will my auxiliary tagging task help? Estimating Auxiliary Tasks Effectivity in Multi-Task Learning. In *Proceedings of NODALIDA*.

Marcel Bollman, Joachim Bingel, and Anders Søgaard. 2017. Learning attention for historical text normalization by learning to pronounce. In *Proceedings of ACL*.

Caroline Brun, Julien Perez, and Claude Roux. 2016. XRCE at SemEval-2016 Task 5: Feedbacked Ensemble Modelling on Syntactico-Semantic Knowledge for Aspect Based Sentiment Analysis. *Proceedings of SemEval* .

Rich Caruana. 1993. Multitask Learning: A Knowledge-Based Source of Inductive Bias. In *Proceedings of ICML*.

Hongshen Chen, Yue Zhang, and Qun Liu. 2016. Neural Network for Heterogeneous Annotations. In *Proceedings of EMNLP*.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Recurrent neural network-based sentence encoder with gated attention for natural language inference. *arXiv preprint arXiv:1708.01353* .

Ronan Collobert and Jason Weston. 2008. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *Proceedings of ICML*.

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

Hal Daumé III. 2009. Bayesian multitask learning with latent hierarchies. In *Proceedings of UAI*.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive Recursive Neural Network for Target-dependent Twitter Sentiment Classification. In *Proceedings of ACL*. pages 49–54.

Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bosnjak, and Sebastian Riedel. 2016. emoji2vec: Learning Emoji Representations from their Description. In *Proceedings of SocialNLP*.

Theodoros Evgeniou, Charles A. Micchelli, and Massimiliano Pontil. 2005. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research* 6:615–637.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of EMNLP*.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A Joint Many-Task Model: Growing a Neural Network for Multiple NLP Tasks. In *Proceedings of EMNLP*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531* .

Laurent Jacob, Jean-Philippe Vert, Francis R Bach, and Jean-philippe Vert. 2009. Clustered Multi-Task Learning: A Convex Formulation. In *Proceedings of NIPS*. pages 745–752.

Robert a. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. 1991. Adaptive Mixtures of Local Experts. *Neural Computation* 3(1):79–87.

Zhuoliang Kang, Kristen Grauman, and Fei Sha. 2011. Learning with Whom to Share in Multi-task Feature Learning. In *Proceedings of ICML*.

Young-Bum Kim, Karl Stratos, Ruhi Sarikaya, and Minwoo Jeong. 2015. New Transfer Learning Techniques for Disparate Label Sets. In *Proceedings of ACL*.

Abhishek Kumar and Hal Daumé III. 2012. Learning Task Grouping and Overlap in Multi-task Learning. *Proceedings of the 29th International Conference on Machine Learning* pages 1383–1390.

Ayush Kumar, Sarah Kohail, Amit Kumar, Asif Ekbal, and Chris Biemann. 2016. IIT-TUDA at SemEval-2016 Task 5: Beyond Sentiment Lexicon: Combining Domain Dependency and Distributional Semantics Features for Aspect Based Sentiment Analysis. *Proceedings of SemEval* .

Ming Li and Zhi-Hua Zhou. 2007. Improve Computer-Aided Diagnosis With Machine Learning Techniques Using Undiagnosed Samples. *IEEE Transactions on Systems, Man and Cybernetics* 37(6):1088–1098.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial Multi-task Learning for Text Classification. In *Proceedings of ACL*.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task Sequence to Sequence Learning. In *Proceedings of ICLR*.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of SemEval*.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Veselin Stoyanov, and Fabrizio Sebastiani. 2016. SemEval-2016 Task 4: Sentiment Analysis in Twitter. In *Proceedings of SemEval*. San Diego, California.

Nikita Nangia, Adina Williams, Angeliki Lazaridou, and Samuel R. Bowman. 2017. The RepEval 2017 Shared Task: Multi-Genre Natural Language Inference with Sentence Representations. In *Proceedings of RepEval*.

Elisavet Palogiannidi, Athanasia Kolovou, Fenia Christopoulou, Filippos Kokkinos, Elias Iosif, Nikolaos Malandrakis, Haris Papageorgiou, Shrikanth Narayanan, and Alexandros Potamianos. 2016. Tweester at SemEval-2016 Task 4: Sentiment Analysis in Twitter Using Semantic-Affective Model Adaptation. In *Proceedings of SemEval*. pages 155–163.

Hao Peng, Sam Thomson, Noah A Smith, and Paul G Allen. 2017. Deep Multitask Learning for Semantic Dependency Parsing. In *Proceedings of ACL 2017*.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss. In *Proceedings of ACL*.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammed AL-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, Veronique Hoste, Marianna Apidianaki, Xavier Tannier, Natalia Loukachevitch, Evgeniy Kotelnikov, Núria Bel, Salud Maria Jiménez-Zafra, and Gülşen Eryiğit. 2016. SemEval-2016 Task 5: Aspect Based Sentiment Analysis. In *Proceedings of SemEval*.

Marek Rei. 2017. Semi-supervised Multitask Learning for Sequence Labeling. In *Proceedings of ACL 2017*.

Benjamin Riedel, Isabelle Augenstein, Georgios P Spithourakis, and Sebastian Riedel. 2017. A simple but tough-to-beat baseline for the Fake News Challenge stance detection task. In *arXiv preprint arXiv:1707.03264*.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation Extraction with Matrix Factorization and Universal Schemas. *Proceedings of NAACL-HLT* pages 74–84.

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2017. Sluice networks: Learning what to share between loosely related tasks. In *CoRR, abs/1705.08142*.

Sebastian Ruder, Parsa Ghaffari, and John G. Breslin. 2016. A Hierarchical Model of Reviews for Aspect-based Sentiment Analysis. *Proceedings of EMNLP* pages 999–1005.

Sebastian Ruder and Barbara Plank. 2017. Learning to select data for transfer learning with Bayesian Optimization. In *Proceedings of EMNLP*.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of ACL*.

Duy-Tin Vo and Yue Zhang. 2015. Target-Dependent Twitter Sentiment Classification with Rich Automatic Features. In *Proceedings of IJCAI*. pages 1347–1353.

Ya Xue, Xuejun Liao, Lawrence Carin, and Balaji Krishnapuram. 2007. Multi-Task Learning for Classification with Dirichlet Process Priors. *Journal of Machine Learning Research* 8:35–63.

Chih-Kuan Yeh, Wei-Chieh Wu, Wei-Jen Ko, and Yu-Chiang Frank Wang. 2017. Learning Deep Latent Space for Multi-Label Classification. In *Proceedings of AAAI*.

Kai Yu, Volker Tresp, and Anton Schwaighofer. 2005. Learning Gaussian processes from multiple tasks. *Proceedings of ICML* 22:1012–1019.

Yuan Zhang, Roi Reichart, Regina Barzilay, and Amir Globerson. 2012. Learning to Map into a Universal POS Tagset. In *Proceedings of EMNLP*.

Word Emotion Induction for Multiple Languages as a Deep Multi-Task Learning Problem

Sven Buechel & Udo Hahn
{sven.buechel|udo.hahn}@uni-jena.de
Jena University Language & Information Engineering (JULIE) Lab
Friedrich-Schiller-Universität Jena, Jena, Germany
http://www.julielab.de

Abstract

Predicting the emotional value of lexical items is a well-known problem in sentiment analysis. While research has focused on polarity for quite a long time, meanwhile this early focus has been shifted to more expressive emotion representation models (such as Basic Emotions or Valence-Arousal-Dominance). This change resulted in a proliferation of heterogeneous formats and, in parallel, often small-sized, non-interoperable resources (lexicons and corpus annotations). In particular, the limitations in size hampered the application of deep learning methods in this area because they typically require large amounts of input data. We here present a solution to get around this language data bottleneck by rephrasing word emotion induction as a multi-task learning problem. In this approach, the prediction of each independent emotion dimension is considered as an individual task and hidden layers are *shared* between these dimensions. We investigate whether multi-task learning is more advantageous than single-task learning for emotion prediction by comparing our model against a wide range of alternative emotion and polarity induction methods featuring 9 typologically diverse languages and a total of 15 conditions. Our model turns out to outperform each one of them. Against all odds, the proposed deep learning approach yields *the largest gain* on *the smallest* data sets, merely composed of one thousand samples.

1 Introduction

Deep Learning (DL) has radically changed the rules of the game in NLP by dramatically boosting performance figures in almost all applications areas. Yet, one of the major premises of high-performance DL engines is their dependence on huge amounts of training data. As such, DL seems ill-suited for areas where training data are scarce, such as in the field of word emotion induction.

We will use the terms *polarity* and *emotion* here to distinguish between research focusing on "semantic orientation" (Hatzivassiloglou and McKeown, 1997) (the positiveness or negativeness) of affective states, on the one hand, and approaches which provide predictions based on some of the many more elaborated representational systems for affective states, on the other hand.

Originally, research activities focused on polarity alone. In the meantime, a shift towards more expressive representation models for emotion can be observed that heavily draws inspirations from psychological theory, e.g., Basic Emotions (Ekman, 1992) or the Valence-Arousal-Dominance model (Bradley and Lang, 1994).

Though this change turned out to be really beneficial for sentiment analysis in NLP, a large variety of mutually incompatible encodings schemes for emotion and, consequently, annotation formats for emotion metadata in corpora have emerged that hinder the interoperability of these resources and their subsequent reuse, e.g., on the basis of alignments or mergers (Buechel and Hahn, 2017).

As an alternative way of dealing with thus unwarranted heterogeneity, we here examine the potential of multi-task learning (MTL; Caruana (1997)) for word-level emotion prediction. In MTL for neural networks, a single model is fitted to solve multiple, independent tasks (in our case, to predict different emotional dimensions) which typically results in learning more robust and meaningful intermediate representations. MTL has been shown to greatly decrease the risk of overfitting (Baxter, 1997), work well for various NLP tasks (Setiawan et al., 2015; Liu et al., 2015; Søgaard and Goldberg, 2016; Cummins et al., 2016; Liu et al., 2017; Peng et al., 2017), and practically increases sample size, thus making it a natural choice for small-sized data sets typically found in the area of word emotion induction.

Proceedings of NAACL-HLT 2018, pages 1907–1918
New Orleans, Louisiana, June 1 - 6, 2018.

After a discussion of related work in Section 2, we will introduce several reference methods and describe our proposed deep MTL model in Section 3. In our experiments (Section 4), we will first validate our claim that MTL is superior to single-task learning for word emotion induction. After that, we will provide a large-scale evaluation of our model featuring 9 typologically diverse languages and multiple publicly available embedding models for a total of 15 conditions. Our MTL model surpasses the current state-of-the-art for each of them, and even performs competitive relative to human reliability. Most notably however, our approach yields the largest benefit on the smallest data sets, comprising merely one thousand samples. This finding, counterintuitive as it may be, strongly suggests that MTL is particularly beneficial for solving the word emotion induction problem. Our code base as well as the resulting experimental data is freely available.[1]

2 Related Work

This section introduces the emotion representation format underlying our study and describes external resources we will use for evaluation before we discuss previous methodological work.

Emotion Representation and Data Sets. Psychological models of emotion can typically be subdivided into *discrete* (or *categorical*) and *dimensional* ones (Stevenson et al., 2007; Calvo and Mac Kim, 2013). Discrete models are centered around particular sets of emotional categories considered to be fundamental. Ekman (1992), for instance, identifies six *Basic Emotions* (Joy, Anger, Sadness, Fear, Disgust and Surprise).

In contrast, dimensional models consider emotions to be composed of several influencing factors (mainly two or three). These are often referred to as *Valence* (a positive–negative scale), *Arousal* (a calm–excited scale), and *Dominance* (perceived degree of control over a (social) situation)—the VAD model (Bradley and Lang (1994); see Figure 1 for an illustration). Many contributions though omit Dominance (the VA model) (Russell, 1980). For convenience, we will still use the term "VAD" to jointly refer to both variants (with and without Dominance).

VAD is the most common framework to acquire empirical emotion values for words in psychology.

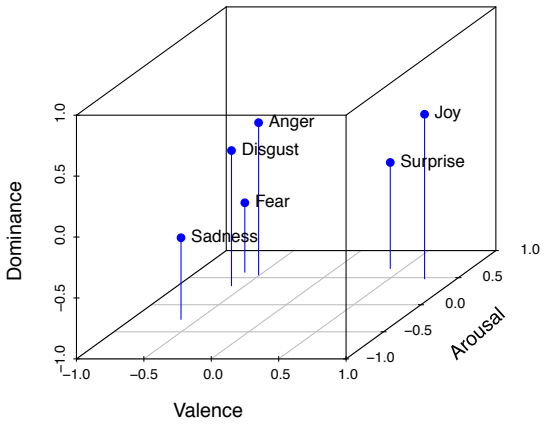


Figure 1: Affective space spanned by the Valence-Arousal-Dominance (VAD) model, together with the position of six Basic Emotions; as determined by Russell and Mehrabian (1977).

Over the years, a considerable number of such resources (also called "emotion lexicons") have emerged from psychological research labs (as well as some NLP labs) for diverse languages. The emotion lexicons we use in our experiments are listed in Table 1. An even more extensive list of such data sets is presented by Buechel and Hahn (2018). For illustration, we also provide three sample entries from one of those lexicons in Table 2. As can be seen, the three affective dimensions behave complementary to each other, e.g., "terrorism" and "orgasm" display similar Arousal but opposing Valence.

The task we address in this paper is to predict the values for Valence, Arousal and Dominance, given a lexical item. As is obvious from these examples, we consider emotion prediction as a regression, not as a classification problem (see arguments discussed in Buechel and Hahn (2016)).

In this paper, we focus on the VAD format for the following reasons: First, note that the Valence dimension exactly corresponds to polarity (Turney and Littman, 2003). Hence, with the VAD model, emotion prediction can be seen as a generalization over classical polarity prediction. Second, to the best of our knowledge, the amount and diversity of available emotion lexicons with VAD encodings is larger than for any other format (see Table 1).

Word Embeddings. Word embeddings are dense, low-dimensional vector representations of words trained on large volumes of raw text in an unsupervised manner. The following are among today's most popular embedding algorithms:

[1] https://github.com/JULIELab/wordEmotions

Source	ID	Language	Format	# Entries
Bradley and Lang (1999)	EN	English	VAD	1,034
Warriner et al. (2013)	EN+	English	VAD	13,915
Redondo et al. (2007)	ES	Spanish	VAD	1,034
Stadthagen-Gonzalez et al. (2017)	ES+	Spanish	VA	14,031
Schmidtke et al. (2014)	DE	German	VAD	1,003
Yu et al. (2016a)	ZH	Chinese	VA	2,802
Imbir (2016)	PL	Polish	VAD	4,905
Montefinese et al. (2014)	IT	Italian	VAD	1,121
Soares et al. (2012)	PT	Portuguese	VAD	1,034
Moors et al. (2013)	NL	Dutch	VAD	4,299
Sianipar et al. (2016)	ID	Indonesian	VAD	1,490

Table 1: Emotion lexicons used in our experiments (with their bibliographic source, identifier, language they refer to, emotion representation format, and number of lexical entries they contain).

Word	Valence	Arousal	Dominance
sunshine	8.1	5.3	5.4
terrorism	1.6	7.4	2.7
orgasm	8.0	7.2	5.8

Table 2: Three sample entries from Warriner et al. (2013). They use 9-point scales ranging from 1 (most negative/calm/submissive) to 9 (most positive/excited/dominant).

WORD2VEC (with its variants SGNS and CBOW) features an extremely trimmed down neural network (Mikolov et al., 2013). FASTTEXT is a derivative of WORD2VEC, also incorporating sub-word character n-grams (Bojanowski et al., 2017). Unlike the former two algorithms which fit word embeddings in a streaming fashion, GLOVE trains word vectors directly on a word co-occurrence matrix under the assumption to make more efficient use of word statistics (Pennington et al., 2014). Somewhat similar, SVD_{PPMI} performs singular value decomposition on top of a point-wise mutual information co-occurrence matrix (Levy et al., 2015).

In order to increase the reproducibility of our experiments, we rely on the following widely used, publicly available embedding models trained on very large corpora (summarized in Table 3): the SGNS model trained on the Google News corpus[2] (GOOGLE), the FASTTEXT model trained on Common Crawl[3] (COMMON), as well as the FASTTEXT models for a wide range of languages trained on the respective Wikipedias[4] (WIKI). Note that WIKI denotes multiple embedding models with different training and vocabulary sizes (see Grave et al. (2018) for further details). Additionally, we were given the opportunity to reuse the English embedding model from Sedoc et al. (2017) (GIGA), a strongly related contribution (see below). Their embeddings were trained on the English Gigaword corpus (Parker et al., 2011).

Word-Level Prediction. One of the early approaches to word *polarity* induction which is still popular today (Köper and Schulte im Walde, 2016) was introduced by Turney and Littman (2003). They compute the polarity of an unseen word based on its point-wise mutual information (PMI) to a set of positive and negative seed words, respectively.

SemEval-2015 Task 10E featured polarity induction on Twitter (Rosenthal et al., 2015). The best system relied on support vector regression (SVR) using a radial base function kernel (Amir et al., 2015). They employ the embedding vector of the target word as features. The results of their SVR-based system were beaten by the DENSIFIER algorithm (Rothe et al., 2016). DENSIFIER learns an orthogonal transformation of an embedding space into a subspace of strongly reduced dimensionality.

Hamilton et al. (2016) developed SENTPROP, a graph-based, semi-supervised learning algorithm which builds up a word graph, where vertices correspond to words (of known as well as unknown polarity) and edge weights correspond to the similarity between them. The polarity information is then propagated through the graph, thus computing scores for unlabeled nodes. According to their evaluation, DENSIFIER seems to be superior overall, yet SENTPROP produces competitive results

[2] https://code.google.com/archive/p/word2vec/

[3] https://fasttext.cc/docs/en/english-vectors.html

[4] https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md

ID	Language	Method	Corpus	# Tokens	# Types	# Dimensions
GOOGLE	English	SGNS	Google News	1×10^{11}	3×10^{6}	300
COMMON	English	FASTTEXT	Common Crawl	6×10^{11}	2×10^{6}	300
GIGA	English	CBOW	Gigawords	4×10^{9}	2×10^{6}	300
WIKI	all	FASTTEXT	Wikipeda	—	—	300

Table 3: Embedding models used for our experiments with identifier, language, embedding algorithm, training corpus, its size in the number of tokens, size of the vocabulary (types) of the resulting embedding model and its dimensionality.

only when the seed lexicon or the corpus the word embeddings are trained on is very small.[5]

For word *emotion* induction, a very similar approach to SENTPROP has been proposed by Wang et al. (2016a). They also propagate affective information (Valence and Arousal, in this case) through a word graph with similarity weighted edges.

Sedoc et al. (2017) recently proposed an approach based on signed spectral clustering where a word graph is constructed not only based on word similarity but also on the considered affective information (again, Valence and Arousal). The emotion value of a target word is then computed based on the seed words in its cluster. They report to outperform the results from Wang et al. (2016a).

Contrary to the trend to graph-based methods, the best system of the IALP 2016 Shared Task on Chinese word emotion induction (Yu et al., 2016b) employed a simple feed-forward neural network (FFNN) with one hidden layer in combination with boosting (Du and Zhang, 2016).

Another very recent contribution which advocates a supervised set-up was published by Li et al. (2017). They propose ridge regression, again using word embeddings as features. Even with this simple approach, they report to outperform many of the above methods in the VAD prediction task.[6]

Sentence-Level and Text-Level Prediction. Different from the word-level prediction task (the one we focus on in this contribution), the determination of emotion values for higher-level linguistic units (especially sentences and texts) is also heavily investigated. For this problem, DL approaches are meanwhile fully established as the method of choice (Wang et al., 2016b; Abdul-Mageed and Ungar, 2017; Felbo et al., 2017; Mohammad and Bravo-Marquez, 2017).

[5]Personal correspondence with William L. Hamilton; See also README at https://github.com/williamleif/socialsent

[6]However, they also report extremely weak performance figures for some of their reference methods.

It is important to note, however, that the methods discussed for these higher-level units cannot easily be transferred to solve the word emotion induction problem. Sentence-level and text-level architectures are either adapted to *sequential* input data (typical for RNN, LSTM, GRNN and related architectures) or *spatially arranged* input data (as with CNN architectures). However, for word embeddings (the default input for word emotion induction) there does not seem to be any meaningful order of their components. Therefore, these more sophisticated DL methods are, for the time being, not applicable for the study at hand.

3 Methods

In this section, we will first introduce various reference methods (two originally polarity-based for which we offer adaptations for VAD prediction) before defining our own neural MTL model and discussing its difference from previous work.

Let $V := \{w_1, w_2, ..., w_m\}$ be our word vocabulary and let $E := \{e_1, e_2, ..., e_m\}$ be a set of embedding vectors such that $e_i \in \mathbb{R}^n$ denotes the n-dimensional vector representation of word w_i. Let $D := \{d_1, d_2, ..., d_l\}$ be a set of emotional dimensions. Our task is to predict the empirically determined emotion vector $emo(w) \in \mathbb{R}^l$ given a word w and the embedding space E.

3.1 Reference Methods

Linear Regression Baseline (LinReg). We propose (multi-variate) linear regression as an obvious baseline for the problem:

$$emo_{LR}(w_k) := We_k + b \quad (1)$$

where W is a matrix, W_{i*} contains the regression coefficients for the i-th affective dimension and b is the vector of bias terms. The model parameters are fitted using ordinary least squares. Technically, we use the `scikit-learn.org` implementation with default parameters.

Ridge Regression (RidgReg). Li et al. (2017) propose ridge regression for word emotion induction. Ridge regression works identically to linear regression during prediction, but introduces L_2 regularization during training. Following the authors, for our implementation, we again use the `scikit-learn` implementation with default parameters.

Turney-Littman Algorithm (TL). As one of the earliest contributions in the field, Turney and Littman (2003) defined a simple PMI-based approach to determine the semantic polarity SP_{TL} of a word w:

$$SP_{TL}(w) := \sum_{s \in seeds^+} pmi(w,s) - \sum_{s \in seeds^-} pmi(w,s) \quad (2)$$

where $seeds^+$ and $seeds^-$ are sets of positive and negative seed words, respectively. Since this algorithm is still popular today (Köper and Schulte im Walde, 2016), we here provide a novel modification for adapting this originally polarity-based approach to word emotion induction with vectorial seed and output values.

First, we replace PMI-based association of seed and target word w and s by their similarity sim based on their word embeddings e_w and e_s:

$$sim(w,s) := max(0, \frac{e_w \cdot e_s}{||e_w|| \times ||e_s||}) \quad (3)$$

$$emo(w) := \sum_{s \in seeds^+} sim(w,s) - \sum_{s \in seeds^-} sim(w,s) \quad (4)$$

Although this step is technically not required for the adaptation, it renders the TL algorithm more comparable to the other approaches evaluated in Section 4 besides from most likely increasing performance. Equation (4) can be rewritten as

$$emo(w) := \sum_{s \in seeds} sim(w,s) \times emo(s) \quad (5)$$

where $seeds := seeds^+ \cup seeds^-$ and $emo(s)$ maps to 1, if $s \in seeds^+$, and -1, if $s \in seeds^-$.

Equation (5) can be trivially adapted to an n-dimensional emotion format by redefining $emo(s)$ such that it maps to a vector from $\mathbb{R}^n$ instead of $\{-1, 1\}$. Our last step is to introduce a normalization term such that $emo(w)_{TL}$ lies within the range of the seed lexicon.

$$emo_{TL}(w) := \frac{\sum_{s \in seeds} sim(w,s) \times emo(s)}{\sum_{s \in seeds} sim(w,s)} \quad (6)$$

As can be seen from Equation (6), for the more general case of n-dimensional emotion prediction, the Turney-Littman algorithm naturally translates into a weighted average where the seed emotion values are weighted according to the similarity to the target item.

Densifier. Rothe et al. (2016) train an orthogonal matrix $Q \in \mathbb{R}^{n \times n}$ (n being the dimensionality of the word embeddings) such that applying Q to an embedding vector e_i concentrates all the polarity information in its first dimension such that the polarity of a word w_i can be computed as

$$SP_{\text{DENSIFIER}}(w_i) := pQe_i \quad (7)$$

where $p = (1, 0, 0, ..., 0)^T \in \mathbb{R}^{1 \times n}$.

For fitting Q, the seeds are arranged into pairs of equal polarity (the set $pairs^=$) and those of opposing polarity ($pairs^\neq$). A good fit for Q will minimize the distance within the former and maximize the distance within the latter which can be expressed by the following two training objectives:

$$\underset{Q}{\text{argmin}} \sum_{(w_i,w_j) \in pairs^=} |pQ(e_i - e_j)| \quad (8)$$

$$\underset{Q}{\text{argmax}} \sum_{(w_i,w_j) \in pairs^\neq} |pQ(e_i - e_j)| \quad (9)$$

The objectives described in the expressions (8) and (9) are combined into a single loss function (using a weighting factor $\alpha \in [0, 1]$) which is then minimized using stochastic gradient descent (SGD).

To adapt this algorithm to dimensional emotion formats, we construct a positive seed set, $seeds_v^+$, and a negative seed set, $seeds_v^-$, for each emotion dimension $v \in D$. Let M_v be the mean value of all the entries of the training lexicon for the affective dimension v. Let SD_v be the respective standard deviation and $\beta \in \mathbb{R}$, $\beta \geq 0$. Then all entries greater than $M_v + \beta SD_v$ are assigned to $seeds_v^+$ and those less than $M_v - \beta SD_v$ are assigned to $seeds_v^-$. Q is fitted individually for each emotion dimension v.

Training was performed according to the original paper with the exception that (following Hamilton et al. (2016)) we did not apply the proposed re-orthogonalization after each training

step, since we did not find any evidence that this procedure actually results in improved performance. The hyperparameters α and β were set to .7 and .5 (respectively) for all experiments based on a pilot study. Since the original implementation is not accessible, we devised our own using `tensorflow.org`.

Boosted Neural Networks (ensembleNN). Du and Zhang (2016) propose simple FFNNs in combination with a boosting algorithm. An FFNN consists of an *input* or *embedding layer* with activation $a^{(0)} \in \mathbb{R}^n$ which is equal to the embedding vector e_k when predicting the emotion of a word w_k. The input layer is followed by multiple hidden layers with activation

$$a^{(l+1)} := \sigma(W^{(l+1)}a^{(l)} + b^{(l+1)}) \qquad (10)$$

where $W^{(l+1)}$ and $b^{(l+1)}$ are the weights and biases for layer $l+1$ and σ is a nonlinear activation function. Since we treat emotion prediction as a regression problem, the activation on the output layer a^{out} (where out is the number of non-input layers in the network) is computed as the affine transformation

$$a^{(out)} := W^{(out)}a^{(out-1)} + b^{(out)} \qquad (11)$$

Boosting is a general machine learning technique where several weak estimators are combined to form a strong estimator. The authors used FFNNs with a single hidden layer of 100 units and rectified linear unit (ReLU) activation. The boosting algorithm AdaBoost.R2 (Drucker, 1997) was used to train the ensemble (one per affective dimension). Our re-implementation copies their technical set-up[7] exactly using `scikit-learn`.

3.2 Multi-Task Learning Neural Network

The approaches introduced in Section 3.1 and Section 2 vary largely in their methodological foundations, i.e., they comprise semi-supervised and supervised machine learning techniques—both statistical and neural ones. Yet, they all have in common that they treat the prediction of the different emotional dimensions *as separate* tasks. That is, they fit one individual model per VAD dimension without sharing parameters between them.

In contradistinction, the key feature of our approach is that we fit a single FFNN model to predict *all VAD dimensions jointly*, thus applying multi-task learning to word emotion induction. Hence, we treat the prediction of Valence, Arousal and Dominance as three independent tasks. Our multi-task learning neural network (MTLNN) (depicted in Figure 2) has an output layer of three units such that each output unit represents one of the VAD dimensions. However, the activation in our two hidden layers (of 256 and 128 units, respectively) is *shared* across all VAD dimensions, and so are the associated weights and biases.

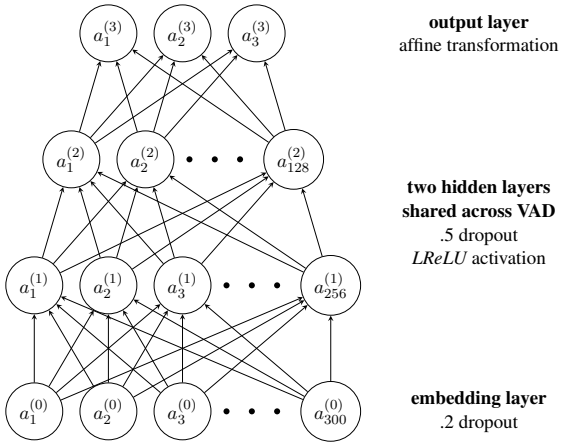


Figure 2: MTL architecture for VAD prediction.

Thus, while we train our MTLNN model it is forced to learn intermediate representations of the input which are generally informative for all VAD dimensions. This serves as a form of regularization, since it becomes less likely for our model to fit the noise in the training set as noise patterns may vary across emotional dimensions. Simultaneously, this has an effect similar to an increase of the training size, since each sample now leads to additional error signals during backpropagation. Intuitively, both properties seem extremely useful for relatively small-sized emotion lexicons (see Section 4 for empirical evidence).

The remaining specifications of our model are as follows. We use *leaky* ReLU activation (LReLU) as nonlinearity (Maas et al., 2013).

$$LReLU(z_i) := max(\gamma z_i, z_i) \qquad (12)$$

with $\gamma := .01$ for our experiments. For regularization, dropout (Srivastava et al., 2014) is applied during training with a probability of .2 on the embedding layer and .5 on the hidden layers. We train for $15,000$ iterations (well beyond convergence on each data set we use) with the ADAM optimizer (Kingma and Ba, 2015) of .001 base learning rate,

[7] Original settings available at `https://github.com/StevenLOL/ialp2016_Shared_Task`

batch size of 128 and Mean-Squared-Error loss. The weights are randomly initialized (drawn from a normal distribution with a standard deviation .001) and biases are uniformly initialized as .01. `Tensorflow` is used for implementation.

4 Results

In this section, we first validate our assumption that MTL is superior to single-task learning for word emotion induction. Next, we compare our proposed MTLNN model in a large-scale evaluation experiment.

Performance figures will be measured as Pearson correlation (r) between our automatically predicted values and human gold ratings. The Pearson correlation between two data series $X = x_1, x_2, ..., x_n$ and $Y = y_1, y_2, ..., y_n$ takes values between $+1$ (perfect positive correlation) and -1 (perfect negative correlation) and is computed as

$$r_{xy} := \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \quad (13)$$

where $\bar{x}$ and $\bar{y}$ denote the mean values for X and Y, respectively.

4.1 Single-Task vs. Multi-Task Learning

The main hypothesis of this contribution is that an MTL set-up is superior to single-task learning for word emotion induction. Before proceeding to the large-scale evaluation of our proposed model, we will first examine this aspect of our work.

For this, we use the following experimental set-up: We will compare the MTLNN model against its single-task learning counterpart (SepNN). SepNN simultaneously trains three separate neural networks where only the input layer, yet no parameters of the intermediate layers are shared across the models. Each of the separate networks is identical to MTLNN (same layers, dropout, initialization, etc.), yet has only one output neuron, thus modeling only one of the three affective VAD dimensions. SepNN is equivalent to fitting our proposed model (but with only one output unit) to the different VAD dimensions individually, one after the other. Yet, training these separate networks simultaneously (not jointly!) makes both approaches, MTLNN and SepNN, easier to compare.

We will run MTLNN against SepNN on the EN and the EN+ data set (the former is very small, the latter relatively large; see Table 1) using the following set-up: for each gold lexicon and model, we randomly split the data $9/1$ and train for $15,000$ iterations on the larger split (the same number of steps is used for the main experiment). After each one-thousand iterations step, model performance is tested on the held-out data. This process will be repeated 20 times and the performance figures at each one-thousand iterations step will be averaged. In a final step, we will average the results for each of the three emotional dimensions and only plot this average value. The results of this experiment are depicted in Figure 3.

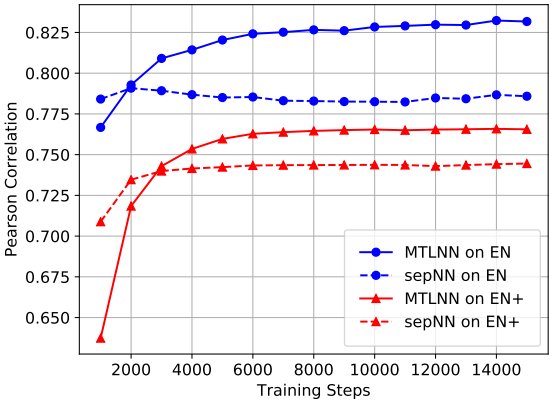


Figure 3: Performance of our proposed MTLNN model vs. its single-task learning counterpart SepNN against training steps.

First of all, each combination of model and data set displays a satisfactory performance of at least $r \approx .75$ after 15,000 steps compared to previous work (see below). Overall, performance is higher for the smaller EN lexicon. Although counterintuitive (since smaller lexicons lead to fewer training samples), this finding is consistent with prior work (Sedoc et al., 2017; Li et al., 2017) and is probably related to the fact that smaller lexicons usually comprise a larger portion of strongly emotion-bearing words. In contrast, larger lexicons add more neutral words which tend to be harder to predict in terms of correlation.

As hypothesized, the MTLNN model does indeed outperform the single task model on both data sets. Our data also suggest that the gain from the MTL approach is larger on smaller data sets (again in concordance with our expectations). Figure 3 reveals that this might be due to the regularizing effect of MTL, since the SepNN model shows signs of overfitting on the EN data set. Yet, even

Language	Data	Embeddings	LinReg	RidgReg	TL	Densifier	ensembleNN	MTLNN
English	EN+	GOOGLE	0.696	0.696	0.631	0.622	0.728	**0.739*****
English	EN+	COMMON	0.719	0.719	0.659	0.652	0.762	**0.767*****
English	EN+	WIKI	0.666	0.666	0.591	0.584	0.706	**0.712*****
English	EN	GOOGLE	0.717	0.732	0.723	0.712	0.688	**0.810*****
English	EN	COMMON	0.731	0.741	0.741	0.726	0.717	**0.824*****
English	EN	WIKI	0.656	0.667	0.674	0.665	0.681	**0.777*****
Spanish	ES	WIKI	0.698	0.709	0.704	0.690	0.700	**0.804*****
Spanish	ES+	WIKI	0.693	0.694	0.603	0.598	0.766	**0.778*****
German	DE	WIKI	0.709	0.719	0.714	0.710	0.700	**0.801*****
Chinese	ZH	WIKI	0.716	0.717	0.586	0.599	0.737	**0.744****
Polish	PL	WIKI	0.650	0.650	0.577	0.553	0.687	**0.712*****
Italian	IT	WIKI	0.656	0.665	0.672	0.659	0.630	**0.751*****
Portuguese	PT	WIKI	0.673	0.684	0.685	0.678	0.672	**0.768*****
Dutch	NL	WIKI	0.651	0.652	0.559	0.532	0.704	**0.730*****
Indonesian	ID	WIKI	0.581	0.586	0.581	0.576	0.575	**0.660*****
Average			0.638	0.659	0.611	0.605	0.676	**0.728*****

Table 4: Results of our main experiment in averaged Pearson correlation; best result per condition (in rows) in bold, second best result underlined; significant difference (paired two-tailed t-test) over the second best system marked with "*", "**", or "***" for $p < .05$, .01, or .001, respectively.

when the separate model does not overfit (as on the EN+ lexicon), MTLNN reveals better results.

Although SepNN needs fewer *training steps* before convergence, the MTLNN model trains much faster, thus still converging faster in terms of *runtime* (about a minute on a middle-class GPU). This is because MTLNN has only about a third as many parameters as the separate model SepNN.

4.2 Comparison against Reference Methods

We combined each of the selected lexicon data sets (Table 1) with each of the applicable publicly available embedding models (Section 2; the embedding model provided by Sedoc et al. (2017) will be used separately) for a total of 15 conditions, i.e, the rows in Table 4.

For each of these conditions, we performed a 10-fold cross-validation (CV) for each of the 6 methods presented in Section 3 such that each method is presented with the identical data splits.[8] For each condition, algorithm, and VA(D) dimension, we compute the Pearson correlation r between gold ratings and predictions. For conciseness, we present only the average correlation over the respective affective dimensions in Table 4 (Valence and Arousal for ES+ and ZH, VAD for the others). Note that the methods we compare ourselves against comprise the current state-of-the art in both polarity and emotion induction (as described in Section 2).

[8]This procedure constitutes a more direct comparison than using different splits for each method and allows using *paired* t-tests.

As can be seen, our proposed MTLNN model outperforms all other approaches in each of the 15 conditions. Regarding the average over all affective dimensions and conditions, it outperforms the second best system, ensembleNN, by more than 5%-points. In line with our results from Section 4.1, those improvements are especially pronounced on smaller data sets containing one up to two thousand entries (EN, ES, IT, PT, ID) with close to 10%-points improvement over the respective second-best system.

Concerning the relative ordering of the affective dimensions, in line with former studies (Sedoc et al., 2017; Li et al., 2017), the performance figures for the Valence dimension are usually much higher than for Arousal and Dominance. Using MTLNN, for many conditions, we see the pattern that Valence is about 10%-points above the VAD average, Arousal being 10%-points below and Dominance being roughly equal to the average over VAD (this applies, e.g., to EN, EN+ and IT). On other data sets (e.g., PL, NL and ID), the ordering between Arousal and Dominance is less clear though Valence still stands out with the best results. We observe the same general pattern for the reference methods, as well.

Concerning the comparison to Sedoc et al. (2017), arguably one of most related contributions, they report a performance of $r = .768$ for Valence and .582 for Arousal on the EN+ data set in a 10-fold CV using their own embeddings. In contrast, MTLNN using the COMMON model achieves $r = .870$ and .674 in the same set-up—about 10%-

	Valence	Arousal	Dominance
MTLNN EN	.918	.730	.825
MTLNN EN+	.870	.674	.758
ISR EN $\sim$ EN+	.953	.759	.795
SHR EN+	.914	.689	.770

Table 5: Comparison of the MTLNN model against inter-study reliability (ISR) between the EN and the EN+ data set and split-half reliability (SHR) of the EN+ data set (in Pearson correlation).

points better on both dimensions. However, the COMMON model was trained on much more data than the embeddings Sedoc et al. (2017) use. For the most direct comparison, we also repeated this experiment using *their* embedding model (GIGA). We find that MTLNN still clearly outperforms their results with $r = .814$ for Valence and .607 for Arousal.[9]

MTLNN achieves also very strong results in direct comparison to human performance (see Table 5). Warriner et al. (2013) (who created EN+) report an inter-study reliability (ISR; i.e., the correlation of the aggregated ratings from two different studies) between the EN and the EN+ lexicon of $r = .953$, .759 and .795 for VAD, respectively. Since EN is a subset of EN+, we can compare these performance figures against our own results on the EN data set where we achieved $r = .918$, .730 and .825, respectively. Thus, our proposed method did actually outperform human reliability for Dominance and is competitive for Valence and Arousal, as well.

This general observation is also backed up by split-half reliability data (SHR; i.e., when randomly splitting all individual ratings in two groups and averaging the ratings within each group, how strong is the correlation between these averaged ratings?). For the EN+ data set, Warriner et al. (2013) report an SHR of $r = .914$, .689 and .770 for VAD, respectively. Again, our MTLNN model performs very competitive with $r = .870$, .674 and .758, respectively using the COMMON embeddings.

5 Conclusion

In this paper, we propose multi-task learning (MTL) as a simple, yet surprisingly efficient method to improve the performance and, at the same time, to deal with existing data limitations in word emotion induction—the task to predict a complex emotion score for an individual word. We validated our claim that MTL is superior to single-task learning by achieving better results with our proposed method in performance as well as training time compared to its single-task counterpart. We performed an extensive evaluation of our model on 9 typologically diverse languages, using different kinds of word embedding models for a total 15 conditions. Comparing our approach to state-of-the-art methods from word polarity and word emotion induction, our model turns out to be superior in each condition, thus setting a novel state-of-the-art performance for both polarity *and* emotion induction. Moreover, our results are even competitive to human annotation reliability in terms of inter-study as well as split-half reliability. Since this contribution was restricted to the VAD format of emotion representation, in future work we will examine whether MTL yields similar gains for other representational schemes, as well.

[9]We also clearly outperform their results for the NL and ES+ data sets. For these cases, our embedding models were similar in training size.

Acknowledgments

We would like to thank the Positive Psychology Center, University of Pennsylvania for providing us with the embedding model used in Sedoc et al. (2017), Johannes Hellrich, JULIE Lab, for insightful discussions, and the reviewers for their valuable comments.

References

Muhammad Abdul-Mageed and Lyle H. Ungar. 2017. EMONET: Fine-grained emotion detection with gated recurrent neural networks. In *ACL 2017 — Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Vancouver, British Columbia, Canada, July 30 - August 4, 2017, volume 1: Long Papers, pages 718–728.

Silvio Amir, Ramón F. Astudillo, Wang Ling, Bruno Martins, Mário J. Silva, and Isabel Trancoso. 2015. INESC-ID: A regression model for large scale Twitter sentiment lexicon induction. In *SemEval 2015 — Proceedings of the 9th International Workshop on Semantic Evaluation @ NAACL-HLT 2015*. Denver, Colorado, USA, June 4-5, 2015, pages 613–618.

Jonathan Baxter. 1997. A Bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning* 28(1):7–39.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomáš Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5(1):135–146.

Margaret M. Bradley and Peter J. Lang. 1994. Measuring emotion: The Self-Assessment Manikin and the semantic differential. *Journal of Behavior Therapy and Experimental Psychiatry* 25(1):49–59.

Margaret M. Bradley and Peter J. Lang. 1999. Affective Norms for English Words (ANEW): Stimuli, instruction manual and affective ratings. Technical Report C-1, The Center for Research in Psychophysiology, University of Florida, Gainesville, FL.

Sven Buechel and Udo Hahn. 2016. Emotion analysis as a regression problem: Dimensional models and their implications on emotion representation and metrical evaluation. In *ECAI 2016 — Proceedings of the 22nd European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS 2016)*. The Hague, Netherlands, August 29 - September 2, 2016, volume 285 of *Frontiers in Artificial Intelligence and Applications*, pages 1114–1122.

Sven Buechel and Udo Hahn. 2017. EMOBANK: Studying the impact of annotation perspective and representation format on dimensional emotion analysis. In *EACL 2017 — Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. Valencia, Spain, April 3–7, 2017, volume 2: Short Papers, pages 578–585.

Sven Buechel and Udo Hahn. 2018. Representation mapping: A novel approach to generate high-quality multi-lingual emotion lexicons. In *LREC 2018 — Proceedings of the 11th International Conference on Language Resources and Evaluation*. Miyazaki, Japan, May 7–12, 2018.

Rafael A. Calvo and Sunghwan Mac Kim. 2013. Emotions in text: Dimensional and categorical models. *Computational Intelligence* 29(3):527–543.

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

Ronan Cummins, Meng Zhang, and Ted Briscoe. 2016. Constrained multi-task learning for automated essay scoring. In *ACL 2016 — Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Berlin, Germany, August 7-12, 2016, volume 2: Short Papers, pages 789–799.

Harris Drucker. 1997. Improving regressors using boosting techniques. In *ICML '97 — Proceedings of the 14th International Conference on Machine Learning*. Nashville, Tennessee, USA, July 8-12, 1997, pages 107–115.

Steven Du and Xi Zhang. 2016. Aicyber's system for IALP 2016 Shared Task: Character-enhanced word vectors and boosted neural networks. In *IALP 2016 — Proceedings of the [20th] 2016 International Conference on Asian Language Processing*. Tainan, Taiwan, November 21-23, 2016, pages 161–163.

Paul Ekman. 1992. An argument for basic emotions. *Cognition & Emotion* 6(3-4):169–200.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *EMNLP 2017 — Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark, September 9-11, 2017, pages 1615–1625.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomáš Mikolov. 2018. Learning word vectors for 157 languages. In *LREC 2018 — Proceedings of the 11th International Conference on Language Resources and Evaluation*. Miyazaki, Japan, May 7–12, 2018.

William L. Hamilton, Kevin Clark, Jure Leskovec, and Daniel Jurafsky. 2016. Inducing domain-specific sentiment lexicons from unlabeled corpora. In *EMNLP 2016 — Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas, USA, November 1-5, 2016, pages 595–605.

Vasileios Hatzivassiloglou and Kathleen R. McKeown. 1997. Predicting the semantic orientation of adjectives. In *ACL-EACL 1997 — Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics & 8th Conference of the European Chapter of the Association for Computational Linguistics*. Madrid, Spain, July 7-12, 1997, pages 174–181.

Kamil K. Imbir. 2016. Affective Norms for 4900 Polish Words Reload (ANPW_R): Assessments for valence, arousal, dominance, origin, significance, concreteness, imageability and, age of acquisition. *Frontiers in Psychology* 7:#1081.

Diederik Kingma and Jimmy Ba. 2015. ADAM: A method for stochastic optimization. In *ICLR 2015 — Proceedings of the 3rd International Conference on Learning Representations*. San Diego, California, USA, May 7-9, 2015.

Maximilian Köper and Sabine Schulte im Walde. 2016. Automatically generated affective norms of abstractness, arousal, imageability and valence for 350,000 German lemmas. In *LREC 2016 — Proceedings of the 10th International Conference on Language Resources and Evaluation*. Portorož, Slovenia, 23-28 May 2016, pages 2595–2598.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics* 3:211–225.

Minglei Li, Qin Lu, Yunfei Long, and Lin Gui. 2017. Inferring affective meanings of words from word embedding. *IEEE Transactions on Affective Computing* 8(4):443–456.

PengFei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *ACL 2017 — Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Vancouver, British Columbia, Canada, July 30 - August 4, 2017, volume 1: Long Papers, pages 1–10.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *NAACL-HLT 2015 — Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Denver, Colorado, USA, May 31 - June 5, 2015, pages 912–921.

Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the Workshop on Deep Learning for Audio, Speech and Language Processing @ ICML 2013*. Atlanta, Georgia, USA, 16 June 2013.

Tomáš Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26 — NIPS 2013. Proceedings of the 27th Annual Conference on Neural Information Processing Systems*. Lake Tahoe, Nevada, USA, December 5-10, 2013, pages 3111–3119.

Saif M. Mohammad and Felipe Bravo-Marquez. 2017. WASSA-2017 Shared Task on Emotion Intensity. In *WASSA 2017 — Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis @ EMNLP 2017*. Copenhagen, Denmark, September 8, 2017, pages 34–49.

Maria Montefinese, Ettore Ambrosini, Beth Fairfield, and Nicola Mammarella. 2014. The adaptation of the Affective Norms for English Words (ANEW) for Italian. *Behavior Research Methods* 46(3):887–903.

Agnes Moors, Jan De Houwer, Dirk Hermans, Sabine Wanmaker, Kevin Van Schie, Anne-Laura Van Harmelen, Maarten De Schryver, Jeffrey De Winne, and Marc Brysbært. 2013. Norms of valence, arousal, dominance, and age of acquisition for 4,300 Dutch words. *Behavior Research Methods* 45(1):169–177.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English Gigaword Fifth Edition. Technical Report LDC2011T07, Linguistic Data Consortium, Philadelphia, Pennsylvania, USA. https://catalog.ldc.upenn.edu/LDC2011T07.

Hao Peng, Sam Thomson, and Noah A. Smith. 2017. Deep multitask learning for semantic dependency parsing. In *ACL 2017 — Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Vancouver, British Columbia, Canada, July 30 - August 4, 2017, volume 1: Long Papers, pages 2037–2048.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GLOVE: Global vectors for word representation. In *EMNLP 2014 — Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Doha, Qatar, October 25-29, 2014, pages 1532–1543.

Jaime Redondo, Isabel Fraga, Isabel Padrón, and Montserrat Comesaña. 2007. The Spanish adaptation of ANEW (Affective Norms for English Words). *Behavior Research Methods* 39(3):600–605.

Sara Rosenthal, Preslav I. Nakov, Svetlana Kiritchenko, Saif M. Mohammad, Alan Ritter, and Veselin Stoyanov. 2015. SemEval 2015 Task 10: Sentiment analysis in Twitter. In *SemEval 2015 — Proceedings of the 9th International Workshop on Semantic Evaluation @ NAACL-HLT 2015*. Denver, Colorado, USA, June 4-5, 2015, pages 451–463.

Sascha Rothe, Sebastian Ebert, and Hinrich Schütze. 2016. Ultradense word embeddings by orthogonal transformation. In *NAACL-HLT 2016 — Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California, USA, June 12-17, 2016, pages 767–777.

James A. Russell. 1980. A circumplex model of affect. *Journal of Personality and Social Psychology* 39(6):1161–1178.

James A. Russell and Albert Mehrabian. 1977. Evidence for a three-factor theory of emotions. *Journal of Research in Personality* 11(3):273–294.

David S. Schmidtke, Tobias Schröder, Arthur M. Jacobs, and Markus Conrad. 2014. ANGST: Affective Norms for German Sentiment Terms, derived from the Affective Norms for English Words. *Behavior Research Methods* 46(4):1108–1118.

João Sedoc, Daniel Preoţiuc-Pietro, and Lyle H. Ungar. 2017. Predicting emotional word ratings using distributional representations and signed clustering. In *EACL 2017 — Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. Valencia, Spain, April 3-7, 2017, volume 2: Short Papers, pages 564–571.

Hendra Setiawan, Zhongqiang Huang, Jacob Devlin, Thomas Lamar, Rabih Zbib, Richard M. Schwartz, and John Makhoul. 2015. Statistical machine translation features with multitask tensor networks. In *ACL-IJCNLP 2015 — Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics & 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*. Beijing, China, July 26-31, 2015, volume 1: Long Papers, pages 31–41.

Agnes Sianipar, Pieter van Groenestijn, and Ton Dijkstra. 2016. Affective meaning, concreteness, and subjective frequency norms for Indonesian words. *Frontiers in Psychology* 7:#1907.

Ana Paula Soares, Montserrat Comesaña, Ana P Pinheiro, Alberto Simões, and Carla Sofia Frade. 2012. The adaptation of the Affective Norms for English Words (ANEW) for European Portuguese. *Behavior Research Methods* 44(1):256–269.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *ACL 2016 — Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Berlin, Germany, August 7–12, 2016, volume 2: Short Papers, pages 231–235.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15:1929–1958.

Hans Stadthagen-Gonzalez, Constance Imbault, Miguel A. Pérez-Sánchez, and Marc Brysbært. 2017. Norms of valence and arousal for 14,031 Spanish words. *Behavior Research Methods* 49(1):111–123.

Ryan A. Stevenson, Joseph A. Mikels, and Thomas W. James. 2007. Characterization of the affective norms for English words by discrete emotional categories. *Behavior Research Methods* 39(4):1020–1024.

Peter D. Turney and Michael L. Littman. 2003. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems* 21(4):315–346.

Jin Wang, Liang-Chih Yu, K. Robert Lai, and Xuejie Zhang. 2016a. Community-based weighted graph model for valence-arousal prediction of affective words. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24(11):1957–1968.

Jin Wang, Liang-Chih Yu, K. Robert Lai, and Xuejie Zhang. 2016b. Dimensional sentiment analysis using a regional CNN-LSTM model. In *ACL 2016 — Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Berlin, Germany, August 7-12, 2016, volume 2: Short Papers, pages 225–230.

Amy Beth Warriner, Victor Kuperman, and Marc Brysbært. 2013. Norms of valence, arousal, and dominance for 13,915 English lemmas. *Behavior Research Methods* 45(4):1191–1207.

Liang-Chih Yu, Lung-Hao Lee, Shuai Hao, Jin Wang, Yunchao He, Jun Hu, K. Robert Lai, and Xuejie Zhang. 2016a. Building Chinese affective resources in valence-arousal dimensions. In *NAACL-HLT 2016 — Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California, USA, June 12-17, 2016, pages 540–545.

Liang-Chih Yu, Lung-Hao Lee, and Kam-Fai Wong. 2016b. Overview of the IALP 2016 Shared Task on dimensional sentiment analysis for Chinese words. In *IALP 2016 — Proceedings of the [20th] 2016 International Conference on Asian Language Processing*. Tainan, Taiwan, November 21-23, 2016, pages 156–160.

Human Needs Categorization of Affective Events Using Labeled and Unlabeled Data

Haibo Ding
School of Computing
University of Utah
Salt Lake City, UT 84112
hbding@cs.utah.edu

Ellen Riloff
School of Computing
University of Utah
Salt Lake City, UT 84112
riloff@cs.utah.edu

Abstract

We often talk about events that impact us positively or negatively. For example *"I got a job"* is good news, but *"I lost my job"* is bad news. When we discuss an event, we not only understand its affective polarity but also the reason *why* the event is beneficial or detrimental. For example, getting or losing a job has affective polarity primarily because it impacts us financially. Our work aims to categorize affective events based upon *human need* categories that often explain people's motivations and desires: PHYSIOLOGICAL, HEALTH, LEISURE, SOCIAL, FINANCIAL, COGNITION, and FREEDOM. We create classification models based on event expressions as well as models that use contexts surrounding event mentions. We also design a co-training model that learns from unlabeled data by simultaneously training event expression and event context classifiers in an iterative learning process. Our results show that co-training performs well, producing substantially better results than the individual classifiers.

1 Introduction

Recent research has focused on identifying *affective events* in text, which are activities or states that positively or negatively affect the people who experience them. Recognizing affective events in text is challenging because they appear as factual expressions and their affective polarity is often implicit. For example, *"I broke my arm"* and *"I got fired"* are usually negative experiences, while *"I broke a record"* and *"I went to a concert"* are typically positive experiences. Several NLP techniques have been developed to recognize affective events, including patient polarity verb bootstrapping (Goyal et al., 2010, 2013), implicature rules (Deng and Wiebe, 2014), label propagation (Ding and Riloff, 2016), pattern-based learning (Vu et al., 2014; Reed et al., 2017), and semantic consistency optimization (Ding and Riloff, 2018).

Our research aims to probe deeper and understand not just the polarity of affective events, but *the reason for* the polarity. Events can impact people in many ways, and understanding *why* an event is beneficial or detrimental is a fundamental aspect of language understanding and narrative text comprehension. Additionally, many applications could benefit from understanding the nature of affective events, including text summarization, conversational dialogue processing, and mental health therapy or counseling systems. As an illustration, a mental health therapy system can benefit from understanding why someone is in a negative state. If the triggering event for depression is *"I broke my leg"* then the reason is about the person's Health, but if the triggering event is *"I broke up with my girlfriend"* then the reason is based on Social relationships.

We hypothesize that the polarity of affective events can often be attributed to a relatively small set of *human need* categories. Our work is motivated by theories in psychology that explain people's motivations, desires, and overall well-being in terms of categories associated with basic human needs, such as Maslow's Hierarchy of Needs (Maslow et al., 1970) and Fundamental Human Needs (Max-Neef et al., 1991). Drawing upon these works, we propose that the polarity of affective events often arises from 7 types of human needs: PHYSIOLOGICAL, HEALTH, LEISURE, SOCIAL, FINANCIAL, COGNITION, and FREEDOM. For example, *"I broke my arm"* has negative polarity because it negatively impacts one's Health, *"I got fired"* is negative because it negatively impacts one's Finances, and *"I am confused"* is negative because it reflects a problem related to Cognition.

We explore this hypothesis and tackle the chal-

Proceedings of NAACL-HLT 2018, pages 1919–1929
New Orleans, Louisiana, June 1 - 6, 2018.

lenge of categorizing affective events in text with respect to these 7 human need categories. As our evaluation data, we use events extracted from personal blog posts and manually labeled with affective polarity in previous work (Ding and Riloff, 2018). These affective events were then subsequently annotated for the human need categories.

In this paper, we design several types of classification models that learn from both labeled and unlabeled data. First, we present supervised learning models that use lexical and embedding features for the words in event expressions, as well as models that learn from the sentence contexts surrounding mentions of event expressions. Next, we explore self-training and co-training models that exploit both labeled and unlabeled data for training. The most effective system is a co-training model that uses two classifiers with two different views in an iterative learning process: one classifier only uses the words in an event expression, and the other classifier only uses the contexts surrounding instances of an event expression. Our results show that this co-training model effectively uses unlabeled data to substantially improve results compared to classifiers trained only with labeled data, yielding gains in both precision and recall.

2 Related Work

Recently, there has been growing interest in recognizing the affective polarity of events. For example, Goyal et al. (2013) developed a bootstrapped learning method to learn *patient polarity verbs*, which impart affective polarities to their patients. Li et al. (2015) designed methods to extract verb expressions that imply negative opinions from reviews. Rashkin et al. (2016) recently proposed connotation frames to incorporate the connotative polarities for a verb's arguments from the writer's and other event entities' perspectives. Li et al. (2014) proposed a bootstrapping approach to extract major life events from tweets using congratulation and condolence speech acts. Most of these major life events are affective although their work did not identify polarity. Another group of researchers have studied +/- effect events (Deng et al., 2013; Choi and Wiebe, 2014) which they previously called benefactive/malefactive events. Their work mainly focused on inferring implicit opinions through implicature rules (Deng and Wiebe, 2014, 2015).

Ding and Riloff (2016) designed an event context graph model to identify affective events using label propagation. Reed et al. (2017) demonstrated that automatically acquired patterns could benefit the recognition of first-person related affective sentences. Most recently, Ding and Riloff (2018) developed a semantic consistency model to induce a large set of affective events using three types of semantic relations in an optimization framework. (We use their annotated affective event data set in our work.) All of this previous work only identifies affective events and their polarities. In contrast, our work aims to identify the reason for the affective polarity of an event.

The human need categories are inspired by two prior theories. The first one is Maslow's Hierarchy of Needs (Maslow et al., 1970) which was developed to study people's motivations and personalities. The second one is Fundamental Human Needs (Max-Neef et al., 1991) which was developed to help communities identify their strengths and weaknesses. The human need categories are also related to the concept of "goals", which has been proposed by (Schank and Abelson, 1977) to understand narrative stories. Goals could be very specific to a character in a particular narrative story. However, but many types of goals originate from universal needs and desires shared by most people (Max-Neef et al., 1991). In addition, our work is also related to research on wish detection (Goldberg et al., 2009), desire fulfillment (Chaturvedi et al., 2016), and modelling protagonist goals and desires (Rahimtoroghi et al., 2017).

Self-training is a semi-supervised learning method to improve performance by exploiting unlabeled data. Self-training has been successfully used in many NLP applications such as information extraction (Ding and Riloff, 2015) and syntactic parsing (McClosky et al., 2006). Co-training (Blum and Mitchell, 1998) uses both labeled and unlabeled data to train models that have two different views of the data. Co-training has been previously used for many NLP tasks including spectral clustering (Kumar and Daumé, 2011), word sense disambiguation (Mihalcea, 2004), coreference resolution (Phillips and Riloff, 2002), and sentiment analysis (Wan, 2009; Xia et al., 2015).

3 Affective Event Data

The goal of our research is to categorize affective events based on 7 categories of human needs. To facilitate this work, we build upon a large data set

Physiological	Health	Leisure	Social	Finance	Cognition	Freedom	Emotion	None
19 (4%)	52 (10%)	75 (14%)	108 (20%)	29 (5%)	26 (5%)	7 (1%)	128 (24%)	98 (18%)

Table 1: Distribution of Human Need Categories (each cell shows the frequency and percentage).

created for prior research (Ding and Riloff, 2018) which aims to identify affective events. We will refer to this data as the AffectEvent dataset. We will briefly describe this data and the human need category annotations that we added on top of it.

The AffectEvent dataset contains events extracted from a personal story corpus that was created by applying a personal story classifier (Gordon and Swanson, 2009) to 177 million blog posts. The personal story corpus contains 1,383,425 personal story blogs. StanfordCoreNLP (Manning et al., 2014) was used for POS and NER tagging and SyntaxNet (Andor et al., 2016) for parsing. Each event is represented using a frame-like structure to capture the meanings of different types of events. Each event representation contains four components: ⟨**Agent**, **Predicate**, **Theme**, **PP**⟩. The Predicate is a simple verb phrase corresponding to an action or state. The Agent is a named entity, nominal, or pronoun, and is extracted using syntactic heuristics rather than semantic role labeling. We use "Theme" loosely to allow a NP or adjective to fill this role. The PP component is composed of a preposition and a NP. All words in the event are lemmatized, and active and passive voices are normalized to have the same representation. See (Ding and Riloff, 2018) for more details of the event representation. Table 2 shows some examples of extracted events.

Positive Events	**Human Need**
⟨ our pizza; arrive, -, -⟩	Physiological
⟨ ear, be, better, - ⟩	Health
⟨ I, watch, Hellboy II, -⟩	Leisure
⟨ we, get, marry, -⟩	Social
⟨ I, get, my new laptop, -⟩	Finance
⟨ my memory, be, vivid, -⟩	Cognition
⟨ my heart, feel, happy, -⟩	Emotion
⟨ we, be, legal, -⟩	None
Negative Events	**Human Need**
⟨ I, grow, hungry, - ⟩	Physiological
⟨ my face, look, pale, - ⟩	Health
⟨ -, rain out, game, -⟩	Leisure
⟨ you, confront, me, -⟩	Social
⟨ I, be, unemployed, at time ⟩	Finance
⟨ my memory, not serve, me, -⟩	Cognition
⟨ I, be, scared, -⟩	Emotion
⟨ it, not work, -, for me⟩	None

Table 2: Examples of Affective Events with Human Need Category Labels

3.1 Human Need Category Annotations

Affective events impact people in a positive or negative way for a variety of reasons. We hypothesized that the polarity of most affective events arises from the satisfaction or violation of basic human needs. Psychologists have developed theories that explain people's motivations, desires, and overall well-being in terms of categories associated with basic human needs, such as Maslow's Hierarchy of Needs (Maslow et al., 1970) and Fundamental Human Needs (Max-Neef et al., 1991). Based upon this work, we defined 7 human need categories, which are briefly described below.

Physiological Needs maintain our body's basic functions (e.g., air, food, water, sleep). *Health Needs* are to be physically healthy and safe. *Leisure Needs* are to have fun, to be relaxed, to have leisure time, to appreciate and enjoy beauty. *Social Needs* are to have good social relations (e.g., family, friendship), to have good self-worth and self-esteem, and to be respected by others. *Financial Needs* are to obtain and protect financial income, to acquire and maintain valuable possessions, to have a job and satisfying work. *Cognition Needs* are to obtain skills, information, and knowledge, to receive education, to improve one's intelligence, and to mentally process information correctly. *Freedom Needs* are the ability to move or change positions freely, and to access things or services in a timely manner. We also defined two categories for event expressions that represent explicit emotions and opinions (*Emotions/Sentiments/Opinions*) and events that do not fall into any other categories (*None of the Above*).

We added manual annotations for human need categories on top of the manually annotated positive and negative affective events in the AffectEvent dataset. Three people were asked to assign a human need category label to each of the 559 affective events in the AffectEvent test set. Annotators achieved good pairwise inter-annotator agreement ($\kappa \geq .65$) on this task. The Cohen's kappa scores were κ=.69, κ=.66 and κ=.65. We assigned a single category to each event because most of the affective events fell into just one category in our preliminary study, even though some cases could legitimately be argued

for multiple categories. We discuss this issue further in Section 5.4

The distribution of human need categories is shown in Table 1. Since very few affective events were found to belong to the *Freedom* category, this category was merged into None. Additionally, 17 events received three different labels from the annotators, so they were discarded. The majority label was then assigned to the remaining events, yielding a gold standard data set of 542 affective events with human need category labels. Some of the annotated examples are shown in Table 2. A more detailed description of the human need category definitions, data set, and manual annotation effort is described in (Ding et al., 2018). This data set is freely available for other researchers to use.

In the next section, we present classification models designed to tackle this human needs categorization task.

4 Categorizing Human Needs with Labeled and Unlabeled Data

Automatically categorizing affective events in text based on human needs is a new task, so we investigated several types of approaches. First, we designed supervised classifiers to categorize affective events based upon the words in the event expressions, which we will refer to as *Event Expression Classifiers*. We explored lexical features, word embedding features, and semantic category features, along with several types of machine learning algorithms.

Our task is to determine the human need category of an affective event based on the meaning of the event itself, independent of any specific context.[1] But we hypothesized that collecting the contexts around instances of the events could also provide valuable information to infer human need categories. So we also designed *Event Context Classifiers* to use the sentence contexts around event mentions as features.

Our gold standard data set is relatively small, so supervised learning that relies entirely on manually labeled data may not have sufficient coverage to perform well across the human need categories. However, the AffectEvent dataset contains a very large set of events that were extracted from the same blog corpus, but not manually labeled with affective polarity. Consequently, we explored two weakly supervised learning methods to exploit this large set of unlabeled events. First, we tried self-training to iteratively improve the event expression classifier. Second, we designed a co-training model that takes advantage of both an event expression classifier and an event context classifier to learn from the unlabeled events. These two types of classifiers provide complementary views of an event, so new instances labeled by one classifier can be used as valuable new data to benefit the other classifier, in an iterative learning cycle.

4.1 Event Expression Classifiers

The most obvious approach is to use the words in event expressions as features for recognizing human need categories (e.g., {ear, be, better} for the event <ear, be, better>). We experimented with both lexical (string) features and pre-trained word embedding features. For the latter, we used GloVe (Pennington et al., 2014) vectors (200d) pretrained on 27B tweets. For each event expression, we compute its embedding as the average of its words' embeddings.

We also designed semantic features using the lexical categories in the LIWC lexicon (Pennebaker et al., 2007) to capture a more general meaning for each word. LIWC is a dictionary of words associated with "psychologically meaningful" lexical categories, some of which are directly relevant to our task, such as AFFECTIVE, SOCIAL, COGNITIVE, and BIOLOGICAL PROCESS. We identify the LIWC category of the head word of each phrase in the event representation and use them as *Semantic Category* features.

We experimented with three types of supervised classification models: logistic regression (LR), support vector machines (SVM), and recurrent neural network classifiers (RNN). One advantage of the RNN is that it considers the word order in the event expression, which can be important. In our experiments, we used the Scikit-learn implementation (Pedregosa et al., 2011) for the LR classifier, and LIBSVM (Chang and Lin, 2011) with a linear kernel for the SVM classifier. For the RNN, we used the example LSTM implementation from Keras (Chollet et al., 2015) github, which was developed to build a sentiment classifier. We used the default parameters in our experiments[2].

[1] We view this as assuming the most common interpretation of an event, which would be the default in the absence of context.

[2] LR and SVM use the one-vs-rest (ovr) scheme, while RNN is a single multi-class classifier.

4.2 Event Context Classifiers

The event dataset was originally extracted from a large collection of blog posts, which contain many instances of the events in different sentences. We hypothesized that the contexts surrounding instances of an event can also provide strong clues about the human need category associated with the event. Therefore, we also created *Event Context Classifiers* to exploit the sentence contexts around event mentions. We explored several designs for event context classifiers, which are explained below.

Context$^{\text{SentBOW}}$: For each event in the training set, we first collect all sentences mentioning this event and assign the event's human need category as the label for each sentence. Each sentence is then used as a training instance for the event context classifier. We use a bag-of-words representation for each sentence.

Context$^{\text{SentEmbed}}$: This variation labels sentences exactly the same way as the previous model. But each sentence is represented as a dense embedding vector, which is computed as the average of the embeddings for each word in the sentence. We used GloVe (Pennington et al., 2014) vectors (200d) pretrained on 27B tweets.

Context$^{\text{AllBOW}}$: Instead of treating each sentence as a training instance, for this model we aggregate all of the sentences that mention the same event to create one giant context for the event. Each event corresponds to one training instance in this model, which is represented using bag-of-word features.

Context$^{\text{AllEmbed}}$: This variation aggregates the sentences that mention an event exactly like the previous model. But each sentence is represented as a dense embedding vector. First, we compute an embedding vector for each sentence as the average of the embeddings of its words. Then we compute a single context embedding by averaging all of the sentence embeddings.

In the data, some events appear in many sentences, while others appear in just a few sentences. To maintain balance, we randomly sample 10 sentences for each event to use as its contexts.

To predict the human need category of an event, we first apply the event context classifier to contexts that mention the event, which produces a probability distribution over the human need categories. For each category, we compute its mean probability. Finally, we assign the event with the human need category that has the highest mean probability (i.e. argmax).

4.3 Self-Training the Event Expression Classifier

Our labeled data set is relatively small, but as mentioned previously, the AffectEvent dataset contains a large set of unlabeled events as well. So we designed a self-training model to try to iteratively improve the event expression classifier by exploiting the unlabeled event data.

The self-training process works as follows. Initially, the event expression classifier is trained using the manually labeled events. Then the classifier is applied to the unlabeled events and assigns a human need category to each event with a confidence value. For each human need category, we select the unlabeled event that has been assigned to that category with the highest confidence. Therefore, each category will have one additional labeled event at each iteration. The newly labeled events are added to the labeled data set, and the classifier is re-trained for the next iteration.

4.4 Co-Training with Event Expression and Event Context Classifiers

The sentence contexts in which an event appears contain complementary information to the event expression itself. So we designed co-training models to exploit these complementary types of classifiers to iteratively learn from unlabeled data.

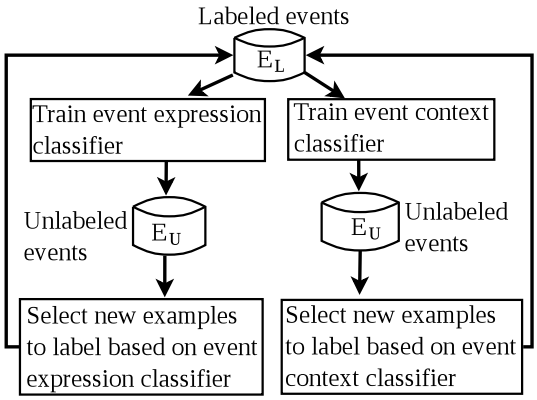


Figure 1: The Co-Training Model

Figure 1 shows the architecture of our co-training model. Initially, an event expression classifier and an event context classifier are independently trained on the manually labeled training data. Each classifier is then applied to the large collection of unlabeled events E_U. For each hu-

man need category, we then select the event that has been assigned to the category with the highest confidence value as a new instance to label. Consequently, each category will receive two additional labeled events at each iteration, one from the event expression classifier and another one from the event context classifier.[3] Both sets of newly labeled events are then added to the labeled set E_L, and each of the classifiers is re-trained on the expanded set of labeled data. Because the classifiers have different views of the events, the new instances labeled by one classifier serve as fresh training instances for the other, unlike self-training with a single classifier where it is learning entirely from its own predictions. The following section describes the co-training algorithm in more detail.

4.4.1 The Co-Training Algorithm

Our co-training algorithm is shown in Algorithm 1. The input to the algorithm are the sets of labeled events E_L and unlabeled events E_U. Each event is associated with both an event expression and the set of sentences in which it occurs in the blogs corpus.

For each iteration, the event expression classifier is first trained using the labeled events E_L with the event expression view. Then, we construct an event context view X_{con} for each event in the labeled set E_L. The context sentences are used differently depending on the type of context model (described in Section 4.2). An event context classifier is then trained using the context view X_{con}. Both classifiers are then independently applied to the unlabeled events E_U. For each human need category, each classifier selects one event to label based on its most confident prediction. All of the newly labeled events are then added to the labeled training set E_L, and the process repeats.

4.4.2 Prediction with Co-Trained Classifiers

The co-training process simultaneously trains two classifiers, so here we explain how we use the resulting classifiers after the co-training process has finished. For each event e in the test set, we apply both the event expression classifier and the event context classifier, which each produce a probability distribution over the human need categories. Then we explore two different methods to combine the two probability distributions for each test event: (1) **sum**, we compute the final probability vector $\boldsymbol{p}(e)$ by applying the element-wise summarization operation to the two predicted probability vectors; (2) **product**, we compute the final $\boldsymbol{p}(e)$ as the element-wise product of the two vectors. Then, the final probability vector is normalized to make sure the sum of probabilities over all classes is 1. Finally, we predict an event's human need category as the one with the highest probability.

[3]The event expression classifier first selects from unlabeled events, then the event context classifier does the selection. This ensures that there are 16 new events in total at each iteration.

Algorithm 1 Co-Training Algorithm

Input: Labeled E_L, unlabeled E_U events
while Not maximum iteration **do**
  Train the event expression classifier on E_L
  Construct context view (X_{con}) of E_L
  Train the event context classifier on X_{con}
  Apply the event expression classifier to E_U and select new labeled events (E_{exp})
  Apply the event context classifier to E_U and select new labeled events (E_{con})
  Update labeled events:
$$E_L = E_L \cup E_{exp} \cup E_{con}$$
end while

5 Evaluation

We conducted experiments to evaluate the methods described in Section 4. For all of our experiments, the results are reported based on 3-fold cross-validation on the 542 affective events manually labeled with human need categories. We show the average results over 3-folds in the following sections. For development, we used a distinct set of events labeled during preliminary studies. We did not tune any of the models, using only their default parameter settings. We present experimental results in terms of precision, recall, and F1 score, macro-averaged over the human need categories.

5.1 Performance of Event Expression Classifiers

Table 4 shows the results[4] for the event expression classifiers. We also evaluated the ability of the LIWC lexicon (Pennebaker et al., 2007) to label the event expressions. We manually aligned the relevant LIWC categories with our human need categories, as shown in Table 3. Then we labeled each event by identifying the human need category of each word in the event phrase and assign-

[4]Since we report the average precision, recall, F1 score over 3-folds, the F1 score can be smaller than both precision and recall in some cases.

ing the most frequent category to the event.[5] If no words were assigned to our categories, we labeled the event as None. The top row of Table 4 shows that LIWC achieved 39% recall but only 47.7% precision. The reason is that some categories in LIWC are more generalized compared with the definitions of our corresponding categories. For example, the words "abandon" and "damage" belong to the Affect category (corresponding to our Emotion category) in LIWC. However, based on our definition the event "my house was damaged" actually belongs to the Finance category. In this way, the Emotion category is overly generalized which leads to low precision for this class.

LIWC Category		**Human Need Category**
Ingest	→	Physiological
Health, Body, Death	→	Health
Leisure	→	Leisure
Social	→	Social
Money, Work	→	Finance
Inhib, Insight	→	Cognition
Affect	→	Emotion

Table 3: LIWC Mapping to Human Need Categories.

The LR and SVM rows in Table 4 show the performance of the logistic regression (LR) and support vector machine (SVM) classifiers, respectively. We evaluated classifiers with bag-of-words features (BOW) and classifiers with event embedding features (Embed), computed as the average of the embeddings for all words in the event expression. We also tried adding semantic category features from LIWC to each feature set, denoted as +SemCat. The results show that the Embed features performed best for both the LR and SVM classifiers. Adding the SemCat features improved upon the bag-of-word representations, but not the embeddings.

The last two rows of Table 4 show the performance of two RNN classifiers, one using lexical words as input ($\text{RNN}^{\text{Words}}$) and one using pre-trained word embeddings as input ($\text{RNN}^{\text{EmbedSeq}}$). The $\text{RNN}^{\text{EmbedSeq}}$ system takes the sequence of word embeddings as input rather than the average embeddings. As with the other classifiers, the word embedding feature representations performed best, achieving an F1 score 54.4%, which is comparable to the F1 score of the LR^{Embed} system. However, the RNN's precision was only 58%, compared to 64.2% for the logistic regression model, with only 2% higher recall that does not fully compensate for the lower precision. Neural net models often need large training sets, so the relatively small size of our training data may not be ideal for an RNN.

[5]For ties, we remove a component one by one in the order of Agent, PP, Theme until we obtain a majority label.

Method	Precision	Recall	F1
LIWC	47.7	39.0	38.6
LR^{BOW}	33.6	28.7	27.3
$\text{LR}^{\text{BOW+SemCat}}$	55.2	39.6	41.9
$\text{LR}^{\text{Embed+SemCat}}$	60.1	49.3	51.9
LR^{Embed}	**64.2**	51.7	**54.8**
SVM^{BOW}	52.3	43.1	44.8
$\text{SVM}^{\text{BOW+SemCat}}$	51.0	45.9	46.8
$\text{SVM}^{\text{Embed+SemCat}}$	50.4	48.4	48.6
$\text{SVM}^{\text{Embed}}$	51.3	50.7	50.5
$\text{RNN}^{\text{Words}}$	45.2	39.6	40.1
$\text{RNN}^{\text{EmbedSeq}}$	58.0	**53.7**	54.4

Table 4: Performance of Event Expression Classifiers

Overall, we concluded that the logistic regression classifier with event embedding features (LR^{Embed}) achieved the best performance because of its F1 score (54.8%) and higher precision (64.2%).

5.2 Performance of Event Context Classifiers

Table 5 shows the performance[4] of the event context classifiers described in Section 4.2. Since logistic regression worked best in the previous experiments, we only evaluated logistic regression classifiers in our remaining experiments. The results show that using each context sentence as an individual training instance ($\text{Context}^{\text{SentBOW}}$ and $\text{Context}^{\text{SentEmbed}}$) substantially outperformed the classifiers that merged all the context sentences as a single training instance ($\text{Context}^{\text{AllBOW}}$ and $\text{Context}^{\text{AllEmbed}}$). Overall, the best performing system $\text{Context}^{\text{SentEmbed}}$ achieved an F1 score of 44.3% with 59.1% Precision.

Method	Precision	Recall	F1
$\text{Context}^{\text{AllBOW}}$	20.6	18.0	17.8
$\text{Context}^{\text{AllEmbed}}$	38.2	29.9	29.1
$\text{Context}^{\text{SentBOW}}$	48.2	31.4	32.8
$\text{Context}^{\text{SentEmbed}}$	**59.1**	**41.9**	**44.3**

Table 5: Performance of Event Context Classifiers

It is worth noting that the precision of the best contextual classifier was only 5% below that of the best event expression classifier, while there was a 10% difference in their recall. Since they achieved (roughly) similar levels of precision and represent complementary views of events, a co-training

framework seemed like a logical way to use them together to gain additional benefits from unlabeled event data.

We also created a classifier that combined event expression features and event context features together. But combining them did not improve performance.

5.3 Performance of Self-Training and Co-Training Models

In this section, we evaluate the weakly supervised self-training and co-training methods that additionally use unlabeled data. To keep the number of unlabeled events manageable, we only used events in the AffectEvent dataset that had frequency ≥ 100, which produced an unlabeled data set of 23,866 events.

We used the best performing event expression classifier (LR^{Embed}) in these models, and the co-training framework includes the best performing event context classifier ($Context^{SentEmbed}$) as well. We also experimented with the **sum** and **product** variants for co-training (described in Section 4.4.2), which are denoted as $CoTrain^{sum}$ and $CoTrain^{prod}$. We ran both the self-training and co-training methods for 20 iterations.

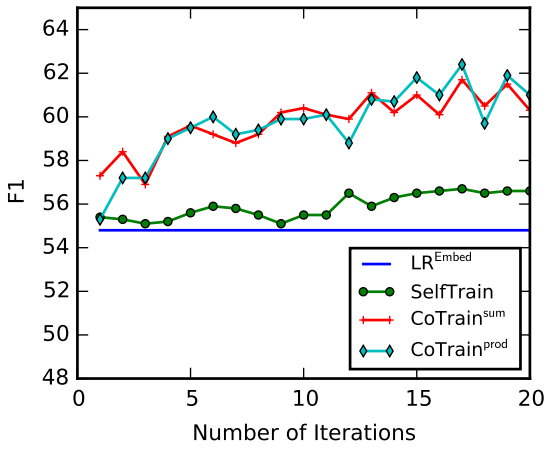


Figure 2: Learning Curves

Figure 2 tracks the performance of the self-training and co-training models after each iteration, in terms of F1 score. The flat line shows the F1 score for the best classifier that uses only labeled data (LR^{Embed}). Both types of models yield performance gains from iteratively learning with the unlabeled data, but the co-training models perform substantially better than the self-training model. Even after just 5 iterations, co-training achieves an F1 score over 58%, and by 20 iterations performance improves to $> 60\%$.

Table 6 shows the results for these models after 20 iterations, which was an arbitrary stopping criterion, and after 17 iterations, which happened to produce the best results for all three systems. The first two rows show the results of the best performing event context classifier ($Context^{SentEmbed}$) and best performing event expression classifier (LR^{Embed}) from the previous experiments, for the sake of comparison.

Table 6 shows that after 20 iterations, the $CoTrain^{prod}$ model performed best, yielding an F1 score of 61% compared to 54.8% for the LR^{Embed} model. Furthermore, we see gains in both recall and precision.

All three systems performed best after 17 iterations, so we show those results as well to give an idea of additional gains that would be possible if we could find an optimal stopping criterion. Our data set was small so we did not feel that we had enough data to fine-tune parameters, but we see the potential to further improve performance given additional tuning data.

Method	Precision	Recall	F1
Supervised Models			
$Context^{SentEmbed}$	59.1	41.9	44.3
LR^{Embed}	64.2	51.7	54.8
After 20 Iterations			
SelfTrain	63.2	54.2	56.6
$CoTrain^{sum}$	66.2	58.2	60.3
$CoTrain^{prod}$	**67.1**	**58.7**	**61.0**
Best Results, After 17 Iterations			
SelfTrain	63.5	54.1	56.7
$CoTrain^{sum}$	68.6	59.0	61.7
$CoTrain^{prod}$	**69.7**	**59.5**	**62.4**

Table 6: Performance of Self-Training and Co-Training

Table 7 shows a breakdown of the performance across the individual human need categories for two models: the best event expression classifier and the best co-training model ($CoTrain^{prod}$ after 17 iterations). We see that the co-training model outperformed the LR^{Embed} model on every category. Co-training improved performance the most for the Finance and Cognition categories, yielding F1 score gains of +12% and +16%, respectively, and notably improving both recall and precision.

5.4 Analysis

We manually examined our system's predictions to better understand its behavior. We found that most of the correctly classified Physiological

Category	LR^{Embed} Pre	Rec	F1	CoTrain^{Prod} Pre	Rec	F1
Physiological	82	57	67	81	68	**74**
Health	65	40	49	68	50	**57**
Leisure	62	59	60	69	63	**66**
Social	61	72	66	68	79	**73**
Finance	61	31	40	67	44	**52**
Cognition	75	31	42	92	46	**58**
Emotion	60	75	66	64	74	**69**
None	47	49	48	48	52	**50**

Table 7: Breakdown of results across Human Need categories. Each cell shows Precision, Recall, and F1.

events were related to food, while the correctly classified Cognition events were primarily about learning and understanding. Our method missed many events for the Health, Finance, and Cognition classes. For Health, many medical symptoms were not recognized, such as *"my face looks pale"* and *"I puked"*. For Finance, the system missed events related to possessions (e.g., *"engine stopped running"* and *"my clock is wrong"*) and jobs (e.g., *"I went to resign"*).

We also took a closer took at which categories were confused with other categories. Figure 3 shows the confusion matrix between CoTrain^{Prod} and the gold annotations. Each cell shows the total number of confusions across the 3-folds of cross-validation. The category names are abbreviated as Physiological (Phy), Health (Hlth), Leisure (Leis), Social (Socl), Finance (Fnc), Cognition (Cog), and Emotion (Emo). #Tot denotes the total number of events in each row or column.

Pred. \ Gold	Phy	Hlth	Leis	Socl	Fnc	Cog	Emo	None	#Tot
Phy	13	1	0	0	1	0	0	2	17
Hlth	1	26	1	0	1	1	4	8	42
Leis	1	1	48	4	0	1	4	10	69
Socl	0	6	4	84	2	3	10	11	120
Fnc	1	0	2	0	13	0	1	5	22
Cog	0	0	0	0	0	12	1	2	15
Emo	1	5	12	12	3	1	91	16	141
None	2	13	8	8	9	8	17	51	116
#Tot	19	52	75	108	29	26	128	105	542

Figure 3: Confusion between Predictions and Gold.

The co-training model had difficulty distinguishing the None category from other classes, presumably because None does not have its own semantics but is used for affective events that do not belong to any of the other categories. We also see that the system often confuses Emotion with Leisure and Social. This happens because many event expressions contain words that refer to emotions. Our guidelines instructed annotators to focus on the event and assign the Emotion label only when no event is described beyond an emotion (e.g., *"I was thrilled"*). Consequently, the gold label of *"I love journey"* is Leisure and *"I'm worried about my mom"* is Social, but both were classified by the system as Emotion. In future work, it may be advantageous to allow event expressions to be labeled as both an explicit Emotion and a Human Need category based on the target of the emotion.

6 Conclusions

In this work, we introduced a new challenge to recognize the reason for the affective polarity of events in terms of basic human needs. We designed four types of classification methods to categorize affective events according to human need categories, exploiting both labeled and unlabeled data. We first evaluated event expression and event context classifiers, trained using only labeled data. Then we designed self-training and co-training methods to additionally exploit unlabeled data. A co-training model that simultaneously trains event expression and event context classifiers produced substantial performance gains over the individual models. However, performance on the human need categories still has substantial room for improvement. In future work, obtaining more human annotations will be useful to build a better human needs categorization system. In addition, applying and analyzing the human needs of affective events in narrative stories and conversations is a fruitful and interesting direction for future research.

7 Acknowledgements

This material is based in part upon work supported by the National Science Foundation under Grant Number IIS-1619394. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. We are very grateful to Tianyu Jiang and Yuanyuan Gao for participating in the manual annotation effort.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally Normalized Transition-Based Neural Networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Avrim Blum and Tom Mitchell. 1998. Combining Labeled and Unlabeled Data with Co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*. ACM.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A Library for Support Sector Machines. *ACM Transactions on Intelligent Systems and Technology (TIST)* 2(3):27.

Snigdha Chaturvedi, Dan Goldwasser, and Hal Daumé III. 2016. Ask, and Shall You Receive? Understanding Desire Fulfillment in Natural Language Text. In *Processings of the 30th AAAI Conference on Artificial Intelligence*.

Yoonjung Choi and Janyce Wiebe. 2014. +/-EffectWordNet: Sense-level Lexicon Acquisition for Opinion Inference. In *Proceedings of the 2014 Conference on Empirical Methods on Natural Language Processing*.

François Chollet et al. 2015. Keras. https://keras.io.

Lingjia Deng, Yoonjung Choi, and Janyce Wiebe. 2013. Benefactive/Malefactive Event and Writer Attitude Annotation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.

Lingjia Deng and Janyce Wiebe. 2014. Sentiment Propagation via Implicature Constraints. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*.

Lingjia Deng and Janyce Wiebe. 2015. Joint Prediction for Entity/Event-Level Sentiment Analysis using Probabilistic Soft Logic Models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Haibo Ding, Tianyu Jiang, and Ellen Riloff. 2018. Why is an Event Affective? Classifying Affective Events based on Human Needs. In *AAAI-18 Workshop on Affective Content Analysis*.

Haibo Ding and Ellen Riloff. 2015. Extracting Information about Medication Use from Veterinary Discussions. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Haibo Ding and Ellen Riloff. 2016. Acquiring Knowledge of Affective Events from Blogs Using Label Propagation. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*.

Haibo Ding and Ellen Riloff. 2018. Weakly Supervised Induction of Affective Events by Optimizing Semantic Consistency. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Andrew B Goldberg, Nathanael Fillmore, David Andrzejewski, Zhiting Xu, Bryan Gibson, and Xiaojin Zhu. 2009. May All Your Wishes Come True: A Study of Wishes and How to Recognize Them. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Andrew Gordon and Reid Swanson. 2009. Identifying Personal Stories in Millions of Weblog Entries. In *Third International Conference on Weblogs and Social Media, Data Challenge Workshop*.

A. Goyal, E. Riloff, and H. Daumé III. 2010. Automatically Producing Plot Unit Representations for Narrative Text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*.

A. Goyal, E. Riloff, and H. Daumé III. 2013. A Computational Model for Plot Units. *Computational Intelligence* 29(3):466–488.

Abhishek Kumar and Hal Daumé. 2011. A Co-training Approach for Multi-view Spectral Clustering. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*.

Huayi Li, Arjun Mukherjee, Jianfeng Si, and Bing Liu. 2015. Extracting Verb Expressions Implying Negative Opinions. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Jiwei Li, Alan Ritter, Claire Cardie, and Eduard Hovy. 2014. Major Life Event Extraction from Twitter based on Congratulations/Condolences Speech Acts. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics*.

Abraham Harold Maslow, Robert Frager, James Fadiman, Cynthia McReynolds, and Ruth Cox. 1970. *Motivation and Personality*, volume 2. Harper & Row New York.

Manfred Max-Neef, Antonio Elizalde, and Martin Hopenhayn. 1991. *Human Scale Development: Conception, Application and Further Reflections*. The Apex Press.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective Self-training for Parsing. In *Proceedings of the Conference on Human Language Technology and North American Chapter of the Association for Computational Linguistics*.

Rada Mihalcea. 2004. Co-training and Self-training for Word Sense Disambiguation. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

James W Pennebaker, Roger J Booth, and Martha E Francis. 2007. Linguistic Inquiry and Word Count: LIWC2007. *Austin, TX: liwc.net* .

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

William Phillips and Ellen Riloff. 2002. Exploiting Strong Syntactic Heuristics and Co-Training to Learn Semantic Lexicons. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*.

Elahe Rahimtoroghi, Jiaqi Wu, Ruimin Wang, Pranav Anand, and Marilyn A Walker. 2017. Modelling Protagonist Goals and Desires in First-Person Narrative. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*.

Hannah Rashkin, Sameer Singh, and Yejin Choi. 2016. Connotation Frames: A Data-Driven Investigation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Lena Reed, Jiaqi Wu, Shereen Oraby, Pranav Anand, and Marilyn A. Walker. 2017. Learning Lexico-Functional Patterns for First-Person Affect. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Roger C. Schank and Robert P. Abelson. 1977. *Scripts, Plans, Goals and Understanding: An Inquiry into Human Knowledge Structures*. L. Erlbaum, Hillsdale, NJ.

Hoa Trong Vu, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2014. Acquiring a Dictionary of Emotion-Provoking Events. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*.

Xiaojun Wan. 2009. Co-training for Cross-lingual Sentiment Classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*.

Rui Xia, Cheng Wang, Xin-Yu Dai, and Tao Li. 2015. Co-training for Semi-supervised Sentiment Classification Based on Dual-view Bags-of-words Representation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*.

The Argument Reasoning Comprehension Task: Identification and Reconstruction of Implicit Warrants

Ivan Habernal[†] **Henning Wachsmuth**[‡] **Iryna Gurevych**[†] **Benno Stein**[‡]

[†] Ubiquitous Knowledge Processing Lab (UKP) and Research Training Group AIPHES
Department of Computer Science, Technische Universität Darmstadt, Germany
www.ukp.tu-darmstadt.de www.aiphes.tu-darmstadt.de

[‡] Faculty of Media, Bauhaus-Universität Weimar, Germany
<firstname>.<lastname>@uni-weimar.de

Abstract

Reasoning is a crucial part of natural language argumentation. To comprehend an argument, one must analyze its *warrant*, which explains why its claim follows from its premises. As arguments are highly contextualized, warrants are usually presupposed and left implicit. Thus, the comprehension does not only require language understanding and logic skills, but also depends on common sense. In this paper we develop a methodology for reconstructing warrants systematically. We operationalize it in a scalable crowdsourcing process, resulting in a freely licensed dataset with warrants for 2k authentic arguments from news comments.[1] On this basis, we present a new challenging task, the *argument reasoning comprehension task*. Given an argument with a claim and a premise, the goal is to choose the correct implicit warrant from two options. Both warrants are plausible and lexically close, but lead to contradicting claims. A solution to this task will define a substantial step towards automatic warrant reconstruction. However, experiments with several neural attention and language models reveal that current approaches do not suffice.

1 Introduction

Most house cats face enemies. Russia has the opposite objectives of the US. There is much innovation in 3-d printing and it is sustainable.

What do the three propositions have in common? They were never uttered but solely presupposed in arguments made by the participants of online discussions. Presuppositions are a fundamental pragmatic instrument of natural language argumentation in which parts of arguments are left unstated. This phenomenon is also referred to as common knowledge (Macagno and Walton, 2014, p. 218), enthymemes (Walton, 2007b, p. 12), tacit major premises (Amossy, 2009, p. 319), or implicit *warrants* (Newman and Marshall, 1991, p. 8). Wilson and Sperber (2004) suggest that, when we comprehend arguments, we reconstruct their warrants driven by the cognitive principle of relevance. In other words, we go straight for the interpretation that seems most relevant and logical within the given context (Hobbs et al., 1993). Although any incomplete argument can be completed in different ways (Plumer, 2016), it is assumed that certain knowledge is shared between the arguing parties (Macagno and Walton, 2014, p. 180).

Filling the gap between the claim and premises (aka reasons) of a natural language argument empirically remains an open issue, due to the inherent difficulty of reconstructing the world knowledge and reasoning patterns in arguments. In a direct fashion, Boltužić and Šnajder (2016) let annotators write down implicit warrants, but they concluded only with a preliminary analysis due to large variance in the responses. In an indirect fashion, implicit warrants correspond to major premises in argumentation schemes; a concept heavily referenced in argumentation theory (Walton, 2012). However, mapping schemes to real-world arguments has turned out difficult even for the author himself.

Our main hypothesis is that, even if there is no limit to the tacit length of the reasoning chain between claims and premises, it is possible to systematically reconstruct a meaningful warrant, depending only on what we take as granted and what needs to be explicit. As warrants encode our current presupposed world knowledge and connect the reason with the claim in a given argument, we expect that other warrants can be found which connect the reason with a different claim. In the ex-

[1] Available at https://github.com/UKPLab/argumentreasoning-comprehension-task/, including source codes and supplementary materials.

Proceedings of NAACL-HLT 2018, pages 1930–1940
New Orleans, Louisiana, June 1 - 6, 2018.

Title: Is Marijuana a Gateway Drug? **Description:** Does using marijuana lead to the use of more dangerous drugs, making it too dangerous to legalize?

Reason: Milk isn't a gateway drug even though most people drink it as children. And since {*Warrant 1* | *Warrant 2*}, **Claim:** Marijuana is not a gateway drug.

✔ **Warrant 1:** milk is similar to marijuana

✗ **Warrant 2:** milk is not marijuana

Figure 1: Instance of the argument reasoning comprehension task. The correct warrant has to be identified. Notice the fallacious presupposed false analogy used by the author to make the argument.

treme case, there may exist an *alternative warrant* in which the same reason is connected to the opposite claim.

The intuition of alternative warrants is key to the systematic methodology that we develop in this paper for reconstructing a warrant for the original claim of an argument. In particular, we first 'twist' the stance of a given argument, trying to plausibly explain its reasoning towards the opposite claim. Then, we twist the stance back and use a similar reasoning chain to come up with a warrant for the original argument. As we discuss further below, this works for real-world arguments with a missing piece of information that is taken for granted and considered as common knowledge, yet, would lead to the opposite stance if twisted.

We demonstrate the applicability of our methodology in a large crowdsourcing study. The study results in 1,970 high-quality instances for a new task that we call *argument reasoning comprehension*: Given a reason and a claim, identify the correct warrant from two opposing options. An example is given in Figure 1. A solution to this task will represent a substantial step towards automatic warrant reconstruction. However, we present experiments with several neural attention and language models which reveal that current approaches based on the words and phrases in arguments and warrants do not suffice to solve the task.

The main contributions of this paper are (1) a *methodology* for obtaining implicit warrants realized by means of scalable crowdsourcing and (2) a new *task* along with a high-quality dataset. In addition, we provide (a) 2,884 user-generated arguments annotated for their stance, covering 50+ controversial topics, (b) 2,026 arguments with annotated reasons supporting the stance, (c) 4,235 rephrased reason gists, useful for argument summarization and sentence compression, and (d) a method for checking the reliability of crowdworkers in document and span labeling using traditional inter-annotator agreement measures.

2 Related Work

It is widely accepted that an argument consists of a *claim* and one or more *premises* (reasons) (Damer, 2013). Toulmin (1958) elaborated on a model of argument in which the reason supports the claim on behalf of a *warrant*. The abstract structure of an argument then is *Reason* → (since) *Warrant* → (therefore) *Claim*. The warrant takes the role of an inference rule, similar to the *major premise* in Walton's terminology (Walton, 2007a).

In principle, the chain *Reason* → *Warrant* → *Claim* is applicable to deductive arguments and syllogisms, which allows us to validate arguments properly formalized in propositional logic. However, most natural language arguments are in fact inductive (Govier, 2010, p. 255) or defeasible (Walton, 2007b, p. 29).[2] Accordingly, the unsuitability of formal logic for natural language arguments has been discussed by argumentation scholars since the 1950's (Toulmin, 1958). To be clear, we do not claim that arguments cannot be represented logically (e.g., in predicate logic), however the drift to *informal logic* in the 20th century makes a strong case that natural language argumentation is more than modus ponens (van Eemeren et al., 2014).

In argumentation theory, the notion of a *warrant* has also been contentious. Some argue that the distinction of warrants from premises is clear only in Toulmin's examples but fails in practice, i.e., it is hard to tell whether the reason of a given argument is a premise or a warrant (van Eemeren et al., 1987, p. 205). However, Freeman (2011) provides alternative views on modeling an argument. Given a claim and two or more premises, the argument structure is *linked* if the reasoning step involves the logical conjunction of the premises. If we treat a warrant as a simple premise, then the linked structure fits the intuition behind Toulmin's model, such that premise and warrant combined give support to the claim. For details, see (Freeman, 2011, Chap. 4).

[2] A recent empirical example is provided by Walker et al. (2014) who propose possible approaches to identify patterns of inference from premises to claims in vaccine court cases. The authors conclude that it is extremely rare that a reasoning is explicitly laid out in a deductively valid format.

What makes comprehending and analyzing arguments hard is that claims and warrants are usually implicit (Freeman, 2011, p. 82). As they are 'taken for granted' by the arguer, the reader has to infer the contextually most relevant content that she believes the arguer intended to use. To this end, the reader relies on common sense knowledge (Oswald, 2016; Wilson and Sperber, 2004).

The reconstruction of implicit premises has already been faced in computational approaches. In light of the design of their argument diagramming tool, Reed and Rowe (2004) pointed out that the automatic reconstruction is a task that skilled analysts find both taxing and hard to explain. More recently, Feng and Hirst (2011) as well as Green (2014) outlined the reconstruction of missing enthymemes or warrants as future work, but they never approached it since. To date, the most advanced attempt in this regard is from Boltužić and Šnajder (2016). The authors let annotators 'reconstruct' several propositions between premises and claims and investigated whether the number of propositions correlates with the semantic distance between the claim and the premises. However, they conclude that the written warrants heavily vary both in depth and in content. By contrast, we explore cases with a missing single piece of information that is considered as common knowledge, yet leading to the opposite conclusion if twisted. Recently, Becker et al. (2017) also experimented with reconstructing implicit knowledge in short German argumentative essays. In contrast to our work, they used expert annotators who iteratively converged to a single proposition.

As the task we propose involves natural language comprehension, we also review relevant work outside argumentation here. In particular, the goal of the semantic inference task *textual entailment* is to classify whether a proposition entails or contradicts a hypothesis (Dagan et al., 2009). A similar task, *natural language inference*, was boosted by releasing the large SNLI dataset (Bowman et al., 2015) containing 0.5M entailment pairs crowdsourced by describing pictures. While the understanding of semantic inference is crucial in language comprehension, argumentation also requires coping with phenomena beyond semantics. Rajpurkar et al. (2016) presented a large dataset for reading comprehension by answering questions over Wikipedia articles (SQuAD). In an analysis of this dataset Sugawara and Aizawa (2016) found, though, that only 6.2% of the questions require causal reasoning, 1.2% logical reasoning, and 0% analogy. In contrast, these reasoning types often make up the core of argumentation (Walton, 2007a). Mostafazadeh et al. (2016) introduced the *cloze story test*, in which the appropriate ending of a narrative has to be selected automatically. The overall context of this task is completely different to ours. Moreover, the narratives were written from scratch by explicitly instructing crowd workers, whereas our data come from genuine argumentative comments. Common-sense reasoning was also approached by Angeli and Manning (2014) who targeted the inference of commonsense facts from a large knowledge base. Since their logical formalism builds upon an enhanced version of Aristotle's syllogisms, its applicability to natural language argumentation remains limited (see our discussion above). In contrast to our data source, a few synthetic datasets for general natural language reasoning have been recently introduced, such as answers to questions over a described physical world (Weston et al., 2016) or an evaluation set of 100 questions in the Winograd Schema Challenge (Levesque et al., 2012).

Finally, we note that, although being related, research on argument mining, argumentation quality, and stance classification is not in the immediate scope of this paper. For details on these, we therefore refer to recent papers from Lippi and Torroni (2016); Habernal and Gurevych (2017) or Mohammad et al. (2016).

3 Argument Reasoning Comprehension

Let R be a reason for a claim C, both of which being propositions extracted from a natural language argument. Then there is a warrant W that justifies the use of R as support for C, but W is left implicit.

For example, in a discussion about whether declawing a cat should be illegal, an author takes the following position (which is her claim C): 'It should be illegal to declaw your cat'. She gives the following reason (R): 'They need to use their claws for defense and instinct'.[3] The warrant W could then be 'If cat needs claws for instincts, declawing would be against nature' or similar. W remains implicit, because R already implies C quite obviously and so, according to common sense, any further explanation seems superfluous.

[3]The example is taken from our dataset introduced below.

Now, the question is how to find the warrant W for a given reason R and claim C. Our key hypothesis in the definition of the argument reasoning comprehension task is the existence of an *alternative warrant* AW that justifies the use of R as support for the opposite $\neg C$ of the claim C (regardless of the question of how strong this justification is).

For the example above, assume that we 'twist' C to 'It should be *legal* to declaw your cat' ($\neg C$) but use the same reason R. Is it possible to come up with an alternative warrant AW that justifies R? In the given case, 'most house cats don't face enemies' would bridge R to $\neg C$ quite plausibly. If we now use a reasoning based on AW but twist AW again such that it leads to the claim C, we get 'most house cats face enemies', which is a plausible warrant W for the original argument containing R and C. [4]

Constructing an alternative warrant is not possible for all reason/claim pairs; in some reasons the arguer's position is deeply embedded. As a result, trying to give a plausible reasoning for the opposite claim $\neg C$ either leads to nonsense or to a proposition that resembles a rebuttal rather than a warrant (Toulmin, 1958). However, if both W and AW are available, they usually capture the core of a reason's relevance and reveal the implicit presuppositions (examples follow further below).

Based on our key hypothesis, we define the argument reasoning comprehension task as:

Given a reason R and a claim C along with the title and a short description of the debate they occur in, identify the correct warrant W from two candidates: the correct warrant W and an incorrect alternative warrant AW.

An instance of the task is thus basically given by a tuple (R, C, W, AW). The debate title and description serve as the context of R and C. As it is binary, we propose to evaluate the task using accuracy.

4 Reconstruction of Implicit Warrants

We now describe our methodology to systematically reconstruct implicit warrants, along with the scalable crowdsourcing process that operationalizes the methodology. The result of the process is a dataset with authentic instances (R, C, W, AW) of the argument reasoning comprehension task.

4.1 Source Data

Instead of extending an existing dataset, we decided to create a new one from scratch, because we aimed to study a variety of controversial issues in user-generated web comments and because we sought for a dataset with a permissive license.

As a source, we opted for the *Room for Debate* section of the New York Times.[5] It provides authentic argumentation on contemporary issues with good editorial work and moderation — as opposed to debate portals such as *createdebate.com*, where classroom assignments, silly topics, and bad writing prevail. We manually selected 188 debates with polar questions in the title. These questions are controversial and provoking, giving a stimulus for stance-taking and argumentation.[6] For each debate we created two explicit opposing claims, e.g., 'It should be illegal to declaw your cat' and 'It should be legal to declaw your cat'. We crawled all comments from each debate and sampled about 11k high-ranked, root-level comments.[7]

4.2 Methodology and Crowdsourcing Process

The methodology we propose consists of eight consecutive steps that are illustrated in Figure 2 and detailed below. Each step can be operationalized with crowdsourcing. For our dataset, we performed crowdsourcing on 5,000 randomly sampled comments using Amazon Mechanical Turk (AMT) from December 2016 to April 2017. Before, each comment was split into elementary discourse units (EDUs) using SistaNLP (Surdeanu et al., 2015).

1. Stance Annotation For each comment, we first classify what stance it is taking (recall that we always have two explicit claims with opposing stance). Alternatively, it may be neutral (consider-

[4] This way, we also reveal the weakness of the original argument that was hidden in the implicit premise. It can be challenged by asking the arguer whether house cats really face enemies.

[5] https://www.nytimes.com/roomfordebate

[6] Detailed theoretical research on polar and alternative questions can be found in (van Rooy and Šafářová, 2003); Asher and Reese (2005) analyze bias and presupposition in polar questions.

[7] To remove 'noisy' candidates, we applied several criteria, such as the absence of quotations or URLs and certain lengths. For details, see the source code we provide. We did not check any quality criteria of arguments, as this was not our focus; see, e.g., (Wachsmuth et al., 2017) for argumentation quality.

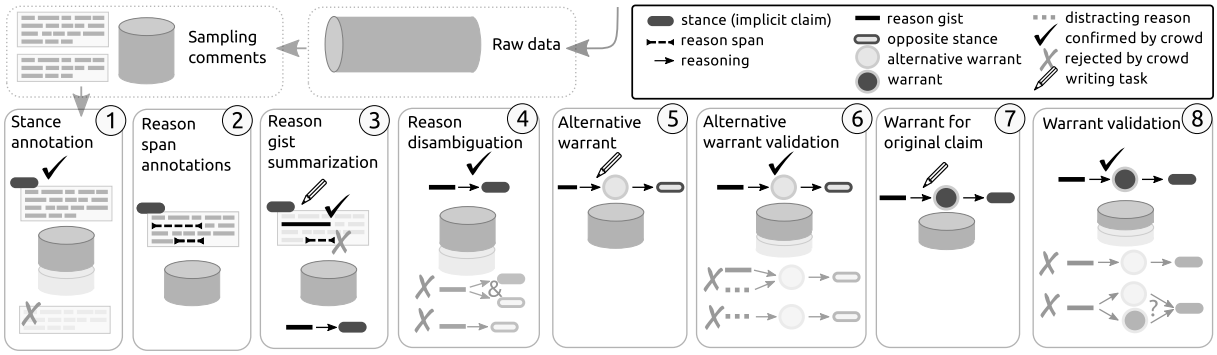


Figure 2: Overview of the methodology of reconstructing implicit warrants for argument reasoning comprehension.

ing both sides) or may not take any stance.[8]

All 2,884 comments in our dataset classified as stance-taking by the crowdworkers were then also annotated as to whether being sarcastic or ironic; both pose challenges in analyzing argumentation not solved so far (Habernal and Gurevych, 2017).

2. Reason Span Annotation For all comments taking a stance, the next step is to select those spans that give a reason for the claim (with a single EDU as the minimal unit).

In our dataset, the workers found 5,119 reason spans, of which 2,026 lay within arguments. About 40 comments lacked any explicit reason.

3. Reason Gist Summarization This new task is, in our view, crucial for downstream annotations. Each reason from the previous step is rewritten, such that the reason's gist in the argument remains the same but the clutter is removed (examples are given in the supplementary material which is available both in the ACL Anthology and the project GitHub site). Besides, wrongly annotated reasons are removed in this step. The result is pairs of reason R and claim C.

All 4,294 gists in our dataset were summarized under Creative Commons Zero license (CC-0).

4. Reason Disambiguation Within our methodology, we need to be able to identify to what extent a reason itself implies a stance: While 'C because R' allows for many plausible interpretations (as discussed above), whether $R \rightarrow C$ or $R \rightarrow \neg C$ depends on how much presupposition is encoded in R. In this step, we decide which claim (C or $\neg C$) is most plausible for R, or whether both are similarly plausible (in the given data, respective reasons turned out to be rather irrelevant though).

We used only those 1,955 instances where R indeed implied C according to the workers, as this suggests at least some implicit presupposition in R.

5. Alternative Warrant This step is the trickiest, since it requires both creativity and 'brain twisting'. As exemplified in Section 3, a plausible explanation needs to be given why R supports $\neg C$ (i.e., the alternative warrant AW). Alternatively, this may be classified as being impossible.

Exact instructions for our workers can be found in the provided sources. All 5,342 alternative warrants in our dataset are written under CC-0 license.

6. Alternative Warrant Validation As the previous step produces largely uncontrolled writings, we validate each fabricated alternative warrant AW as to whether it actually relates to the reason R. To this end, we show AW and $\neg C$ together with two alternatives: R itself and a distracting reason. Only instances with correctly validated R are kept.

For our dataset, we sampled the distracting reason from the same debate topic, using the most dissimilar to R in terms of skip-thought vectors (Kiros et al., 2015) and cosine similarity. We kept 3,791 instances, for which the workers also rated how 'logical' the explanation of AW was (0–2 scale).

7. Warrant For Original Claim This step refers to the second task in the example from Section 3: Given R and C, make minimal modifications to the alternative warrant AW, such that it becomes an actual warrant W (i.e., such that $R \rightarrow W \rightarrow C$).

For our dataset, we restricted this step to those 2,613 instances that had a 'logic score' of at least 0.68 (obtained from the annotations mentioned

[8] We also experimented with approaching the annotations top-down starting by annotating explicit claims, but the results were unsatisfying. This is in line with empirical observations made by Habernal and Gurevych (2017) who showed that the majority of claims in user-generated arguments are implicit.

above), in order to filter out nonsense alternative warrants. All resulting 2,447 warrants were written by the workers again under CC0 license.

8. Warrant Validation To ensure that each tuple (R,C,W,AW) allows only one logical explanation (i.e., either $R \rightarrow W \rightarrow C$ or $R \rightarrow AW \rightarrow C$ is correct, not both), all instances are validated again.

Disputed cases in the dataset (according to our workers) were fixed by an expert to ensure quality. We ended up with 1,970 instances to be used for the argument reasoning comprehension task.

4.3 Agreement and Dataset Statistics

To strictly assess quality in the entire crowdsourcing process, we propose an evaluation method that enables 'classic' inter-annotator agreement measures for crowdsourcing, such as Fleiss' κ or Krippendorff's α. Applying κ and α directly to crowdsourced data has been disputed (Passonneau and Carpenter, 2014). For estimating gold labels from the crowd, several models have been proposed; we rely on MACE (Hovy et al., 2013). Given a number of noisy workers, MACE outputs best estimates, outperforming simple majority votes. At least five workers are recommended for a crowdsourcing task, but how reliable is the output really?

We hence collected 18 assignments per item and split them into two groups (9+9) based on their submission time. We then considered each group as an independent crowdsourcing experiment and estimated gold labels using MACE for each group, thus yielding two 'experts from the crowd.' Having two independent 'experts' from the crowd allowed us to compute standard agreement scores. We also varied the size of the sub-sample from each group from 1 to 9 by repeated random sampling of assignments. This revealed how the score varies with respect to the crowd size per 'expert'.

Figure 3 shows the Cohen's κ agreement for stance annotation with respect to the crowd size computed by our method. As MACE also includes a threshold for keeping only the most confident predictions in order to benefit precision, we tuned this parameter, too. Deciding on the number of workers per task is a trade-off between the desired quality and the budget. For example, reason span annotation is a harder task; however, the results for six workers are comparable to those for the expert annotations of Habernal and Gurevych (2017).[9]

[9]The supplementary material contains a detailed figure; not to be confused with Figure 3 which refers to stance annotation.

Figure 3: Cohen's κ agreement for stance annotation on 98 comments. As a trade-off between reducing costs (i.e., discarding fewer instances) and increasing reliability, we chose 5 annotators and a threshold of 0.95 for this task, which resulted in $\kappa = 0.58$ (moderate to substantial agreement).

Table 1 lists statistics of the entire crowdsourcing process carried out for our dataset, including tasks for which we created data as a by-product.

4.4 Examples

Below, we show three examples in which implicit common-sense presuppositions were revealed during the construction of the alternative warrant *AW* and the original warrant *W*. For brevity, we omit the debate title and description here. A full walkthrough example is found in the supplementary material.

R: Cooperating with Russia on terrorism ignores Russia's overall objectives.
C: Russia cannot be a partner.
AW: Russia has the same objectives of the US.
W: Russia has the opposite objectives of the US.

R: Economic growth needs innovation.
C: 3-D printing will change the world.
AW: There is no innovation in 3-d printing since it's unsustainable.
W: There is much innovation in 3-d printing and it is sustainable.

R: College students have the best chance of knowing history.
C: College students' votes do matter in an election.
AW: Knowing history doesn't mean that we will repeat it.
W: Knowing history means that we won't repeat it.

5 Experiments

Given the dataset, we performed first experiments to assess the complexity of argument reasoning comprehension. To this end, we split the 1,970 instances into three sets based on the year of the de-

#	Methodology Step	Input Sata	Size	Output Data	Size	Quality Assurance	Use of Data
1	Stance annotation	Comment, topic	5,000	Stance-taking arguments	2,884	Cohen's κ 0.58	Argument stance detection; sarcastic argument detection
2	Reason span annotation	Stance-taking argument	2,884	Reason spans (in arguments)	5,119 (2,026)	Krippendorff's α_u 0.51	Argument component detection; argumentative text segmentation
3	Reason gist summarization	Claim, reason span	5,119	Summarized reason gists (in arguments)	4,294 (1,927)	Qualified workers, manual inspection	Abstractive argument summarization; reason clustering; empirical analysis of controversies
4	Reason disambiguation	Reason gist, both claims	4,235	Reasons implying original stance	1,955	Cohen's κ 0.42 (task-important categories)	Argument component stance detection
5	Writing alternative warrant	Reason gist, opposing claim	1,955	Fabricated warrant for reason and opposing claim	5,342	Qualified workers, manual inspection	–
6	Alternative warrant validation	Opposing claim, alternative warrant, reason, distracting reason	5,342	Plausible triple of reason, alternative warrant, and opposing claim	3,791	–	Reason/Warrant relevance detection
7	Writing warrant for original claim	Claim, reason, alternative warrant	2,613*	Warrant similar to alternative warrant for reason and claim	2,447	Qualified workers, manual inspection	–
8	Warrant validation	Claim, reason, warrant, alternative warrant	2,447	Validated triple of reason, warrant, and claim	1,970	Qualified workers, experts for hard cases	Argument reasoning comprehension (our main task)

Table 1: Details and statistics of the datasets resulting from the eight steps of our methodology implemented in a crowdsourcing process. *Input instances were filtered by their 'logic score' assigned in Step 6, such that the weakest 30% were discarded. A more detailed description is available in the readme file of the source code.

bate they were taken from: 2011–2015 became the training set (1,210 instances), 2016 the development set (316 instances), and 2017 the test set (444 instances). This follows the paradigm of learning on past data and predicting on new ones. In addition, it removes much lexical and topical overlap.

5.1 Human Upper Bounds

To evaluate human upper bounds for the task, we sampled 100 random questions (such as those presented in Section 4.4) from the test set and distributed them among 173 participants of an AMT survey. Every participant had to answer 10 questions. We also asked the participants about their highest completed education (six categories) and the amount of formal training they have in reasoning, logic, or argumentation (no training, some, or extensive). In addition, they specified for each question how familiar they were with the topic (3-point scale).

How Hard is the Task for Humans? It depends, as shown in Figure 4. Whereas education had almost negligible influence on the performance, the more extensive formal training in reasoning the participants had, the higher their score was. Overall, 30 of the 173 participants scored 100%. The mean score for those with extensive

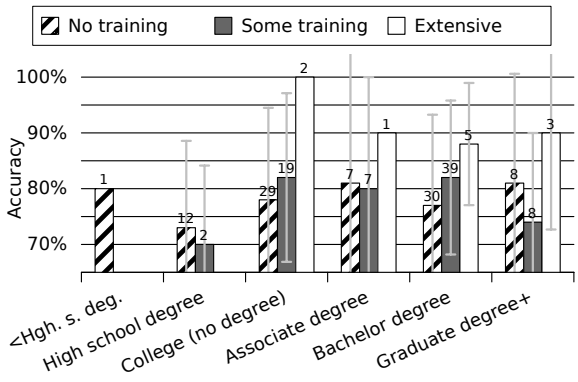


Figure 4: Human upper bounds on the argument reasoning comprehension task with respect to education and formal training in reasoning, logic, or argumentation. For each configuration, the mean values are displayed together with the number of participants (above the bar) and with their standard deviations (error bars).

formal training was 90.9%. For all participants, the mean was 79.8%. However, we have to note that some of the questions are more difficult than others, for which we could not control explicitly.

Does Topic Familiarity Affect Human Performance? Not really, i.e., we found no significant (Spearman) correlation between the mean score and familiarity of a participant in almost all education/training configurations. This suggests that ar-

gument reasoning comprehension skills are likely to be independent of topic-specific knowledge.

5.2 Computational Models

To assess the complexity of computationally approaching argument reasoning comprehension, we carried out first experiments with systems based on the following models.

The simplest considered model was the *random baseline*, which chooses either of the candidate warrants of an instance by chance. As another baseline, we used a 4-gram Modified Kneser-Ney *language model* trained on 500M tokens (100k vocabulary) from the C4Corpus (Habernal et al., 2016). The effectiveness of language models was demonstrated by Rudinger et al. (2015) for the narrative cloze test where they achieved state-of-the-art results. We computed log-likelihood of the candidate warrants and picked the one with lower score.[10]

To specifically appoach the given task, we implemented two neural models based on a bidirectional LSTM. In the standard *attention* version, we encoded the reason and claim using a BiLSTM and provided it as an attention vector after max-pooling to LSTM layers from the two available warrants W_0 and W_1 (corresponding to W and AW, see below). Our more elaborated version used *intra-warrant attention*, as shown in Figure 5. Both versions were also extended with the debate title and description added as context to the attention layer (*w/ context*). We trained the resulting four models using the ADAM optimizer, with heavy dropout (0.9) and early stopping (5 epochs), tuned on the development set. Input embeddings were pre-trained word2vec's (Mikolov et al., 2013). We ran each model three times with random initializations.

To evaluate all systems, each instance in our dataset is represented as a tuple (R, C, W_0, W_1) with a label (0 or 1). If the label is 0, W_0 is the correct warrant, otherwise W_1. Recall that we have two warrants W and AW whose correctness depends on the claim: W is correct for R and the original claim C, whereas AW would be correct for R and the opposite claim $\neg C$. We thus doubled the training data by adding a permuted instance (R, C, W_1, W_0) with the respective correct label; this led to increased performance. The overall

[10] This might seem counterintuitive, but since W is created by rewriting AW, it may suffer from some dis-coherency, which is then caught by the language model.

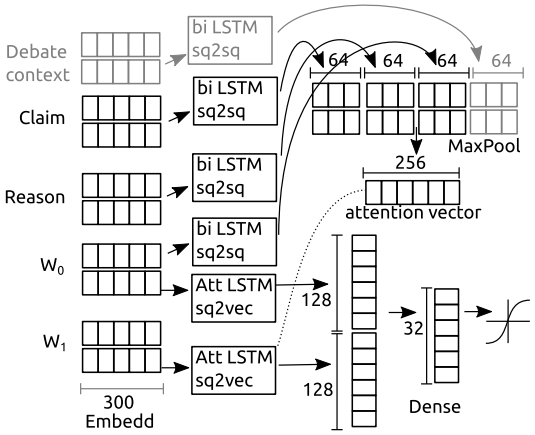


Figure 5: Intra-warrant attention. Only the attention vector for the warrant W_1 is shown; the attention vector for W_0 is constructed analogously. Grey areas represent a modification with additional context.

results of all approaches (humans and systems) are shown in Table 2. Intra-warrant attention with rich context outperforms standard neural models with a simple attention, but it only slightly beats the language model on the dev set. The language model is basically random on the test set.

A manual error analysis of 50 random wrong predictions (a single run of the best-performing system on the dev set) revealed no explicit pattern of encountered errors. Drawing any conclusions is hard given the diversity of included topics and the variety of reasoning patterns. A possible approach would be to categorize warrants using, e.g., argumentation schemes (Walton et al., 2008) and break down errors accordingly. However, this is beyond the scope here and thus left for future work.

Can We Benefit from Alternative Warrants and Opposite Claims? Since the reasoning chain $R \rightarrow AW \rightarrow \neg C$ is correct, too, we also tried adding respective instances to the training set (thus doubling the size). In this configuration, however, the neural models failed to learn anything better than a random guess. The reason behind is probably that the opposing claims are lexically very close, usually negated, and the models cannot pick this up. This underlines that argument reasoning comprehension cannot be solved by simply looking at the occurring words or phrases.

6 Conclusion and Outlook

We presented a new task called *argument reasoning comprehension* that tackles the core of reasoning in natural language argumentation — im-

Approach	Dev	(±)	Test	(±)
Human average			.798	.162
Human w/ training in reasoning			.909	.114
Random baseline	.473	.039	.491	.031
Language model	.617		.500	
Attention	.488	.006	.513	.012
Attention w/ context	.502	.031	.512	.014
Intra-warrant attention	**.638**	.024	.556	.016
Intra-warrant attent. w/ context	.637	.040	**.560**	.055

Table 2: Accuracy of each approach (humans and systems) on the development set and test set, respectively.

plicit warrants. Moreover, we proposed a methodology to systematically reconstruct implicit warrants in eight consecutive steps. So far, we implemented the methodology in a manual crowdsourcing process, along with a strategy that enables standard inter-annotator agreement measures in crowdsourcing.

Following the process, we constructed a new dataset with 1,970 instances for the task. This number might not seem large (e.g., compared to 0.5M from SNLI), but tasks with hand-crafted data are of a similar size (e.g., 3,744 Story Cloze Test instances). Also, the crowdsourcing process is scalable and is limited only by the budget.[11] Moreover, we created several data 'by-products' that are valuable for argumentation research: 5,000 comments annotated with stance, which outnumbers the 4,163 tweets for stance detection of Mohammad et al. (2016); 2,026 arguments with 4,235 annotated reasons, which is six times larger than the 340 documents of Habernal and Gurevych (2017); and 4,235 summarized reason gists — we are not aware of any other hand-crafted dataset for abstractive argument summarization built upon authentic arguments.

Based on the dataset, we evaluated human performance in argument reasoning comprehension. Our findings suggest that the task is harder for people without formal argumentation training, while being solvable without knowing the topic. We also found that neural attention models outperform language models on the task.

In the short run, we plan to draw more attention to this topic by running a SemEval 2018 shared task.[12] A deep qualitative analysis of the warrants from the theoretical perspective of reasoning patterns or argumentation schemes is also necessary. In the long run, an automatic generation and validation warrants can be understood as the ultimate goal in argument evaluation. It has been claimed that for reconstructing and evaluating natural language arguments, one has to fully 'roll out' their implicit premises (van Eemeren et al., 2014, Chap. 3.2) and leverage knowledge bases (Wyner et al., 2016). We believe that a system that can distinguish between the wrong and the right warrant given its context will be helpful in filtering out good candidates in argument reconstruction.

For the moment, we just made a first empirical step towards exploring how much common-sense reasoning is necessary in argumentation and how much common sense there might be at all.

[11] In our case, the total costs were about $6,000 including bonuses and experiments with the workflow set-up.

[12] https://competitions.codalab.org/competitions/17327

Acknowledgments

This work has been supported by the ArguAna Project GU 798/20-1 (DFG), and by the DFG-funded research training group "Adaptive Preparation of Information form Heterogeneous Sources" (AIPHES, GRK 1994/1).

References

Ruth Amossy. 2009. The New Rhetoric's Inheritance. Argumentation and Discourse Analysis. *Argumentation* 23(3):313–324. https://doi.org/10.1007/s10503-009-9154-y.

Gabor Angeli and Christopher D Manning. 2014. NaturalLI: Natural Logic Inference for Common Sense Reasoning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 534–545. http://www.aclweb.org/anthology/D14-1059.

Nicholas Asher and Brian Reese. 2005. Negative Bias in Polar Questions. In Emar Maier, Corien Bary, and Janneke Huitink, editors, *Proceedings of Sinn und Bedeutung (SuB9)*. Nijmegen Centre of Semantics, Nijmegen, NL, pages 30–43.

Maria Becker, Michael Staniek, Vivi Nastase, and Anette Frank. 2017. Enriching Argumentative Texts with Implicit Knowledge. In *Proceedings of NLDB*. Springer International Publishing, Liége, Belgium, pages 84–96. https://doi.org/10.1007/978-3-319-59569-6_9.

Filip Boltužić and Jan Šnajder. 2016. Fill the Gap! Analyzing Implicit Premises between Claims from Online Debates. In *Proceedings of the Third Workshop on Argument Mining*. Association for Computational Linguistics, Berlin, Germany, pages 124–133.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 632–642. http://aclweb.org/anthology/D15-1075.

Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. 2009. Recognizing textual entailment: Rational, evaluation and approaches. *Natural Language Engineering* 15(Special Issue 04):i–xvii. https://doi.org/10.1017/S1351324909990209.

T. Edward Damer. 2013. *Attacking Faulty Reasoning: A Practical Guide to Fallacy-Free Arguments*. Cengage Learning, Boston, MA, 7th edition.

Vanessa Wei Feng and Graeme Hirst. 2011. Classifying arguments by scheme. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*. Association for Computational Linguistics, Portland, Oregon, HLT '11, pages 987–996.

James B. Freeman. 2011. *Argument Structure: Representation and Theory*, volume 18 of *Argumentation Library*. Springer Netherlands.

Trudy Govier. 2010. *A Practical Study of Argument*. Wadsworth, Cengage Learning, 7th edition.

Nancy L Green. 2014. Towards Creation of a Corpus for Argumentation Mining the Biomedical Genetics Research Literature. In *Proceedings of the First Workshop on Argumentation Mining*. Association for Computational Linguistics, Baltimore, Maryland USA, pages 11–18.

Ivan Habernal and Iryna Gurevych. 2017. Argumentation Mining in User-Generated Web Discourse. *Computational Linguistics* 43(1):125–179. https://doi.org/10.1162/COLI_a_00276.

Ivan Habernal, Omnia Zayed, and Iryna Gurevych. 2016. C4Corpus: Multilingual Web-size Corpus with Free License. In *LREC*. Portorož, Slovenia, pages 914–922. http://www.lrec-conf.org/proceedings/lrec2016/pdf/388_Paper.pdf.

Jerry R. Hobbs, Mark E. Stickel, Douglas E. Appelt, and Paul Martin. 1993. Interpretation as abduction. *Artificial Intelligence* 63(1):69 –142. https://doi.org/10.1016/0004-3702(93)90015-4.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with MACE. In *Proceedings of NAACL-HLT 2013*. Association for Computational Linguistics, Atlanta, Georgia, pages 1120–1130. http://www.aclweb.org/anthology/N13-1132.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*. pages 3294–3302.

Hector J. Levesque, Ernest Davis, and Leora Morgenstern. 2012. The Winograd Schema Challenge. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*. Association for the Advancement of Artificial Intelligence, Rome, Italy, pages 552–561. http://www.aaai.org/ocs/index.php/KR/KR12/paper/download/4492/4924.

Marco Lippi and Paolo Torroni. 2016. Argumentation mining: State of the art and emerging trends. *ACM Transactions on Internet Technology* 16(2):10:1–10:25. https://doi.org/10.1145/2850417.

Fabrizio Macagno and Douglas Walton. 2014. *Emotive Language in Argumentation*. Cambridge University Press.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., pages 3111–3119.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. Semeval-2016 Task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. Association for Computational Linguistics, San Diego, California, pages 31–41. http://www.aclweb.org/anthology/S16-1003.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A Corpus and Cloze Evaluation for Deeper Understanding of Commonsense Stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, CA, USA, pages 839–849. http://www.aclweb.org/anthology/N16-1098.

S. Newman and C. Marshall. 1991. Pushing Toulmin Too Far: Learning From an Argument Representation Scheme. Technical report, Xerox Palo Alto Research Center, Palo Alto, CA.

Steve Oswald. 2016. Commitment attribution and the reconstruction of arguments. In Fabio Paglieri, Laura Bonelli, and Silvia Felletti, editors, *The Psychology of Argument: Cognitive Approaches to Argumentation and Persuasion*, College Publications, pages 17–32.

Rebecca J. Passonneau and Bob Carpenter. 2014. The Benefits of a Model of Annotation. *Transactions of the Association for Computational Linguistics* 2:311–326. http://aclweb.org/anthology/Q/Q14/Q14-1025.pdf.

Gilbert Plumer. 2016. Presumptions, Assumptions, and Presuppositions of Ordinary Arguments. *Argumentation* 31(3):469–484. https://doi.org/10.1007/s10503-016-9419-1.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 2383–2392. https://aclweb.org/anthology/D16-1264.

Chris Reed and Glenn Rowe. 2004. Araucaria: software for argument analysis, diagramming and representation. *International Journal on Artificial Intelligence Tools* 13(04):961–979. https://doi.org/10.1142/S0218213004001922.

Rachel Rudinger, Pushpendre Rastogi, Francis Ferraro, and Benjamin Van Durme. 2015. Script induction as language modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1681–1686. http://aclweb.org/anthology/D15-1195.

Saku Sugawara and Akiko Aizawa. 2016. An Analysis of Prerequisite Skills for Reading Comprehension. In *Proceedings of the Workshop on Uphill Battles in Language Processing: Scaling Early Achievements to Robust Methods*. Association for Computational Linguistics, Austin, TX, USA, pages 1–5. http://aclweb.org/anthology/W16-6001.

Mihai Surdeanu, Tom Hicks, and Marco Antonio Valenzuela-Escarcega. 2015. Two practical rhetorical structure theory parsers. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*. Association for Computational Linguistics, Denver, Colorado, pages 1–5. http://www.aclweb.org/anthology/N15-3001.

Stephen E. Toulmin. 1958. *The Uses of Argument*. Cambridge University Press.

Frans H. van Eemeren, Bart Garssen, Erik C. W. Krabbe, A. Francisca Snoeck Henkemans, Bart Verheij, and Jean H. M. Wagemans. 2014. *Handbook of Argumentation Theory*. Springer, Berlin/Heidelberg.

Frans H. van Eemeren, Rob Grootendorst, and Tjark Kruiger. 1987. *Handbook of argumentation theory: A critical survey of classical backgrounds and modern studies*. Foris Publications, Dordrecht, Netherlands.

Robert van Rooy and Marie Šafářová. 2003. On polar questions. In Robert B. Young and Yuping Zhou, editors, *Proceedings of the 13th Semantics and Linguistic Theory Conference (SALT XIII)*. Ithaka, NY, USA, pages 292–309.

Henning Wachsmuth, Nona Naderi, Ivan Habernal, Yufang Hou, Graeme Hirst, Iryna Gurevych, and Benno Stein. 2017. Argumentation Quality Assessment: Theory vs. Practice. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 250–255. http://aclweb.org/anthology/P17-2039.

Vern R Walker, Karina Vazirova, and Cass Sanford. 2014. Annotating Patterns of Reasoning about Medical Theories of Causation in Vaccine Cases: Toward a Type System for Arguments. In *Proceedings of the First Workshop on Argumentation Mining*. Association for Computational Linguistics, Baltimore, Maryland USA, pages 1–10.

Douglas Walton. 2007a. *Dialog Theory for Critical Argumentation*. John Benjamins Publishing Company, 5 edition.

Douglas Walton. 2007b. *Media Argumentation: Dialect, Persuasion and Rhetoric*. Cambridge University Press.

Douglas Walton. 2012. Using argumentation schemes for argument extraction: A bottom-up method. *International Journal of Cognitive Informatics and Natural Intelligence* 6(3):33–61.

Douglas Walton, Christopher Reed, and Fabrizio Macagno. 2008. *Argumentation Schemes*. Cambridge University Press.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M. Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2016. Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks. In *Procedings of the 5th International Conference on Learning Representations (ICLR)*. San Juan, Puerto Rico, pages 1–14. http://arxiv.org/abs/1502.05698.

Deirdre Wilson and Dan Sperber. 2004. Relevance Theory. In Laurence R. Horn and Gregory Ward, editors, *The Handbook of Pragmatics*, Wiley-Blackwell, Oxford, UK, chapter 27, pages 607–632.

Adam Wyner, Tom van Engers, and Anthony Hunter. 2016. Working on the argument pipeline: Through flow issues between natural language argument, instantiated arguments, and argumentation frameworks. *Argument & Computation* 7(1):69–89. https://doi.org/10.3233/AAC-160002.

Linguistic Cues to Deception and Perceived Deception in Interview Dialogues

Sarah Ita Levitan, Angel Maredia, & Julia Hirschberg
Department of Computer Science
Columbia University
New York, NY, USA
{sarahita@cs, asm2221, julia@cs}.columbia.edu

Abstract

We explore deception detection in interview dialogues. We analyze a set of linguistic features in both truthful and deceptive responses to interview questions. We also study the perception of deception, identifying characteristics of statements that are perceived as truthful or deceptive by interviewers. Our analysis show significant differences between truthful and deceptive question responses, as well as variations in deception patterns across gender and native language. This analysis motivated our selection of features for machine learning experiments aimed at classifying globally deceptive speech. Our best classification performance is 72.74 F1-Score (about 27% better than human performance), which is achieved using a combination of linguistic features and individual traits.

1 Introduction

Deception detection is a critical problem studied by psychologists, criminologists, and computer scientists. In recent years the NLP and speech communities have increased their interest in deception detection. Language cues are inexpensive and easy to collect, and research examining text-based and speech-based cues to deception has been quite promising. Prior work has examined deceptive language in several domains, including fake reviews, mock crime scenes, and opinions about topics such as abortion or the death penalty. In this work we explore the domain of interview dialogues, which are similar to many real-world deception conditions.

Previous work has presented the results of classification experiments using linguistic features, attempting to identify which features contribute most to classification accuracy. However, studies often do not include an empirical analysis of features. We might know that a particular feature set (e.g. LIWC categories) is useful for deception classification, but we lack insight about the nature of the deceptive and truthful language that makes the feature set useful, and whether the differences in language use are statistically significant. In this work we conduct an empirical analysis of feature sets and report on the different characteristics of truthful and deceptive language. In addition, previous work has focused on the characteristics of deceptive language, and not on the characteristics of *perceived* deceptive language. We are also interested in human perception of deception; that is, what are the characteristics of language that listeners perceive as truthful or deceptive? We examine a unique dataset that includes information about both the deceiver and the interviewer, along with interviewer judgments of deception. Along with an analysis of deceptive and truthful speech, we analyze the believed and disbelieved speech, according to reported interviewer judgments. Finally, previous work has focused on general inferences about deception; here we include analysis of gender and native language, to study their effect on deceptive behavior, and also their effect on perception of deception. This work contributes to the critical problem of automatic deception detection, and increases our scientific understanding of deception, deception perception, and speaker differences in deceptive behavior.

The paper is organized as follows: In Section 2 we review related work in language-based cues to deception. Section 3 describes the dataset used for this work, and Section 4 details the different feature sets we employ. In Section 5, we report on the results of our empirical study of indicators of deception and perceived deception, as well as gender and native language differences. Section 6 presents our machine learning classification results using the deception indicator feature sets. We conclude in Section 7 with a discussion and ideas

Proceedings of NAACL-HLT 2018, pages 1941–1950
New Orleans, Louisiana, June 1 - 6, 2018.

for future work.

2 Related Work

Language-based cues to deception have been analyzed in many genres. Ott et al. (2011) compared approaches to automatically detecting deceptive opinion spam, using a crowdsourced dataset of fake hotel reviews. Several studies use a fake opinion paradigm for collecting data, instructing subjects to write or record deceptive and truthful opinions about controversial topics such as the death penalty or abortion, or about a person that they like/dislike (Newman et al., 2003; Mihalcea and Strapparava, 2009). Other research has focused on real-world data obtained from court testimonies and depositions (Fornaciari and Poesio, 2013; Bachenko et al., 2008; Pérez-Rosas et al., 2015). Real-world deceptive situations are high-stakes, where there is much to be gained or lost if deception succeeds or fails; it is hypothesized that these conditions are more likely to elicit strong cues to deception. However, working with such data requires extensive research to annotate each utterance for veracity, so such datasets are often quite small and not always reliable.

Linguistic features such as n-grams and language complexity have been analyzed as cues to deception (Pérez-Rosas and Mihalcea, 2015; Yancheva and Rudzicz, 2013). Syntactic features such as part of speech tags have also been found to be useful for structured data (Ott et al., 2011; Feng et al., 2012). Statement Analysis (Adams, 1996) is a text-based deception detection approach that combines lexical and syntactic features. An especially useful resource for text-based deception detection is the Linguistic Inquiry and Word Count (LIWC) (Pennebaker and King, 1999), which groups words into psychologically motivated categories. In addition to lexical features, some studies have examined acoustic-prosodic cues to deception (Rockwell et al., 1997; Enos, 2009; Mendels et al., 2017). (Benus et al., 2006) studied pause behavior in deceptive speech. This work is very promising, but it is more difficult to obtain large, cleanly recorded speech corpora with deception annotations than to obtain text corpora. An excellent meta-study of verbal cues to deception can be found in (DePaulo et al., 2003).

3 Data

3.1 Corpus

For this work, we examined the Columbia X-Cultural Deception (CXD) Corpus (Levitan et al., 2015a) a collection of within-subject deceptive and non-deceptive speech from native speakers of Standard American English (SAE) and Mandarin Chinese (MC), all speaking in English. The corpus contains dialogues between 340 subjects. A variation of a fake resume paradigm was used to collect the data. Previously unacquainted pairs of subjects played a ”lying game” with each other. Each subject filled out a 24-item biographical questionnaire and were instructed to create false answers for a random half of the questions. They also reported demographic information including gender and native language, and completed the NEO-FFI personality inventory (Costa and McCrae, 1989).

The lying game was recorded in a sound booth. For the first half of the game, one subject assumed the role of the interviewer, while the other answered the biographical questions, lying for half and telling the truth for the other; questions chosen in each category were balanced across the corpus. For the second half of the game, the subjects roles were reversed, and the interviewer became the interviewee. During the game, the interviewer was allowed to ask the 24 questions in any order s/he chose; the interviewer was also encouraged to ask follow-up questions to aid them in determining the truth of the interviewees answers. Interviewers recorded their judgments for each of the 24 questions, providing information about human perception of deception. The entire corpus was orthographically transcribed using the Amazon Mechanical Turk (AMT)[1] crowd-sourcing platform, and the speech was segmented into *inter-pausal units* (IPUs), defined as pause-free segments of speech separated by a minimum pause length of 50 ms. The speech was also segmented into turn units, where a turn is defined as a maximal sequence of IPUs from a single speaker without any interlocutor speech that is not a *backchannel*. There are two forms of deception annotations in the corpus: local and global. Interviewees labeled their responses with local annotations by pressing a ”T” or ”F” key for each utterance as they spoke. These keypresses were automatically aligned with speaker IPUs and turns. Global la-

[1] https://www.mturk.com/mturk/

bels were provided by the biographical questionnaire, where each of the 24 questions was labeled as truthful or deceptive.

Consider the following dialogue:
Interviewer: What is your mother's job?
Interviewee: My mother is a doctor (F). She has always worked very late hours and I felt neglected as a child (T).

Is the interviewee response true or false? We differentiate between global and local deception. Globally, the response to the question is deceptive. However, it contains local instances of both truth and deception. In this work we focus on dialogue-based deception, using global deception labels.

3.2 Global Segmentation

Previous work with the CXD corpus has focused on IPU-level and turn-level analysis and classification of local deception, mostly with acoustic-prosodic features (Levitan et al., 2015b; Mendels et al., 2017). Here we are interested in exploring global deception at the dialogue-level for the first time in this corpus. We define response-segments as sets of turns that are related to a single question (of the 24 interview questions). In order to annotate these segments, we first used a question detection and identification system (Maredia et al., 2017) that uses word embeddings to match semantically similar variations of questions to a target question list. This was necessary because interviewers asked the 24 questions using different wording from the original list of questions. On this corpus, (Maredia et al., 2017) obtained an F1-score of .95%.

After tagging interviewer turns with this system, we labeled the set of interviewee turns between two interviewer questions q1 and q2 as corresponding to question q1. The intuition behind this was that those turns were responses to follow up questions related to q1, and while the question detection and identification system discussed above did not identify follow up questions, we found that most of the follow up questions after an interviewer question q1 would be related to q1 in our hand annotation. We evaluated this global segmentation on a hand-annotated test set of 17 interviews (about 10% of the corpus) consisting of 2,671 interviewee turns, 408 interviewer questions, and 977 follow up questions. Our global segmentation approach resulted in 77.8% accuracy on our hand-labeled test set (errors were mostly due to turns that were unrelated to any question).

We performed our analysis and classification on two segmentations of the data using this tagging method: (1) **first turn:** we analyzed only the single interviewee turn directly following the original question, and (2) **multiple turns** we analyzed the entire segment of interviewee turns that were responding to the original interviewer question and subsequent follow-up questions. In our classification experiments, we explore whether a deceptive answer is be better classified by the interviewee's initial response or by all of the follow-up conversation between interviewer and interviewee.

4 Features

LIWC Previous work has found that deceivers tend to use different word usage patterns when they are lying (Newman et al., 2003). We used LIWC (Pennebaker et al., 2001) to extract semantic features from each utterance. LIWC is a text analysis program that computes features consisting of normalized word counts for 93 semantic classes. LIWC dimensions have been used in many studies to predict outcomes including personality (Pennebaker and King, 1999), deception (Newman et al., 2003), and health (Pennebaker et al., 1997). We extracted a total of 93 features using LIWC 2015 [2], including standard linguistic dimensions (e.g. percentage of words that are pronouns, articles), markers of psychological processes (e.g. affect, social, cognitive), punctuation categories (e.g periods, commas), and formality measures (e.g. fillers, swear words).

Linguistic We extracted 23 linguistic features [3] which we adopted from previous deception studies such as (Enos, 2009; Bachenko et al., 2008). Included in this list are binary and numeric features capturing hedge words, filled pauses, laughter, complexity, contractions, and denials. We include Dictionary of Affect Language (DAL) (Whissell et al., 1986) scores that measure the emotional meaning of texts, and a specificity score which measures level of detail (Li and Nenkova, 2015). The full list of features is: 'hasAbsolutelyReally', 'hasContraction', 'hasI', 'hasWe', 'hasYes', 'hasNAposT' (turns

[2] A full description of the features is found here: https://s3-us-west-2.amazonaws.com/downloads.liwc.net/LIWC2015_OperatorManual.pdf

[3] A detailed explanation of these linguistic features and how they were computed is found here: http://www.cs.columbia.edu/speech/cxd/features.html

that contain words with the contraction "n't"), 'hasNo', 'hasNot', 'isJustYes', 'isJustNo', 'noYesOrNo', 'specificDenial', 'thirdPersonPronouns', 'hasFalseStart', 'hasFilledPause', 'numFilledPauses', 'hasCuePhrase', 'numCuePhrases', 'hasHedgePhrase', 'numHedgePhrases', 'hasLaugh', 'complexity', 'numLaugh', 'DAL-wc', 'DAL-pleasant', 'DAL-activate', 'DAL-imagery', 'specScores' (specificity score).

Response Length Previous work has found that response length, in seconds, is shorter in deceptive speech, and that the difference in number of words in a segment of speech is insignificant between deceptive and truthful speech (DePaulo et al., 2003). For our question-level analysis, we used four different measures for response length: the total number of seconds of an interviewee response-segment, the total number of words in an interviewee response-segment, the average response time of a turn in an interviewee response-segment, and the average number of words per turn in an interviewee response-segment.

Individual Traits We analyzed gender and native language of the speakers to determine if these traits were related to ability to deceive and to detect deception. We also analyzed linguistic cues to deception across gender and native language, and used gender and native language information in our classification experiments. All speakers were either male or female, and their native language was either Standard American English or Mandarin Chinese. In addition, we used the NEO-FFI (5 factor) personality inventory scores as features in classification experiments, but not for the statistical analysis in this paper.

Follow-up Questions Follow-up questions are questions that an interviewer asks after they ask a question from the original prescribed set of questions. We hypothesized that if an interviewer asked more follow-up questions, they were more likely to identify deceptive responses, because asking follow-up questions indicated interviewer doubt of the interviewee's truthfulness. For each interviewee response-segment, we counted the number of follow-up questions interviewees were asked by the interviewer.

5 Analysis

In order to analyze the differences between deceptive and truthful speech, we extracted the above features from each question response-segment, and calculated a series of paired t-tests between the features of truthful speech and deceptive speech. All tests for significance correct for family-wise Type I error by controlling the false discovery rate (FDR) at $\alpha = 0.05$. The k^{th} smallest p value is considered significant if it is less than $\frac{k*\alpha}{n}$.

5.1 Interviewee Responses

Table 1 shows the features that were statistically significant indicators of truth and deception in interviewee response-segments consisting of multiple turns. Below, we highlight some interesting findings.

In contrast to (DePaulo et al., 2003), we found that the total duration of an interviewee response-segment was longer for deceptive speech than for truthful speech. Additionally, while (DePaulo et al., 2003) showed that the number of words in a segment of speech was not significantly different between deceptive and truthful speech, we found that deceptive response-segments had more words than truthful response-segments. Furthermore, we found that longer average response time per turn and more words per sentence were significant indicators of deception. These results show that when interviewees are trying to deceive, not only is their aggregate response longer in duration and number of words, but their individual responses to each follow-up question are also longer. Consistent with (DePaulo et al., 2003), we found that more filled pauses in an interviewee response-segment was a significant indicator of deception. Deceivers are hypothesized to experience an increase in cognitive load (Vrij et al., 1996), and this can result in difficulties in speech planning, which can be signaled by filled pauses. Although (Benus et al., 2006) found that, in general, the use of pauses correlates more with truthful than with deceptive speech, we found that filled pauses such as "um" were correlated with deceptive speech. The LIWC *cogproc* (cognitive processes) dimension, which includes words such as "cause", "know", "ought" was significantly more frequent in truthful speech, also supporting the theory that cognitive load is increased while practicing deception.

We found that increased $DALimagery$ scores, which compute words often used in speech to create vivid descriptions, were indicators of deception. We also found that the LIWC language summary variables of $authenticity$ and $adjectives$

Feature	Deception	Truth	Neutral
Lexical	DAL.activate, DAL.imagery, DAL.pleasant noYesOrNo, numCuePhrase, numFilledPauses, numHedgePhrases specScores, thirdPersonPronouns	isJustNo	complexity, DAL.wc isJustYes numLaugh specificDenial
LIWC	achieve, adj, adverb, affiliation analytic, article, authentic cause, clout, compare, conj dash, discrep, drives, family feel, focusfuture, focuspast, friend health, interrog, ipron, male motion, percept, ppron, prep pronoun, power, relativ, reward shehe, social, space, swear verb, WC, we, WPS, you	certain, dic function, negate, netspeak	affect, apostro, assent auxverb, body, cogproc colon, comma, death differ, female, filler i, ingest, insight, leisure posemo, quant, quote relig, sad, see sixltr, they, tone, work
Response length	num words, response length avg response length, avg num words		
Followup	num turns		

Table 1: Statistically significant indicators of truth and deception in interviewee response-segments consisting of multiple turns related to a single question.

Feature	Deception	Truth	Neutral
Lexical	DAL.imagery, DAL.pleasant numCuePhrases, numFilledPauses numHedgePhrases, specificDenial specScores, thirdPersonPronoun	DAL.actvate isJustNo	complexity, DAL.wc isJustYes, noYesOrNo numLaugh
LIWC	adverb, article,authentic, body conj, focuspast, interrog, ipron prep, pronoun, WC, WPS	negate	apostro, bio, cause certain, clout, cogproc, compare discrep, focusfuture, function insight, money, motion negemo, nonflu, number posemo, ppron, relative
Response length	num words response length avg num words avg response length		
Followup	num turns		

Table 2: Statistically significant indicators of perceived truth and deception in interviewer judgments of interviewee responses.

were indicators of deception: in an effort to sound more truthful and authentic, interviewees may have provided a level of detail that is uncharacteristic of truthful speech. Similarly, the $specificity$ metric was indicative of deception: deceptive responses contained more detailed language. Words in the LIWC $clout$ category - a category describing words that indicate power of influence - were more prevalent in deceptive responses, suggesting that subjects sounded more confident while lying. $Interrogatives$ were an indicator of deception. In the context of the interviewer-interviewee paradigm, these are interviewee questions to the interviewer. Perhaps this was a technique used to stall so that they had more time to develop an answer (e.g. "Can you repeat the question?"), or to deflect the interviewer's attention from their deception and put the interviewer on the spot. We observed that $hedge$ words and phrases, which speakers use to distance themselves from a proposition, were more frequent in deceptive speech. This is consistent with Statement Analysis (Adams, 1996), which posits that hedge words are used in deceptive statements to intentionally create vagueness that obscures facts. Consistent with this finding, $certainty$ in language (words such as "always" or "never") was a strong indicator of truthfulness.

It is also interesting to note the features that were not significant indicators of truth or decep-

tion. For example, there was no significant difference in laughter frequency or apostrophes (used for contractions in this corpus) between truthful and deceptive responses.

When we compared indicators of truth vs. deception across multiple turns to indicators of truth vs. deception in just the first turns of interviewee response-segments, we found that, generally, indicators in first turns are a subset of indicators across multiple turns. In some cases there were interesting differences. For example, although $tone$ (emotional tone - higher numbers indicate more positive, and lower indicate negative) was not a significant indicator of deception for the entire interviewee response-segment, negative tone was a moderate indicator of deception in first turns. This suggests that the tone of interviewees, when they have just started their lie, is different from when they are given the opportunity to expand on that lie. The findings from our analysis of first turns suggest that there might be enough information in the first response alone to distinguish between deceptive and truthful speech; we test this in our classification experiments in Section 6.

5.2 Interviewer Judgments of Deception

In addition to analyzing the linguistic differences between truthful and deceptive speech, we were interested in studying the characteristics of speech that is believed or disbelieved. Since the CXD corpus includes interviewer judgments of deception for each question asked, we have the unique opportunity to study human perception of deception on a large scale. Table 2 shows the features that were statistically significant indicators of truth and deception in interviewee responses - consisting of multiple turns - that were perceived as true or false by interviewers. Here we highlight some interesting findings. There were many features that were prevalent in speech that interviewers perceived as deceptive, which were in fact cues to deception. For example, speech containing more words in a response-segment and more words per sentence was generally perceived as deceptive by interviewers, and indeed, this perception was correct. Disbelieved answers had a greater frequency of filled pauses and hedge words, and greater specificity, all of which were increased in deceptive speech.

There were also several features that were indicators of deception, but were not found in higher rates in statements that were perceived as false. For example, the LIWC dimensions $clout$ and $certain$ were not significantly different in believed vs. disbelieved interviewee responses, but $clout$ was increased in deceptive speech and $certain$ language was increased in truthful speech. There were also features that were significantly different between believed and disbelieved statements, but were not indicators of deception. For example, statements that were perceived as false by interviewers had a greater proportion of $specificDenials$ (e.g. "I did not") than those that were perceived as true; this was not a valid cue to deception. Number of turns was increased in dialogue segments where the interviewer did not ultimately believe the interviewee response. That is, more follow up questions were asked when an interviewer did not believe their interlocutor's response, which is an intuitive behavior. When we compared indicators of speech that was perceived as deceptive across multiple turns to indicators of speech that was perceived as deceptive in just the first turns, we found that, generally, indicators in first turns are a subset of indicators across multiple turns.

On average, human accuracy at judging truth and deception in the CXD corpus was 56.75%, and accuracy at judging deceptive statements only was 47.93%. The average F1-score for humans was 46. Thus, although some cues were correctly perceived by interviewers, humans were generally poor at deception perception. Nonetheless, characterizing the nature of speech that is believed or not believed is useful for applications where we would ultimately like to synthesize speech that is trustworthy.

5.3 Gender and Native Language Differences in Deception Behavior

Having discovered many differences between deceptive and truthful language across all speakers, we were interested in analyzing differences in deceptive language across groups of speakers. Using gender and native language (English or Mandarin Chinese) as group traits, we conducted two types of analysis. First, we directly compared deception performance measures (ability to deceive as interviewee, and ability to detect deception as interviewer) between speakers with different traits, to assess the effect of individual characteristics on deception abilities. In addition, we compared the features of deceptive and truthful language in sub-

Group	Deception	Truth
Male Female	analytic, friend, interrog achieve, adverb, article authentic, cause compare discrep,family, feel focusfuture, percept, power relativ, we	posemo
English	acheve, adverb, affiliation compare, interrog, power relativ, space, swear	
Chinese	analytic, bio cause discrep, feel, health percep, (filler)	certain (informal) (netspeak)

Table 3: Gender-specific and language-specific indicators of deception and truth. We consider a result to approach significance if its uncorrected p value is less than 0.05 and indicate this with () in the table.

sets of the corpus, considering only people with a particular trait, in order to determine group-specific patterns of deceptive language. As before, tests for significance correct for family-wise Type I error by controlling the false discovery rate (FDR) at $\alpha = 0.05$. The k^{th} smallest p value is considered significant if it is less than $\frac{k*\alpha}{n}$.

5.3.1 Gender

There were no significant differences in deception ability between male and female speakers. However, there were many differences in language between male and female speakers. Further, some features were only discriminative between deception and truth for a specific gender. Table 3 shows linguistic features that were significantly different between truthful and deceptive speech, but only for one gender. In some cases the feature was found in different proportions in male and females, and in other cases there was no significant difference. For example, $family$ words were indicative of deception only in female speakers, and these words were also used more frequently by female speakers than male speakers.

The LIWC category of $compare$ was also indicative of deception for females only, and this feature was generally found more frequently in female speech. Article usage was only significantly different between truthful and deceptive speech in females (more articles were found in deceptive speech), but articles were used more frequently in male speech. On the other hand, the LIWC category of $posemo$ (positive emotion) was increased in truthful speech for male speakers only, and there was no significant difference of $posemo$ frequency across gender.

5.3.2 Native Language

Interviewees were more successful at deceiving native Chinese speakers than at deceiving native English speakers ($t(170) = -2.13, p = 0.033$). This was true regardless of interviewee gender and native language, and slightly stronger for female interviewers ($t(170) = -2.22, p = 0.027$). When considering only female interviewers, interviewees were more successful at deceiving non-native speakers than native speakers, but this difference was not significant when considering only male interviewers. As with gender, there were several features that were discriminative between deception and truth for only native speakers of English, or only native speakers of Mandarin. Table 3 shows LIWC categories and their relation to deception, broken down by native language. For example, power words were found more frequently in deception statements, when considering native English speakers only. In general, power words were used more by native Mandarin speakers than by native English speakers. LIWC categories of $compare$, $relative$, and $swear$ were more prevalent in deceptive speech, only for English speakers. On the other hand, $feel$ and $perception$ dimensions were only indicators of deception for native Mandarin speakers, although there was no significant difference in the use of these word categories across native language. $Informal$ and $netspeak$ word dimensions tended to be more frequent in truthful speech for native Chinese speakers only (approaching significance), and these word categories were generally more frequent in native Mandarin speech. Finally, $filler$ words tended to be more frequent in deceptive speech (approaching significance) only for native Mandarin speakers, and these were used more frequently by native Mandarin speakers than native English speakers.

Overall, our findings suggest that deceptive behavior in general, and deceptive language in particular, are affected by a person's individual characteristics, including gender and native language. When building a deception classification system, it is important to account for this variation across speaker groups.

Features	Segmentation	Accuracy	Precision	Recall	F1-score
Human baseline	Multiple turns	56.75	56.50	40.00	46.50
LIWC	Single turn	65.75	65.79	65.74	65.72
	Multiple turns	72.78	72.84	72.74	72.74
Lexical	Single turn	66.95	66.97	66.95	66.94
	Multiple turns	70.33	70.46	70.28	70.25
LIWC+lexical	Single turn	68.35	68.36	68.35	68.35
	Multiple turns	71.66	71.77	71.60	71.58
LIWC+individual	Single turn	67.50	67.50	67.50	67.49
	Multiple turns	71.85	71.93	71.80	71.79
Lexical+individual	Single turn	69.32	69.33	69.32	69.31
	Multiple turns	69.95	70.06	69.89	69.86
LIWC+lexical+individual	Single turn	70.87	70.87	70.87	70.87
	Multiple turns	72.40	72.50	72.34	72.33

Table 4: Random Forest classification of single turn and multiple turn segmentations, using text-based features and individual traits (gender, native language, NEO-FFI personality scores).

6 Deception Classification

Motivated by our analysis showing many significant differences in the language of truthful and deceptive responses to interview questions, we trained machine learning classifiers to automatically distinguish between truthful and deceptive text, using the feature sets described in section 4. We compared classification performance for the two segmentation methods described in section 3.2: first turn and multiple turns. This allowed us to explore the role of context in automatic deception detection. When classifying interviewee response-segments, should the immediate response only be used for classification, or is inclusion of surrounding turns helpful? This has implications not only for deception classification, but for practitioners as well. Should human interviewers make use of responses to follow up questions when determining response veracity, or should the initial response receive the most consideration?

We compared the performance of 3 classification algorithms: Random Forest, Logistic Regression, and SVM (sklearn implementation). In total, there were 7,792 question segments for both single turn and multiple turns segmentations. We divided this into 66% train and 33% test, and used the same fixed test set in experiments for both segmentations in order to directly compare results. The random baseline performance is 50, since the dataset is balanced for truthful and deceptive statements. Another baseline is human performance, which is 46.0 F1 in this corpus. The Random Forest classifier was consistently the best performing, and we only report those results due to space constraints. Table 4 displays the classification performance for each feature set individually, as well as feature combinations, for both single turn and multiple turn segmentations. It also shows the human baseline performance, obtained from the interviewers' judgments of deception in the corpus, which were made after asking each question along with related follow-up questions (i.e. multiple turn segmentation).

The best performance (72.74 F1-score) was obtained using LIWC features extracted from multiple turns. This is a 22.74% absolute increase over the random baseline of 50, and a 26.74% absolute increase over the human baseline of 46. The performance of classifiers trained on multiple turns was consistently better than those trained on single turns, for all feature sets. For multiple turns, LIWC features were better than the lexical feature set, and combining lexical with LIWC features did not improve over the performance of LIWC features alone. Adding individual traits information was also not beneficial. However, when considering the first turn only, the best results (70.87 F1-score) were obtained using a combination of LIWC+lexical+individual features. Using the first turns segmentation, lexical features were slightly better than LIWC features, and interestingly, adding individual traits helped both feature sets. A combination of LIWC and lexical features was better than each on its own.

These results suggest that contextual informa-

tion, in the form of follow up questions, is beneficial for deception classification. It seems that individual traits, including gender, native language, and personality scores, are helpful in deception classification under the condition where contextual information is not available. When the contextual information is available, the the additional lexical content is more useful than individual traits.

7 Conclusions and Future Work

In this paper we presented a study of deceptive language in interview dialogues. Our analysis of linguistic characteristics of deceptive and truthful speech provides insight into the nature of deceptive language. We also analyzed the linguistic characteristics of speech that is *perceived* as deceptive and truthful, which is important for understanding the nature of trustworthy speech. We explored variation across gender and native language in linguistic cues to deception, highlighting cues that are specific to particular groups of speakers. We built classifiers that use combinations of linguistic features and individual traits to automatically identify deceptive speech. We compared the performance of using cues from the single first turn of an interviewee response-segment with using cues from the full context of multiple interviewee turns, achieving performance as high as 72.74% F1-score (about 27% better than human detection performance).

This work contributes to the critical problem of automatic deception detection, and increases our scientific understanding of deception, deception perception, and individual differences in deceptive behavior. In future work, we plan to conduct similar analysis in additional deception corpora in other domains, in order to identify consistent domain-independent deception indicators. In addition, we plan to conduct cross-corpus machine learning experiments, to evaluate the robustness of these and other feature sets in deception detection. We also would like to explore additional feature combinations, such as adding acoustic-prosodic features. Finally, we plan to conduct an empirical analysis of deception behavior across personality types.

Acknowledgments

This work was partially funded by AFOSR FA9550-11-1-0120 and by NSF DGE-11-44155. Thank you to Bingyan Hu for her assistance with feature extraction. We thank the anonymous reviewers for their helpful comments.

References

Susan H Adams. 1996. Statement analysis: What do suspects' words really reveal. *FBI L. Enforcement Bull.* 65:12.

Joan Bachenko, Eileen Fitzpatrick, and Michael Schonwetter. 2008. Verification and implementation of language-based deception indicators in civil and criminal narratives. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*. Association for Computational Linguistics, pages 41–48.

Stefan Benus, Frank Enos, Julia Hirschberg, and Elizabeth Shriberg. 2006. Pauses in deceptive speech. In *Speech Prosody*. volume 18, pages 2–5.

PT Costa and RR McCrae. 1989. Neo five-factor inventory (neo-ffi). *Odessa, FL: Psychological Assessment Resources* .

Bella M DePaulo, James J Lindsay, Brian E Malone, Laura Muhlenbruck, Kelly Charlton, and Harris Cooper. 2003. Cues to deception. *American Psychological Association, Inc.* pages 74–118.

Frank Enos. 2009. *Detecting deception in speech*. Ph.D. thesis, Citeseer.

Song Feng, Ritwik Banerjee, and Yejin Choi. 2012. Syntactic stylometry for deception detection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*. Association for Computational Linguistics, pages 171–175.

Tommaso Fornaciari and Massimo Poesio. 2013. Automatic deception detection in italian court cases. *Artificial intelligence and law* 21(3):303–340.

Sarah I Levitan, Guzhen An, Mandi Wang, Gideon Mendels, Julia Hirschberg, Michelle Levine, and Andrew Rosenberg. 2015a. Cross-cultural production and detection of deception from speech. In *Proceedings of the 2015 ACM on Workshop on Multimodal Deception Detection*. ACM, pages 1–8.

Sarah I Levitan, Guzhen An, Mandi Wang, Gideon Mendels, Julia Hirschberg, Michelle Levine, and Andrew Rosenberg. 2015b. Cross-cultural production and detection of deception from speech. In *Proceedings of the 2015 ACM on Workshop on Multimodal Deception Detection*. ACM, pages 1–8.

Junyi Jessy Li and Ani Nenkova. 2015. Fast and accurate prediction of sentence specificity. In *Proceedings of the Twenty-Ninth Conference on Artificial Intelligence (AAAI)*. pages 2281–2287.

Angel S Maredia, Kara Schechtman, Sarah I Levitan, and Julia Hirschberg. 2017. Comparing approaches for automatic question identification. SEM.

Gideon Mendels, Sarah Ita Levitan, Kai-Zhan Lee, and Julia Hirschberg. 2017. Hybrid acoustic-lexical deep learning approach for deception detection. *Proc. Interspeech 2017* pages 1472–1476.

Rada Mihalcea and Carlo Strapparava. 2009. The lie detector: Explorations in the automatic recognition of deceptive language. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*. Association for Computational Linguistics, pages 309–312.

Matthew L Newman, James W Pennebaker, Diane S Berry, and Jane M Richards. 2003. Lying words: Predicting deception from linguistic styles. *Personality and social psychology bulletin* 29(5):665–675.

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pages 309–319.

James W Pennebaker, Martha E Francis, and Roger J Booth. 2001. Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates* 71:2001.

James W Pennebaker and Laura A King. 1999. Linguistic styles: language use as an individual difference. *Journal of personality and social psychology* 77(6):1296.

James W Pennebaker, Tracy J Mayne, and Martha E Francis. 1997. Linguistic predictors of adaptive bereavement. *Journal of personality and social psychology* 72(4):863.

Verónica Pérez-Rosas, Mohamed Abouelenien, Rada Mihalcea, and Mihai Burzo. 2015. Deception detection using real-life trial data. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*. ACM, pages 59–66.

Verónica Pérez-Rosas and Rada Mihalcea. 2015. Experiments in open domain deception detection. In *Proceedings of EMNLP 2015*. ACL, pages 1120–1125.

Patricia Rockwell, David B Buller, and Judee K Burgoon. 1997. The voice of deceit: Refining and expanding vocal cues to deception. *Communication Research Reports* 14(4):451–459.

Aldert Vrij, Gun R Semin, and Ray Bull. 1996. Insight into behavior displayed during deception. *Human Communication Research* 22(4):544–562.

Cynthia Whissell, Michael Fournier, René Pelland, Deborah Weir, and Katherine Makarec. 1986. A dictionary of affect in language: Iv. reliability, validity, and applications. *Perceptual and Motor Skills* 62(3):875–888.

Maria Yancheva and Frank Rudzicz. 2013. Automatic detection of deception in child-produced speech using syntactic complexity features. In *ACL (1)*. pages 944–953.

Unified Pragmatic Models for Generating and Following Instructions

Daniel Fried Jacob Andreas Dan Klein
Computer Science Division
University of California, Berkeley
{dfried,jda,klein}@cs.berkeley.edu

Abstract

We show that explicit pragmatic inference aids in correctly generating and following natural language instructions for complex, sequential tasks. Our pragmatics-enabled models reason about why speakers produce certain instructions, and about how listeners will react upon hearing them. Like previous pragmatic models, we use learned base listener and speaker models to build a *pragmatic speaker* that uses the base listener to simulate the interpretation of candidate descriptions, and a *pragmatic listener* that reasons counterfactually about alternative descriptions. We extend these models to tasks with sequential structure. Evaluation of language generation and interpretation shows that pragmatic inference improves state-of-the-art listener models (at correctly interpreting human instructions) and speaker models (at producing instructions correctly interpreted by humans) in diverse settings.

1 Introduction

How should speakers and listeners reason about each other when they communicate? A core insight of computational pragmatics is that speaker and listener agents operate within a cooperative game-theoretic context, and that each agent benefits from reasoning about others' intents and actions within that context. Pragmatic inference has been studied by a long line of work in linguistics, natural language processing, and cognitive science. In this paper, we present a technique for layering explicit pragmatic inference on top of models for complex, sequential instruction-following and instruction-generation tasks. We investigate a range of current data sets for both tasks, showing that pragmatic behavior arises naturally from this inference procedure, and gives rise to state-of-the-art results in a variety of domains.

Consider the example shown in Figure 1a, in which a speaker agent must describe a route to a target position in a hallway. A conventional learned instruction-generating model produces a truthful description of the route (*walk forward four times*). But the pragmatic speaker in this paper, which is capable of reasoning about the listener, chooses to also include additional information (*the intersection with the bare concrete hall*), to reduce potential ambiguity and increase the odds that the listener reaches the correct destination.

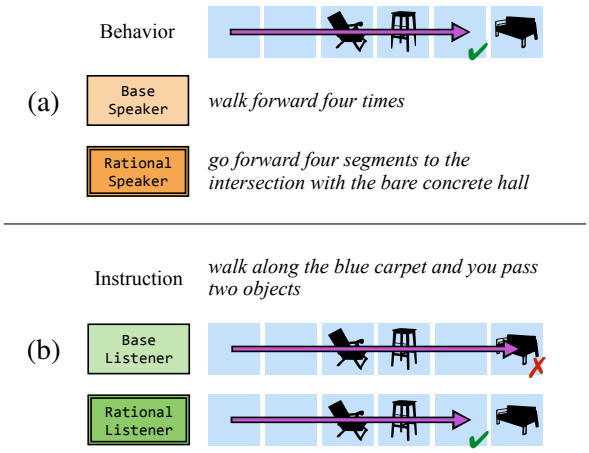


Figure 1: Real samples for the SAIL navigation environments, comparing *base* models, without explicit pragmatic inference, to the *rational* pragmatic inference procedure. (a) The rational speaker, which reasons about listener behavior, generates instructions which in this case are more robust to uncertainty about the listener's initial orientation. (b) The base listener moves to an unintended position (even though it correctly *passes two objects*). The rational listener, which reasons about the speaker, infers that a route ending at the sofa would have been described differently, and stops earlier.

This same reasoning procedure also allows a listener agent to overcome ambiguity in instructions by reasoning counterfactually about the speaker (Figure 1b). Given the command *walk along the blue carpet and you pass two objects*, a conven-

Proceedings of NAACL-HLT 2018, pages 1951–1963
New Orleans, Louisiana, June 1 - 6, 2018.

tional learned instruction-following model is willing to consider all paths that pass two objects, and ultimately arrives at an unintended final position. But a pragmatic listener that reasons about the speaker can infer that the long path would have been more easily described as *go to the sofa*, and thus that the shorter path is probably intended. In these two examples, which are produced by the system we describe in this paper, a unified reasoning process (choose the output sequence which is most preferred by an embedded model of the other agent) produces pragmatic behavior for both speakers and listeners.

The application of models with explicit pragmatic reasoning abilities has so far been largely restricted to simple reference games, in which the listener's only task is to select the right item from among a small set of candidate referents given a single short utterance from the speaker. But as the example shows, there are real-world instruction following and generation tasks with rich action spaces that might also benefit from pragmatic modeling. Moreover, approaches that learn to map directly between human-annotated instructions and action sequences are ultimately limited by the effectiveness of the humans themselves. The promise of pragmatic modeling is that we can use these same annotations to build a model with a different (and perhaps even better) mechanism for interpreting and generating instructions.

The primary contribution of this work is to show how existing models of pragmatic reasoning can be extended to support instruction following and generation for challenging, multi-step, interactive tasks. Our experimental evaluation focuses on four instruction-following domains which have been studied using both semantic parsers and attentional neural models. We investigate the interrelated tasks of instruction following and instruction generation, and show that incorporating an explicit model of pragmatics helps in both cases. Reasoning about the human listener allows a speaker model to produce instructions that are easier for humans to interpret correctly in all domains (with absolute gains in accuracy ranging from 12% to 46%). Similarly, reasoning about the human speaker improves the accuracy of the listener models in interpreting instructions in most domains (with gains in accuracy of up to 10%). In all cases, the resulting systems are competitive with, and in many cases exceed, results from past state-of-the-art systems for these tasks.[1]

2 Problem Formulation

Consider the instruction following and instruction generation tasks shown in Figure 1, where an agent must produce or interpret instructions about a structured world context (e.g. *walk along the blue carpet and you pass two objects*).

In the **instruction following** task, a listener agent begins in a world state (in Figure 1 an initial map location and orientation). The agent is then tasked with following a sequence of direction sentences $d_1 \dots d_K$ produced by humans. At each time t the agent receives a percept y_t, which is a feature-based representation of the current world state, and chooses an action a_t (e.g. move forward, or turn). The agent succeeds if it is able to reach the correct final state described by the directions.

In the **instruction generation** task, the agent receives a sequence of actions $a_1, \cdots a_T$ along with the world state $y_1, \cdots y_T$ at each action, and must generate a sequence of direction sentences $d_1, \dots d_K$ describing the actions. The agent succeeds if a human listener is able to correctly follow those directions to the intended final state.

We evaluate models for both tasks in four domains. The first domain is the SAIL corpus of virtual environments and navigational directions (MacMahon et al., 2006; Chen and Mooney, 2011), where an agent navigates through a two-dimensional grid of hallways with patterned walls and floors and a discrete set of objects (Figure 1 shows a portion of one of these hallways).

In the three SCONE domains (Long et al., 2016), the world contains a number of objects with various properties, such as colored beakers which an agent can combine, drain, and mix. Instructions describe how these objects should be manipulated. These domains were designed to elicit instructions with a variety of context-dependent language phenomena, including ellipsis and coreference (Long et al., 2016) which we might expect a model of pragmatics to help resolve (Potts, 2011).

3 Related Work

The approach in this paper builds upon long lines of work in pragmatic modeling, instruction following, and instruction generation.

[1] Source code is available at http://github.com/dpfried/pragmatic-instructions

Pragmatics Our approach to pragmatics (Grice, 1975) belongs to a general category of rational speech acts models (Frank and Goodman, 2012), in which the interaction between speakers and listeners is modeled as a probabilistic process with Bayesian actors (Goodman and Stuhlmüller, 2013). Alternative formulations (e.g. with best-response rather than probabilistic dynamics) are also possible (Golland et al., 2010). Inference in these models is challenging even when the space of listener actions is extremely simple (Smith et al., 2013), and one of our goals in the present work is to show how this inference problem can be solved even in much richer action spaces than previously considered in computational pragmatics. This family of pragmatic models captures a number of important linguistic phenomena, especially those involving conversational implicature (Monroe and Potts, 2015); we note that many other topics studied under the broad heading of "pragmatics," including presupposition and indexicality, require different machinery.

Williams et al. (2015) use pragmatic reasoning with weighted inference rules to resolve ambiguity and generate clarification requests in a human-robot dialog task. Other recent work on pragmatic models focuses on the referring expression generation or "contrastive captioning" task introduced by Kazemzadeh et al. (2014). In this family are approaches that model the listener at training time (Mao et al., 2016), at evaluation time (Andreas and Klein, 2016; Monroe et al., 2017; Vedantam et al., 2017; Su et al., 2017) or both (Yu et al., 2017b; Luo and Shakhnarovich, 2017).

Other conditional sequence rescoring models that are structurally similar but motivated by concerns other than pragmatics include Li et al. (2016) and Yu et al. (2017a). Lewis et al. (2017) perform a similar inference procedure for a competitive negotiation task. The language learning model of Wang et al. (2016) also features a structured output space and uses pragmatics to improve online predictions for a semantic parsing model. Our approach in this paper performs both generation and interpretation, and investigates both structured and unstructured output representations.

Instruction following Work on instruction following tasks includes models that parse commands into structured representations processed by a rich execution model (Tellex et al., 2011; Chen, 2012; Artzi and Zettlemoyer, 2013; Guu et al., 2017), and models that map directly from instructions to a policy over primitive actions (Branavan et al., 2009), possibly mediated by an intermediate alignment or attention variable (Andreas and Klein, 2015; Mei et al., 2016). We use a model similar to Mei et al. (2016) as our base listener in this paper, evaluating on the SAIL navigation task (MacMahon et al., 2006) as they did, as well as the SCONE context-dependent execution domains (Long et al., 2016).

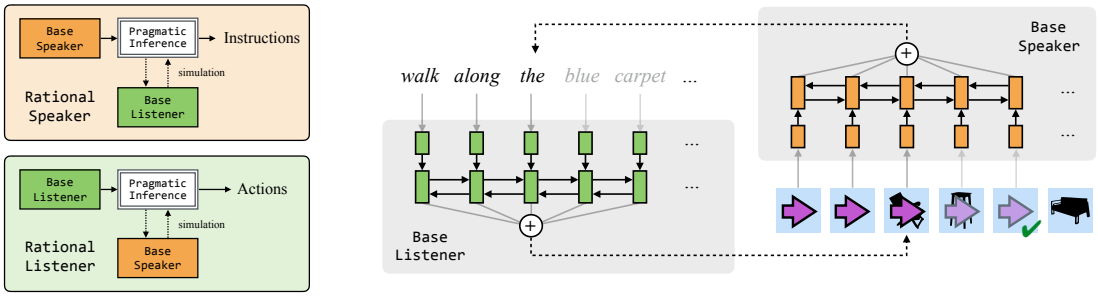
Instruction generation Previous work has also investigated the instruction generation task, in particular for navigational directions. The GIVE shared tasks (Byron et al., 2009; Koller et al., 2010; Striegnitz et al., 2011) have produced a large number of interactive direction-giving systems, both rule-based and learned. The work most immediately related to the generation task in this paper is that of Daniele et al. (2017), which also focuses on the SAIL dataset but requires substantial additional structured annotation for training, while both our base and pragmatic speaker models learn directly from strings and action sequences.

Older work has studied the properties of effective human strategies for generating navigational directions (Anderson et al., 1991). Instructions of this kind can be used to extract templates for generation (Look, 2008; Dale et al., 2005), while here we focus on the more challenging problem of learning to generate new instructions from scratch. Like our pragmatic speaker model, Goeddel and Olson (2012) also reason about listener behavior when generating navigational instructions, but rely on rule-based models for interpretation.

4 Pragmatic inference procedure

As a foundation for pragmatic inference, we assume that we have *base* listener and speaker models to map directions to actions and vice-versa. (Our notation for referring to models is adapted from Bergen et al. (2016).) The base listener, L_0, produces a probability distribution over sequences of actions, conditioned on a representation of the directions and environment as seen before each action: $P_{L_0}(a_{1:T}|d_{1:K}, y_{1:T})$. Similarly, the base speaker, S_0, defines a distribution over possible descriptions conditioned on a representation of the actions and environment: $P_{S_0}(d_{1:K}|a_{1:T}, y_{1:T})$.

Our pragmatic inference procedure requires these base models to produce candidate outputs from a given input (actions from descriptions, for



(a) (b)

Figure 2: (a) Rational pragmatic models embed base listeners and speakers. Potential candidate sequences are drawn from one base model, and then the other scores each candidate to simulate whether it produces the desired pragmatic behavior. (b) The base listener and speaker are neural sequence-to-sequence models which are largely symmetric to each other. Each produces a representation of its input sequence (a description, for the listener; actions with associated environmental percepts, for the listener) using an LSTM encoder. The output sequence is generated by an LSTM decoder attending to the input.

the listener; descriptions from actions, for the speaker), and calculate the probability of a fixed output given an input, but is otherwise agnostic to the form of the models.

We use standard sequence-to-sequence models with attention for both the base listener and speaker (described in Section 5). Our models use segmented action sequences, with one segment (sub-sequence of actions) aligned with each description sentence d_j, for all $j \in \{1 \dots K\}$. This segmentation is either given as part of the training and testing data (in the instruction following task for the SAIL domain, and in both tasks for the SCONE domain, where each sentence corresponds to a single action), or is predicted by a separate segmentation model (in the generation task for the SAIL domain), see Section 5.

4.1 Models

Using these base models as self-contained modules, we derive a *rational speaker* and *rational listener* that perform inference using embedded instances of these base models (Figure 2a). When describing an action sequence, a rational speaker S_1 chooses a description that has a high chance of causing the listener modeled by L_0 to follow the given actions:

$$S_1(a_{1:T}) = \operatorname*{argmax}_{d_{1:K}} P_{L_0}(a_{1:T}|d_{1:K}, y_{1:T}) \quad (1)$$

(noting that, in all settings we explore here, the percepts $y_{1:T}$ are completely determined by the actions $a_{1:T}$). Conversely, a rational listener L_1 follows a description by choosing an action sequence which has high probability of having caused the speaker, modeled by S_0, to produce the description:

$$L_1(d_{1:K}) = \operatorname*{argmax}_{a_{1:T}} P_{S_0}(d_{1:K}|a_{1:T}, y_{1:T}) \quad (2)$$

These optimization problems are intractable to solve for general base listener and speaker agents, including the sequence-to-sequence models we use, as they involve choosing an input (from a combinatorially large space of possible sequences) to maximize the probability of a fixed output sequence. We instead follow a simple approximate inference procedure, detailed in Section 4.2.

We consider also incorporating the scores of the base model used to produce the candidates. For the case of the speaker, we define a *combined rational speaker*, denoted $S_0 \cdot S_1$, that selects the candidate that maximizes a weighted product of probabilities under both the base listener and the base speaker:

$$\operatorname*{argmax}_{d_{1:K}} P_{L_0}(a_{1:T}|d_{1:K}, y_{1:T})^{\lambda} \times P_{S_0}(d_{1:K}|a_{1:T}, y_{1:T})^{1-\lambda} \quad (3)$$

for a fixed interpolation hyperparameter $\lambda \in [0, 1]$. There are several motivations for this combination with the base speaker score. First, as argued by Monroe et al. (2017), we would expect varying degrees of base and reasoned interpretation in human speech acts. Second, we want the descriptions produced by the model to be fluent descriptions of the actions. Since the base models are trained discriminatively, maximizing the probability of an output sequence for a fixed input sequence, their scoring behaviors for fixed outputs paired with inputs dissimilar to those seen in the training set may be

poorly calibrated (for example when conditioning on ungrammatical descriptions). Incorporating the scores of the base model used to produce the candidates aims to prevent this behavior.

To define rational listeners, we use the symmetric formulation: first, draw candidate action sequences from L_0. For L_1, choose the actions that achieve the highest probability under S_0; and for the combination model $L_0 \cdot L_1$ choose the actions with the highest weighted combination of S_0 and L_0 (paralleling equation 3).

4.2 Inference

As in past work (Smith et al., 2013; Andreas and Klein, 2016; Monroe et al., 2017), we approximate the optimization problems in equations 1, 2, and 3: use the base models to generate candidates, and rescore them to find ones that are likely to produce the desired behavior.

In the case of the rational speaker S_1, we use the base speaker S_0 to produce a set of n candidate descriptions $w^{(1)}_{1:K_1} \dots w^{(n)}_{1:K_n}$ for the sequences $a_{1:T}, y_{1:T}$, using beam search. We then find the score of each description under P_{L_0} (using it as the input sequence for the observed output actions we want the rational speaker to describe), or a weighted combination of P_{L_0} and the original candidate score P_{S_0}, and choose the description $w^{(j)}_{1:K_j}$ with the largest score, approximately solving the maximizations in equations 1 or 3, respectively. We perform a symmetric procedure for the rational listener: produce action sequence candidates from the base listener, and rescore them using the base speaker.[2]

As the rational speaker must produce long output sequences (with multiple sentences), we interleave the speaker and listener in inference, determining each output sentence sequentially. From a list of candidate direction sentences from the base speaker for the current subsequence of actions, we choose the top-scoring direction under the listener model (which may also condition on the directions which have been output previously), and then move on to the next subsequence of actions.[3]

5 Base model details

Given this framework, all that remains is to describe the base models L_0 and S_0. We implement these as sequence-to-sequence models that map directions to actions (for the listener) or actions to directions (for the speaker), additionally conditioning on the world state at each timestep.

5.1 Base listener

Our base listener model, L_0, predicts action sequences conditioned on an encoded representation of the directions and the current world state. In the SAIL domain, this is the model of Mei et al. (2016) (illustrated in green in Figure 2b for a single sentence and its associated actions), see "domain specifics" below.

Encoder Each direction sentence is encoded separately with a bidirectional LSTM (Hochreiter and Schmidhuber, 1997); the LSTM's hidden states are reset for each sentence. We obtain a representation h^e_k for the kth word in the current sentence by concatenating an embedding for the word with its forward and backward LSTM outputs.

Decoder We generate actions incrementally using an LSTM decoder with monotonic alignment between the direction sentences and subsequences of actions; at each timestep the decoder predicts the next action for the current sentence $w_{1:M}$ (including choosing to shift to the next sentence). The decoder takes as input at timestep t the current world state, y_t and a representation z_t of the current sentence, updates the decoder state h^d, and outputs a distribution over possible actions:

$$h^d_t = \text{LSTM}_d(h^d_{t-1}, [W_y y_t, z_t])$$
$$q_t = W_o(W_y y_t + W_h h^d_t + W_z z_t)$$
$$p(a_t \mid a_{1:t-1}, y_{1:t}, w_{1:M}) \propto \exp(q_t)$$

where all weight matrices W are learned parameters. The sentence representation z_t is produced using an attention mechanism (Bahdanau et al., 2015) over the representation vectors $h^e_1 \dots h^e_M$

[2] We use ensembles of models for the base listener and speaker (subsection 5.3), and to obtain candidates that are high-scoring under the combination of models in the ensemble, we perform standard beam search using all models in lock-step. At every timestep of the beam search, each possible extension of an output sequence is scored using the product of the extension's conditional probabilities across all models in the ensemble.

[3] We also experimented with sampling from the base models to produce these candidate lists, as was done in previous work (Andreas and Klein, 2016; Monroe et al., 2017). In early experiments, however, we found better performance with beam search in the rational models for all tasks.

for words in the current sentence:

$$\alpha_{t,k} \propto \exp(v \cdot \tanh(W_d h^d_{t-1} + W_e h^e_k))$$
$$z_t = \sum_{k=1}^{M} \alpha_{t,k} h^e_k$$

where the attention weights $\alpha_{t,k}$ are normalized to sum to one across positions k in the input, and weight matrices W and vector v are learned.

Domain specifics For SAIL, we use the alignments between sentences and route segments annotated by Chen and Mooney (2011), which were also used in previous work (Artzi and Zettlemoyer, 2013; Artzi et al., 2014; Mei et al., 2016). Following Mei et al. (2016), we reset the decoder's hidden state for each sentence.

In the SCONE domains, which have a larger space of possible outputs than SAIL, we extend the decoder by: (i) decomposing each action into an action type and arguments for it, (ii) using separate attention mechanisms for types and arguments and (iii) using state-dependent action embeddings. See Appendix A in the supplemental material for details. The SCONE domains are constructed so that each sentence corresponds to a single (non-decomposed) action; this provides our segmentation of the action sequence.

5.2 Base speaker

While previous work (Daniele et al., 2017) has relied on more structured approaches, we construct our base speaker model S_0 using largely the same sequence-to-sequence machinery as above. S_0 (illustrated in orange in Figure 2b) encodes a sequence of actions and world states, and then uses a decoder to output a description.

Encoder We encode the sequence of vector embeddings for the actions a_t and world states y_t using a bidirectional LSTM. Similar to the base listener's encoder, we then obtain a representation h^e_t for timestep t by concatenating a_t and y_t with the LSTM outputs at that position.

Decoder As in the listener, we use an LSTM decoder with monotonic alignment between direction sentences and subsequences of actions, and attention over the subsequences of actions. The decoder takes as input at position k an embedding for the previously generated word w_{k-1} and a representation z_k of the current subsequence of actions and world states, and produces a distribution over words (including ending the description for the current subsequence and advancing to the next). The decoder's output distribution is produced by:

$$h^d_k = \text{LSTM}_d(h^d_{k-1}, [w_{k-1}, z_k])$$
$$q_k = W_h h^d_k + W_z z_k$$
$$p(w_k \mid w_{1:k-1}, a_{1:T}, y_{1:T}) \propto \exp(q_k)$$

where all weight matrices W are learned parameters.[4] As in the base listener, the input representation z_k is produced by attending to the vectors $h^e_1 \ldots h^e_T$ encoding the input sequence (here, encoding the subsequence of actions and world states to be described):

$$\alpha_{k,t} \propto \exp(v \cdot \tanh(W_d h^d_{k-1} + W_e h^e_t))$$
$$z_k = \sum_{t=1}^{T} \alpha_{k,t}\ h^e_t$$

The decoder's LSTM state is reset at the beginning of each sentence.

Domain specifics In SAIL, for comparison to the generation system of Daniele et al. (2017) which did not use segmented routes, we train a route segmenter for use at test time. We also represent routes using a collapsed representation of action sequences. In the SCONE domains, we (i) use the same context-dependent action embeddings used in the listener, and (ii) don't require an attention mechanism, since only a single action is used to produce a given sentence within the sequence of direction sentences. See Appendix A for more details.

5.3 Training

The base listener and speaker models are trained independently to maximize the conditional likelihoods of the actions–directions pairs in the training sets. See Appendix A for details on the optimization, LSTM variant, and hyperparameters.

We use ensembles for the base listener L_0 and base speaker S_0, where each ensemble consists of 10 models trained from separate random parameter initializations. This follows the experimental setup of Mei et al. (2016) for the SAIL base listener.

[4] All parameters are distinct from those used in the base listener; the listener and speaker are trained separately.

listener	Single-sentence Rel	Single-sentence Abs	Multi-sentence Rel	Multi-sentence Abs
past work	69.98 (MBW)	65.28 (AZ)	26.07 (MBW)	35.44 (ADP)
L_0	68.40	59.62	24.79	13.53
$L_0 \cdot L_1$	**71.64**	64.38	**34.05**	24.50
accuracy gain	+3.24	+4.76	+9.26	+10.97

Table 1: Instruction-following results on the SAIL dataset. The table shows cross-validation test accuracy for the base listener (L_0) and pragmatic listeners ($L_0 \cdot L_1$), along with the gain given by pragmatics. We report results for the single- and multi-sentence conditions, under the relative and absolute starting conditions[5], comparing to the best-performing prior work by Artzi and Zettlemoyer (2013) (AZ), Artzi et al. (2014) (ADP), and Mei et al. (2016) (MBW). Bold numbers show new state-of-the-art results.

listener	Alchemy	Scene	Tangrams
GPLL	52.9	46.2	37.3
L_0	69.7	70.9	69.6
$L_0 \cdot L_1$	72.0	72.7	69.6
accuracy gain	+2.3	+1.8	+0.0

Table 2: Instruction-following results in the SCONE domains. The table shows accuracy on the test set. For reference, we also show prior results from Guu et al. (2017) (GPLL), although our models use more supervision at training time.

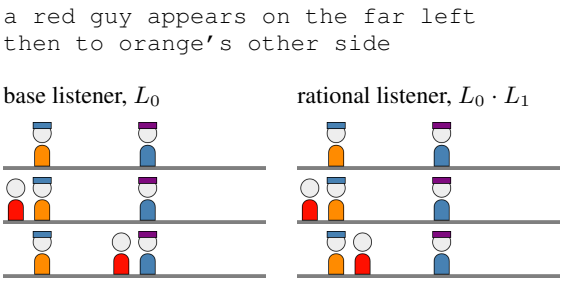


Figure 3: Action traces produced for a partial instruction sequence (two instructions out of five) in the Scene domain. The base listener moves the red figure to a position that is a marginal, but valid, interpretation of the directions. The rational listener correctly produces the action sequence the directions were intended to describe.

6 Experiments

We evaluate speaker and listener agents on both the instruction following and instruction generation tasks in the SAIL domain and three SCONE domains (Section 2). For all domains, we compare the rational listener and speaker against the base listener and speaker, as well as against past state-of-the-art results for each task and domain. Finally, we examine pragmatic inference from a model combination perspective, comparing the pragmatic reranking procedure to ensembles of a larger number of base speakers or listeners.

For all experiments, we use beam search both to generate candidate lists for the rational systems (section 4.2) and to generate the base model's output. We fix the beam size n to be the same in both the base and rational systems, using $n = 20$ for the speakers and $n = 40$ for the listeners. We tune the weight λ in the combined rational agents ($L_0 \cdot L_1$ or $S_0 \cdot S_1$) to maximize accuracy (for listener models) or BLEU (for speaker models) on each domain's development data.

6.1 Instruction following

We evaluate our listener models by their accuracy in carrying out human instructions: whether the systems were able to reach the final world state which the human was tasked with guiding them to.

SAIL We follow standard cross-validation evaluation for the instruction following task on the SAIL dataset (Artzi and Zettlemoyer, 2013; Artzi et al., 2014; Mei et al., 2016).[5] Table 1 shows improvements over the base listener L_0 when using the rational listener $L_0 \cdot L_1$ in the single- and multi-sentence settings. We also report the best accuracies from past work. We see that the largest relative gains come in the multi-sentence setting, where handling ambiguity is potentially more important to avoid compounding errors. The rational model improves on the published results of Mei et al. (2016), and while it is still below the systems of Artzi and Zettlemoyer (2013) and Artzi et al. (2014), which use additional supervision in the form of hand-annotated seed lexicons and logical domain representations, it approaches their results in the single-sentence setting.

SCONE In the SCONE domains, past work has trained listener models with weak supervision

[5]Past work has differed in the handling of undetermined orientations in the routes, which occur in the first state for multi-sentence routes and the first segment of their corresponding single-sentence routes. For comparison to both types of past work, we train and evaluate listeners in two settings: *Abs*, which sets these undetermined starting orientations to be a fixed absolute orientation, and *Rel*, where an undetermined starting orientation is set to be a 90 degree rotation from the next state in the true route.

speaker	SAIL	Alchemy	Scene	Tangrams
DBW	70.9	—	—	—
S_0	62.8	29.3	31.3	60.0
$S_0 \cdot S_1$	**75.2**	**75.3**	**69.3**	**88.0**
accuracy gain	+12.4	+46.0	+38.0	+28.0
human-generated	73.2	83.3	78.0	66.0

Table 3: Instruction generation results. We report the accuracies of human evaluators at following the outputs of the speaker systems (as well as other humans) on 50-instance samples from the SAIL dataset and SCONE domains. DBW is the system of Daniele et al. (2017). Bold numbers are new state-of-the-art results.

speaker	SAIL	Alchemy	Scene	Tangrams
DBW	11.00	—	—	—
S_0	12.04	19.34	18.09	21.75
$S_0 \cdot S_1$	10.78	18.70	27.15	23.03
BLEU gain	-1.26	-0.64	+9.06	+1.28
accuracy gain (from Table 3)	+12.4	+46.0	+38.0	+28.0

Table 4: Gains in how easy the directions are to follow are not always associated with a gain in BLEU. This table shows corpus-level 4-gram BLEU comparing outputs of the speaker systems to human-produced directions on the SAIL dataset and SCONE domains, compared to gains in accuracy when asking humans to carry out a sample of the systems' directions (see Table 3).

(with no intermediate actions between start and end world states) on a subset of the full SCONE training data. We use the full training set, and to use a model and training procedure consistent with the SAIL setting, train listener and speaker models using the intermediate actions as supervision as well.[6] The evaluation method and test data are the same as in past work on SCONE: models are provided with an initial world state and a sequence of 5 instructions to carry out, and are evaluated on their accuracy in reaching the intended final world state.

Results are reported in Table 2. We see gains from the rational system $L_0 \cdot L_1$ in both the Alchemy and Scene domains. The pragmatic inference procedure allows correcting errors or overly-literal interpretations from the base listener. An example is shown in Figure 3. The base listener (left) interprets *then to orange's other side* incorrectly, while the rational listener discounts this interpretation (it could, for example, be better described by *to the left of blue*) and produces the action the descriptions were meant to describe (right). To the extent that human annotators already account for pragmatic effects when generating instructions, examples like these suggest that our model's explicit reasoning is able to capture interpretation behavior that the base sequence-to-sequence listener model is unable to model.

6.2 Instruction generation

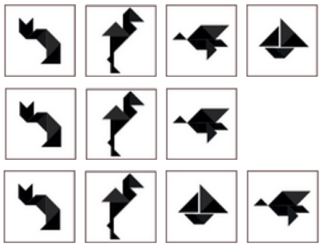
As our primary evaluation for the instruction generation task, we had Mechanical Turk workers carry out directions produced by the speaker models (and by other humans) in a simulated version of each domain. For SAIL, we use the simulator released by Daniele et al. (2017) which was used in their human evaluation results, and we construct simulators for the three SCONE domains. In all settings, we take a sample of 50 action sequences from the domain's test set (using the same sample as Daniele et al. (2017) for SAIL), and have three separate Turk workers attempt to follow the systems' directions for the action sequence.

Table 3 gives the average accuracy of subjects in reaching the intended final world state across all sampled test instances, for each domain. The "human-generated" row reports subjects' accuracy at following the datasets' reference directions. The directions produced by the base speaker S_0 are often much harder to follow than those produced by humans (e.g. 29.3% of S_0's directions are correctly interpretable for Alchemy, vs. 83.3% of human directions). However, we see substantial gains from the rational speaker $S_0 \cdot S_1$ over S_0 in all cases (with absolute gains in accuracy ranging from 12.4% to 46.0%), and the average accuracy of humans at following the rational speaker's directions is substantially higher than for human-produced directions in the Tangrams domain. In the SAIL evaluation, we also include the directions produced by the system of Daniele et al. (2017) (DBW), and find that the rational speaker's directions are followable to comparable accuracy.

We also compare the directions produced by the systems to the reference instructions given by humans in the dataset, using 4-gram BLEU[7] (Pap-

[6] Since the pragmatic inference procedure we use is agnostic to the models' training method, it could also be applied to the models of Guu et al. (2017); however we find that pragmatic inference can improve even upon our stronger base listener models.

[7] See Appendix A for details on evaluating BLEU in the SAIL setting, where there may be a different number of reference and predicted sentences for a given example.



human	take away the last item undo the last step
S_0	remove the last figure add it back
$S_0 \cdot S_1$	remove the last figure add it back in the 3rd position

Figure 4: Descriptions produced for a partial action sequence in the Tangrams domain. Neither the human nor base speaker S_0 correctly specifies where to add the shape in the second step, while the rational speaker $S_0 \cdot S_1$ does.

ineni et al., 2002) in Table 4. Consistent with past work (Krahmer and Theune, 2010), we find that BLEU score is a poor indicator of whether the directions can be correctly followed.

Qualitatively, the rational inference procedure is most successful in fixing ambiguities in the base speaker model's descriptions. Figure 4 gives a typical example of this for the last few timesteps from a Tangrams instance. The base speaker correctly describes that the shape should be added back, but does not specify where to add it, which could lead a listener to add it in the same position it was deleted. The human speaker also makes this mistake in their description. This speaks to the difficulty of describing complex actions pragmatically even for humans in the Tangrams domain. The ability of the pragmatic speaker to produce directions that are easier to follow than humans' in this domain (Table 3) shows that the pragmatic model can generate something different (and in some cases better) than the training data.

6.3 Pragmatics as model combination

Finally, our rational models can be viewed as pragmatically-motivated model combinations, producing candidates using base listener or speaker models and reranking using a combination of scores from both. We want to verify that a rational listener using n ensembled base listeners and n base speakers outperforms a simple ensemble of $2n$ base listeners (and similarly for the rational speaker).

Fixing the total number of models to 20 in each listener experiment, we find that the rational listener (using an ensemble of 10 base listener models and 10 base speaker models) still substantially outperforms the ensembled base listener (using 20 base listener models): accuracy gains are 68.5 $\rightarrow$ 71.6%, 70.1 $\rightarrow$ 72.0%, 71.9 $\rightarrow$ 72.7%, and 69.1 $\rightarrow$ 69.6% for SAIL single-sentence Rel, Alchemy, Scene, and Tangrams, respectively.

For the speaker experiments, fixing the total number of models to 10 (since inference in the speaker models is more expensive than in the follower models), we find similar gains as well: the rational speaker improves human accuracy at following the generated instructions from 61.9 $\rightarrow$ 73.4%, 30.7 $\rightarrow$ 74.7%, 32.0 $\rightarrow$ 66.0%, 58.7 $\rightarrow$ 92.7%, for SAIL, Alchemy, Scene, and Tangrams, respectively.[8]

7 Conclusion

We have demonstrated that a simple procedure for pragmatic inference, with a unified treatment for speakers and listeners, obtains improvements for instruction following as well as instruction generation in multiple settings. The inference procedure is capable of reasoning about sequential, interdependent actions in non-trivial world contexts. We find that pragmatics improves upon the performance of the base models for both tasks, in most cases substantially. While this is perhaps unsurprising for the generation task, which has been discussed from a pragmatic perspective in a variety of recent work in NLP, it is encouraging that pragmatic reasoning can also improve performance for a grounded listening task with sequential, structured output spaces.

Acknowledgments

We are grateful to Andrea Daniele for sharing the SAIL simulator and their system's outputs, to Hongyuan Mei for help with the dataset, and to Tom Griffiths and Chris Potts for helpful comments and discussion. This work was supported by DARPA through the Explainable Artificial Intelligence (XAI) program. DF is supported by a Huawei / Berkeley AI fellowship. JA is supported by a Facebook graduate fellowship.

[8] The accuracies for the base speakers are slightly different than in Table 3, despite being produced by the same systems, since we reran experiments to control as much as possible for time variation in the pool of Mechanical Turk workers.

References

Anne H. Anderson, Miles Bader, Ellen Gurman Bard, Elizabeth Boyle, Gwyneth Doherty, Simon Garrod, Stephen Isard, Jacqueline Kowtko, Jan McAllister, Jim Miller, et al. 1991. The HCRC map task corpus. *Language and speech* 34(4):351–366.

Jacob Andreas and Dan Klein. 2015. Alignment-based compositional semantics for instruction following. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Jacob Andreas and Dan Klein. 2016. Reasoning about pragmatics with neural listeners and speakers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Yoav Artzi, Dipanjan Das, and Slav Petrov. 2014. Learning compact lexicons for CCG semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

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

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations* .

Leon Bergen, Roger Levy, and Noah Goodman. 2016. Pragmatic reasoning through semantic inference. *Semantics and Pragmatics* 9.

S.R.K. Branavan, Harr Chen, Luke S. Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 82–90.

Donna Byron, Alexander Koller, Kristina Striegnitz, Justine Cassell, Robert Dale, Johanna Moore, and Jon Oberlander. 2009. Report on the first NLG challenge on generating instructions in virtual environments (GIVE). In *Proceedings of the 12th european workshop on natural language generation*. Association for Computational Linguistics, pages 165–173.

David L Chen. 2012. Fast online lexicon learning for grounded language acquisition. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. pages 430–439.

David L. Chen and Raymond J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the Meeting of the Association for the Advancement of Artificial Intelligence*. volume 2, pages 1–2.

Robert Dale, Sabine Geldof, and Jean-Philippe Prost. 2005. Using natural language generation in automatic route. *Journal of Research and practice in Information Technology* 37(1):89.

Andrea F. Daniele, Mohit Bansal, and Matthew R. Walter. 2017. Navigational instruction generation as inverse reinforcement learning with neural machine translation. *Proceedings of Human-Robot Interaction* .

Michael C Frank and Noah D Goodman. 2012. Predicting pragmatic reasoning in language games. *Science* 336(6084):998–998.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems 29 (NIPS)*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*. volume 9, pages 249–256.

Robert Goeddel and Edwin Olson. 2012. Dart: A particle-based method for generating easy-to-follow directions. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, pages 1213–1219.

Dave Golland, Percy Liang, and Dan Klein. 2010. A game-theoretic approach to generating spatial descriptions. In *Proceedings of the 2010 conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 410–419.

Noah D Goodman and Andreas Stuhlmüller. 2013. Knowledge and implicature: Modeling language understanding as social cognition. *Topics in cognitive science* 5(1):173–184.

Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. 2016. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems* .

H. P. Grice. 1975. Logic and conversation. In P. Cole and J. L. Morgan, editors, *Syntax and Semantics: Vol. 3: Speech Acts*, Academic Press, San Diego, CA, pages 41–58.

Kelvin Guu, Panupong Pasupat, Evan Zheran Liu, and Percy Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Association for Computational Linguistics (ACL)*.

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

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara L Berg. 2014. ReferItGame: Referring to objects in photographs of natural scenes. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. pages 787–798.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *International Conference on Learning Representations* .

Alexander Koller, Kristina Striegnitz, Andrew Gargett, Donna Byron, Justine Cassell, Robert Dale, Johanna Moore, and Jon Oberlander. 2010. Report on the second NLG challenge on generating instructions in virtual environments (GIVE-2). In *Proceedings of the 6th international natural language generation conference*. Association for Computational Linguistics, pages 243–250.

Emiel Krahmer and Mariët Theune, editors. 2010. *Empirical Methods in Natural Language Generation: Data-oriented Methods and Empirical Evaluation*. Springer-Verlag, Berlin, Heidelberg.

Mike Lewis, Denis Yarats, Yann N Dauphin, Devi Parikh, and Dhruv Batra. 2017. Deal or no deal? end-to-end learning for negotiation dialogues. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the Annual Meeting of the North American Chapter of the Association for Computational Linguistics*.

Reginald Long, Panupong Pasupat, and Percy Liang. 2016. Simpler context-dependent logical forms via model projections. In *Association for Computational Linguistics (ACL)*.

Gary Wai Keung Look. 2008. *Cognitively-inspired direction giving*. Ph.D. thesis, Massachusetts Institute of Technology.

Ruotian Luo and Gregory Shakhnarovich. 2017. Comprehension-guided referring expressions. In *Computer Vision and Pattern Recognition*.

Matt MacMahon, Brian Stankiewicz, and Benjamin Kuipers. 2006. Walk the talk: Connecting language, knowledge, and action in route instructions. *Proceedings of the Meeting of the Association for the Advancement of Artificial Intelligence* 2(6):4.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan Yuille, and Kevin Murphy. 2016. Generation and comprehension of unambiguous object descriptions. In *Computer Vision and Pattern Recognition*.

Hongyuan Mei, Mohit Bansal, and Matthew Walter. 2016. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *Proceedings of the Meeting of the Association for the Advancement of Artificial Intelligence*.

Will Monroe, Robert X.D. Hawkins, Noah D. Goodman, and Christopher Potts. 2017. Colors in context: A pragmatic neural model for grounded language understanding. *Transactions of the Association for Computational Linguistics* .

Will Monroe and Christopher Potts. 2015. Learning in the Rational Speech Acts model. In *Proceedings of 20th Amsterdam Colloquium*. ILLC, Amsterdam.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 311–318.

Christopher Potts. 2011. *Pragmatics*, Oxford University Press.

Nathaniel J Smith, Noah Goodman, and Michael Frank. 2013. Learning and using language via recursive pragmatic reasoning about other agents. In *Advances in Neural Information Processing Systems*. pages 3039–3047.

Kristina Striegnitz, Alexandre Denis, Andrew Gargett, Konstantina Garoufi, Alexander Koller, and Mariët Theune. 2011. Report on the second second challenge on generating instructions in virtual environments GIVE-2.5. In *Proceedings of the 13th European workshop on natural language generation*. Association for Computational Linguistics, pages 270–279.

Jong-Chyi Su, Chenyun Wu, Huaizu Jiang, and Subhransu Maji. 2017. Reasoning about fine-grained attribute phrases using reference games. In *International Conference on Computer Vision*.

Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R. Walter, Ashis Gopal Banerjee, Seth Teller, and Nicholas Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *In Proceedings of the National Conference on Artificial Intelligence*.

Ramakrishna Vedantam, Samy Bengio, Kevin Murphy, Devi Parikh, and Gal Chechik. 2017. Context-aware captions from context-agnostic supervision. In *Computer Vision and Pattern Recognition (CVPR)*. volume 3.

Sida I. Wang, Percy Liang, and Christopher D. Manning. 2016. Learning language games through interaction. In *Association for Computational Linguistics (ACL)*.

Tom Williams, Gordon Briggs, Bradley Oosterveld, and Matthias Scheutz. 2015. Going beyond literal command-based instructions: Extending robotic natural language interaction capabilities. In *AAAI*. pages 1387–1393.

Lei Yu, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Tomas Kocisky. 2017a. The neural noisy channel. *International Conference on Learning Representations* .

Licheng Yu, Hao Tan, Mohit Bansal, and Tamara L. Berg. 2017b. A joint speaker-listener-reinforcer model for referring expressions. In *Computer Vision and Pattern Recognition*.

type	arguments	contextual embedding
Alchemy		
MIX	source i	contents of i
POUR	source i, target j	contents of i and j
DRAIN	amount a, source i	a, contents of i
Scene		
ENTER	color c, source i	people at $i-1$ and $i+1$
EXIT	source i	people at i, $i-1$, $i+1$
MOVE	source i, target j	people at i, $j-1$, $j+1$
SWITCH	source i, target j	people at i and j
TAKEHAT	source i, target j	people at i and j
Tangrams		
REMOVE	position i	—
SWAP	positions i and j	—
INSERT	position i, shape s	index of step when s was removed

Table 5: Action types, arguments, and elements of the world state or action history that are extracted to produce contextual action embeddings.

A Supplemental Material

A.1 SCONE listener details

We factor action production in each of the three SCONE domains, separately predicting the action type and the arguments specific to that action type. Action types and arguments are listed in the first two columns of Table 5. For example, Alchemy's actions involve predicting the action type, a potential source beaker index i and target beaker index j, and potential amount to drain a. All factors of the action (the type and options for each argument) are predicted using separate attention mechanisms, which produce a vector q_f giving unnormalized scores for factor f (e.g. scoring each possible type, or each possible choice for the argument).

We also obtain state-specific embeddings of actions, to make it easier for the model to learn relevant features from the state embeddings (e.g. rather than needing to learn to select the region of the state vector corresponding to the 5th beaker in the action MIX(5) in Alchemy, this action's contextual embedding encodes the current content of the 5th beaker). We incorporate these state-specific embeddings into computation of the action probabilities using a bilinear bonus score:

$$b(a) = q^\top W_{qa} a + w_a^\top a$$

where q is the concatenation of all q_f factor scoring vectors, and W_{qa} and w_a are a learned parameter matrix and vector, respectively. This bonus score $b(a)$ for each action is added to the unnormalized score for the corresponding action a (computed by summing the entries of the q_f vectors which correspond to the factored action components), and the normalized output distribution is then produced using a softmax over all valid actions.

A.2 SAIL speaker details

Since our speaker model operates on segmented action sequences, we train a route segmenter on the training data and then predict segmentations for the test data. This provides a closer comparison to the generation system of Daniele et al. (2017) which did not use segmented routes. The route segmenter runs a bidirectional LSTM over the concatenated state and action embeddings (as in the speaker encoder), then uses a logistic output layer to classify whether the route should be split at each possible timestep. We also collapse consecutive sequences of forward movement actions into single actions (e.g. MOVE4 representing four consecutive forward movements), which we found helped prevent counting errors (such as outputting *move forward three* when the correct route moved forward four steps).

A.3 SCONE speaker details

We use a one-hot representation of the arguments (see Table 5) and contextual embedding (as described in A.1) for each action a_t as input to the SCONE speaker encoder at time t (along with the representation e_t of the world state, as in SAIL). Since SCONE uses a monotonic, one-to-one alignment between actions and direction sentences, the decoder does not use a learned attention mechanism but fixes the contextual representation z_k to be the encoded vector at the action corresponding to the sentence currently being generated.

A.4 Training details

We optimize model parameters using ADAM (Kingma and Ba, 2015) with default hyperparameters and the initialization scheme of Glorot and Bengio (2010). All LSTMs have one layer. The LSTM cell in both the listener and the follower use coupled input and forget gates, and peephole connections to the cell state (Greff et al., 2016). We also apply the LSTM variational dropout scheme of Gal and Ghahramani (2016), using the same dropout rate for inputs, outputs, and recurrent connections. See Table 6 for hyperparameters. We

model	domain	dropout rate	hidden dim	attention dim
L_0	SAIL	0.25	100	100
L_0	Alchemy	0.1	50	50
L_0	Scene	0.1	100	100
L_0	Tangrams	0.3	50	100
S_0	SAIL	0.25	100	100
S_0	Alchemy	0.3	100	–
S_0	Scene	0.3	100	–
S_0	Tangrams	0.3	50	–

Table 6: Hyperparameters for the base listener (L_0) and speaker (S_0) models. The SCONE speakers do not use an attention mechanism.

perform early stopping using the evaluation metric (accuracy for the listener and BLEU score for the speaker) on the development set.

A.5 Computing BLEU for SAIL

To compute BLEU in the SAIL experiments, as the speaker models may choose produce a different number of sentences for each route than in the true description, we obtain a single sequence of words from a multi-sentence description produced for a route by concatenating the sentences, separated by end-of-sentence tokens. We then calculate corpus-level 4-gram BLEU between all these sequences in the test set and the true multi-sentence descriptions (concatenated in the same way).

Hierarchical Structured Model for Fine-to-coarse Manifesto Text Analysis

Shivashankar Subramanian **Trevor Cohn** **Timothy Baldwin**
School of Computing and Information Systems
The University of Melbourne
shivashankar@student.unimelb.edu.au {t.cohn,tbaldwin}@unimelb.edu.au

Abstract

Election manifestos document the intentions, motives, and views of political parties. They are often used for analysing a party's fine-grained position on a particular issue, as well as for coarse-grained positioning of a party on the left–right spectrum. In this paper we propose a two-stage model for automatically performing both levels of analysis over manifestos. In the first step we employ a hierarchical multi-task structured deep model to predict fine- and coarse-grained positions, and in the second step we perform post-hoc calibration of coarse-grained positions using probabilistic soft logic. We empirically show that the proposed model outperforms state-of-art approaches at both granularities using manifestos from twelve countries, written in ten different languages.

1 Introduction

The adoption of NLP methods has led to significant advances in the field of computational social science (Lazer et al., 2009), including political science (Grimmer and Stewart, 2013). Among a myriad of data sources, election manifestos are a core artifact in political analysis. One of the most widely used datasets by political scientists is the Comparative Manifesto Project (CMP) dataset (Volkens et al., 2017), which contains manifestos in various languages, covering over 1000 parties across 50 countries, from elections dating back to 1945.

In CMP, a subset of the manifestos has been manually annotated at the sentence-level with one of 57 political themes, divided into 7 major categories.[1] Such categories capture party positions (FAVORABLE, UNFAVORABLE or NEITHER) on fine-grained policy themes, and are also useful for downstream tasks including calculating manifesto-level (policy-based) left–right position scores (Budge et al., 2001; Lowe et al., 2011; Däubler and Benoit, 2017). An example sentence from the Green Party of England and Wales 2015 election manifesto where they take an UNFAVORABLE position on MILITARY is:

> *We would: Ensure that ... less is spent on military research.*

Elsewhere, they take a FAVORABLE position on WELFARE STATE:

> *Double Child Benefit.*

Such manual annotations are labor-intensive and prone to annotation inconsistencies (Mikhaylov et al., 2012). In order to overcome these challenges, supervised sentence classification approaches have been proposed (Verberne et al., 2014; Subramanian et al., 2017).

Other than the sentence-level labels, the manifesto text also has a document-level score that quantifies its position on the left–right spectrum. Different approaches have been proposed to derive this score, based on alternate definitions of "left–right" (Slapin and Proksch, 2008; Benoit and Laver, 2007; Lo et al., 2013; Däubler and Benoit, 2017). Among these, the RILE index is the most widely adopted (Merz et al., 2016; Jou and Dalton, 2017), and has been shown to correlate highly with other popular scores (Lowe et al., 2011). RILE is defined as the difference between RIGHT and LEFT positions on (pre-determined) policy themes across sentences in a manifesto (Volkens et al., 2013); for instance, UNFAVORABLE position on MILITARY is categorized as LEFT. RILE is popular in CMP in particular, as mapping individual sentences to LEFT/RIGHT/NEUTRAL categories has

[1] https://manifesto-project.wzb.eu/coding_schemes/mp_v5

Proceedings of NAACL-HLT 2018, pages 1964–1974
New Orleans, Louisiana, June 1 - 6, 2018.

been shown to be less sensitive to systematic errors than other sentence-level class sets (Klingemann et al., 2006; Volkens et al., 2013).

Finally, expert survey scores are gaining popularity as a means of capturing manifesto-level political positions, and are considered to be context- and time-specific, unlike RILE (Volkens et al., 2013; Däubler and Benoit, 2017). We use the Chapel Hill Expert Survey (CHES) (Bakker et al., 2015), which comprises aggregated expert surveys on the ideological position of various political parties. Although CHES is more subjective than RILE, the CHES scores are considered to be the gold-standard in the political science domain.

In this work, we address both fine- and coarse-grained multilingual manifesto text policy position analysis, through joint modeling of sentence-level classification and document-level positioning (or ranking) tasks. We employ a two-level structured model, in which the first level captures the structure within a manifesto, and the second level captures context and temporal dependencies across manifestos. Our contributions are as follows:

- we employ a hierarchical sequential deep model that encodes the structure in manifesto text for the sentence classification task;
- we capture the dependency between the sentence- and document-level tasks, and also utilize additional label structure (categorization into LEFT/RIGHT/NEUTRAL: Volkens et al. (2013)) using a joint-structured model;
- we incorporate contextual information (such as political coalitions) and encode temporal dependencies to calibrate the coarse-level manifesto position using probabilistic soft logic (Bach et al., 2015), which we evaluate on the prediction of the RILE index or expert survey party position score.

2 Related Work

Analysing manifesto text is a relatively new application at the intersection of political science and NLP. One line of work in this space has been on sentence-level classification, including classifying each sentence according to its major political theme (1-of-7 categories) (Zirn et al., 2016; Glavaš et al., 2017a), its position on various policy themes (Verberne et al., 2014; Biessmann, 2016; Subramanian et al., 2017), or its relative disagreement with other parties (Menini et al., 2017). Recent approaches (Glavaš et al., 2017a; Subramanian et al., 2017) have also handled multilingual manifesto text (given that manifestos span multiple countries and languages; see Section 5.1) using multilingual word embeddings.

At the document level, there has been work on using label count aggregation of (manually-annotated) fine-grained policy positions, as features for inductive analysis (Lowe et al., 2011; Däubler and Benoit, 2017). Text-based approaches has used dictionary-based supervised methods, unsupervised factor analysis based techniques and graph propagation based approaches (Hjorth et al., 2015; Bruinsma and Gemenis, 2017; Glavaš et al., 2017b). A recent paper closely aligned with our work is Subramanian et al. (2017), who address both sentence- and document-level tasks jointly in a multilingual setting, showing that a joint approach outperforms previous approaches. But they do not exploit the structure of the text and use a much simpler model architecture: averages of word embeddings, versus our bi-LSTM encodings; and they do not leverage domain information and temporal regularities that can influence policy positions (Greene, 2016). This work will act as a baseline in our experiments in Section 5.

Policy-specific position classification can be seen as related to target-specific stance classification (Mohammad et al., 2017), except that the target is not explicitly mentioned in most cases. Secondly, manifestos have both fine- and coarse-grained positions, similar to sentiment analysis (McDonald et al., 2007). Finally, manifesto text is well structured within and across documents (based on coalition), has temporal dependencies, and is multilingual in nature.

3 Proposed Approach

In this section, we detail the first step of our two-stage approach. We use a hierarchical bidirectional long short-term memory ("bi-LSTM") model (Hochreiter and Schmidhuber, 1997; Graves et al., 2013; Li et al., 2015) with a multi-task objective for the sentence classification and document-level regression tasks. A post-hoc calibration of coarse-grained manifesto position is given in Section 4.

Let D be the set of manifestos, where a manifesto $d \in D$ is made up of L sentences, and a sentence s_i has T words: $w_{i1}, w_{i2}, ...w_{iT}$. The set $D_s \subset D$ is annotated at the sentence-level

with positions on fine-grained policy issues (57 classes). The task here is to learn a model that can: (a) classify sentences according to policy issue classes; and (b) score the overall document on the policy-based left–right spectrum (RILE), in an inter-dependent fashion.

Word encoder: We initialize word vector representations using a multilingual word embedding matrix, W_e. We construct W_e by aligning the embedding matrices of all the languages to English, in a pair-wise fashion. Bilingual projection matrices are built using pre-trained FastText monolingual embeddings (Bojanowski et al., 2017) and a dictionary D constructed by translating 5000 frequent English words using Google Translate. Given a pair of embedding matrices E (English) and O (Other), we use singular value decomposition of $O^T DE$ (which is $U\Sigma V^T$) to get the projection matrix (W^*=UV^T), since it also enforces monolingual invariance (Artetxe et al., 2016; Smith et al., 2017). Finally, we obtain the aligned embedding matrix, W_e, as OW^*.

We use a bi-LSTM to derive a vector representation of each word in context. The bi-LSTM traverses the sentence s_i in both the forward and backward directions, and the encoded representation for a given word $\mathbf{w}_{it} \in s_i$, is defined by concatenating its forward ($\overrightarrow{\mathbf{h}}_{it}$) and backward hidden states ($\overleftarrow{\mathbf{h}}_{it}$), $t \in [1, T]$.

Sentence model: Similarly, we use a bi-LSTM to generate a sentence embedding from the word-level bi-LSTM, where each input sentence s_i is represented using the last hidden state of both the forward and backward LSTMs. The sentence embedding is obtained by concatenating the hidden representations of the sentence-level bi-LSTM, in both the directions, $\mathbf{h}_i = [\overrightarrow{\mathbf{h}}_i, \overleftarrow{\mathbf{h}}_i]$, i $\in [1, L]$. With this representation, we perform fine-grained classification (to one-of-57 classes), using a softmax output layer for each sentence. We minimize the cross-entropy loss for this task, over the sentence-level labeled set $D_s \subset D$. This loss is denoted $\mathcal{L}_S$.

Document model: To represent a document d we use average-pooling over the sentence representations $\mathbf{h}_i$ and predicted output distributions ($\mathbf{y}_i$) of individual sentences,[2] i.e., $\mathbf{V}_d = \frac{1}{L}\sum_{i \in d} \begin{bmatrix} \mathbf{y}_i \\ \mathbf{h}_i \end{bmatrix}$. The range of RILE is $[-100, 100]$, which we scale to the range $[-1, 1]$, and model using a final $\tanh$ layer. We minimize the mean-squared error loss function between the predicted $\hat{r}_d$ and actual RILE score r_d, which is denoted as $\mathcal{L}_D$:

$$\mathcal{L}_D = \frac{1}{|D|}\sum_{d=1}^{|D|} \|\hat{r}_d - r_d\|_2^2 \quad (1)$$

Overall, the loss function for the joint model (Figure 1), combining $\mathcal{L}_S$ and $\mathcal{L}_D$, is:

$$\mathcal{L}_J = \alpha\mathcal{L}_S + (1-\alpha)\mathcal{L}_D \quad (2)$$

where $0 \leq \alpha \leq 1$ is a hyper-parameter which is tuned on a development set.

3.1 Joint-Structured Model

The RILE score is calculated directly from the sentence labels, based on mapping each label according to its positioning on policy themes, as LEFT, RIGHT and NEUTRAL (Volkens et al., 2013). Specifically, 13 out of 57 classes are categorized as LEFT, another 13 as RIGHT, and the rest as NEUTRAL. We employ an explicit structured loss which minimizes the deviation between sentence-level LEFT/RIGHT/NEUTRAL polarity predictions $\mathbf{p}$ and the document-level RILE score. The motivation to do this is two-fold: (a) enabling interaction between the sentence- and document-level tasks with homogeneous target space (polarity and RILE); and (b) since we have more documents with just RILE and no sentence-level labels,[3] augmenting an explicit semi-supervised learning objective could propagate down the RILE label to generate sentence labels that concord with the document score.

For the sentence-level polarity prediction (shown in Figure 1), we use cross-entropy loss over the sentence-level labeled set $D_s \subset D$, which is denoted as $\mathcal{L}_{S_P}$. The explicit structured sentence-document loss is given as:

$$\mathcal{L}_{struc} = \frac{1}{|D|}\sum_{d=1}^{|D|}\left(\frac{1}{L_d}\sum_{i \in d}(p_{i_{right}} - p_{i_{left}}) - r_d\right)^2 \quad (3)$$

[2] Preliminary experiments suggested that this representation performs better than using either hidden representations or just the output distribution.

[3] Strictly speaking, for these documents even, sentence annotation was used to derive the RILE score, but the sentence-level labels were never made available.

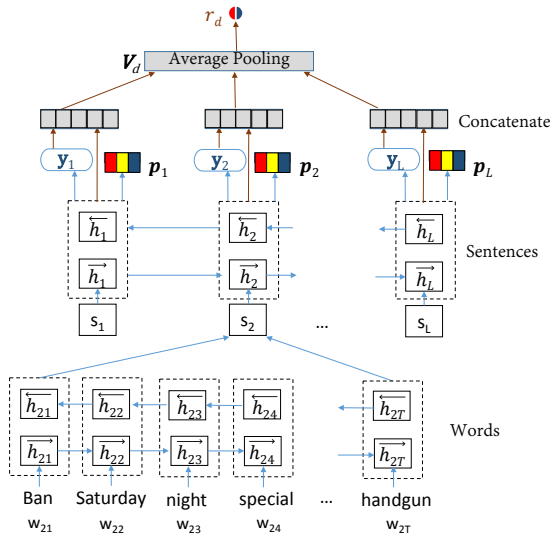


Figure 1: Hierarchical bi-LSTM for joint sentence–document analysis ($\mathbf{y}_i$ denotes the predicted 57-class distribution of sentence s_i; $\mathbf{p}_i$ denotes the distribution over LEFT (in red), RIGHT (in blue) and NEUTRAL (in yellow); r_d denotes the RILE score of d).

where $p_{i_{right}}$ and $p_{i_{left}}$ are the predicted RIGHT and LEFT class probabilities for a sentence s_i ($\in d$), r_d is the actual RILE score for the document d, and L_d is the length of each document, d $\in$ D. We augment the joint model's loss function (Equation (2)) with $\mathcal{L}_{S_P}$ and $\mathcal{L}_{struc}$ to generate a regularized multi-task loss:

$$\mathcal{L}_T = \mathcal{L}_J + \beta\mathcal{L}_{S_P} + \gamma\mathcal{L}_{struc} \quad (4)$$

where $\beta, \gamma \geq 0$ are hyper-parameters which are, once again, tuned on the development set. We refer to the model trained with Equation (2) as "*Joint*", and that trained with Equation (4) as "*Joint*$_{struc}$".

4 Manifesto Position Re-ranking

We leverage party-level information to enforce smoothness and regularity in manifesto positioning on the left–right spectrum (Greene, 2016). For example, manifestos released by parties in a coalition are more likely to be closer in RILE score, and a party's position in an election is often a relative shift from its position in earlier election, so temporal information can provide smoother estimations.

4.1 Probabilistic Soft Logic

To address this, we propose an approach using hinge-loss Markov random fields ("HL-MRFs"), a scalable class of continuous, conditional graphical models (Bach et al., 2013). HL-MRFs have been used for many tasks including political framing analysis on Twitter (Johnson et al., 2017) and user stance classification on socio-political issues (Sridhar et al., 2014). These models can be specified using Probabilistic Soft Logic ("PSL") (Bach et al., 2015), a weighted first order logical template language. An example of a PSL rule is

$$\lambda : \mathtt{P}(\mathtt{a}) \wedge \mathtt{Q}(\mathtt{a}, \mathtt{b}) \rightarrow \mathtt{R}(\mathtt{b})$$

where P, Q, and R are predicates, a and b are variables, and λ is the weight associated with the rule. PSL uses soft truth values for predicates in the interval $[0, 1]$. The degree of ground rule satisfaction is determined using the Lukasiewicz t-norm and its corresponding co-norm as the relaxation of the logical AND and OR, respectively. The weight of the rule indicates its importance in the HL-MRF probabilistic model, which defines a probability density function of the form:

$$P(\mathbf{Y}|\mathbf{X}) \propto \exp\left(-\sum_{r=1}^{M} \lambda_r \phi_r(\mathbf{Y}, \mathbf{X})\right), \quad (5)$$
$$\phi_r(\mathbf{Y}, \mathbf{X}) = \max\{l_r(\mathbf{Y}, \mathbf{X}), 0\}^{\rho_r},$$

where $\phi_r(\mathbf{Y}, \mathbf{X})$ is a hinge-loss potential corresponding to an instantiation of a rule, and is specified by a linear function l_r and optional exponent $\rho_r \in \{1, 2\}$. Note that the hinge-loss potential captures the distance to satisfaction.[4]

4.2 PSL Model

Here we elaborate our PSL model (given in Table 1) based on coalition information, manifesto content-based features (manifesto similarity and right–left ratio), and temporal dependency. Our target `pos` (calibrated RILE) is a continuous variable $[0, 1]$, where 1 indicates that a manifesto occupies an extreme right position, 0 denotes an extreme left position, and 0.5 indicates center. Each instance of a manifesto and its party affiliation are denoted by the predicates `Manifesto` and `Party`.

Coalition: We model multi-relational networks based on regional coalitions within a given country (`RegCoalition`),[5] and also cross-country coalitions in the European parliament

[4] Degree of satisfaction for the example PSL rule r, $\neg\mathtt{P} \vee \neg\mathtt{Q} \vee \mathtt{R}$, using the Lukasiewicz co-norm is given as $\min\{2 - \mathtt{P} - \mathtt{Q} + \mathtt{R}, 1\}$. From this, the distance to satisfaction is given as $\max\{\mathtt{P} + \mathtt{Q} - \mathtt{R} - 1, 0\}$, where $\mathtt{P} + \mathtt{Q} - \mathtt{R} - 1$ indicates the linear function l_r.

[5] http://www.parlgov.org/

PSL$_{coal}$ — Coalition features
$\texttt{Manifesto(x)} \wedge \texttt{Party(x,a)} \wedge \texttt{Manifesto(y)} \wedge \texttt{Party(y,b)} \wedge \texttt{SameElec(x,y)} \wedge \texttt{RegCoalition(a,b)} \wedge \texttt{pos(x)} \rightarrow \texttt{pos(y)}$
$\texttt{Manifesto(x)} \wedge \texttt{Party(x,a)} \wedge \texttt{Manifesto(y)} \wedge \texttt{Party(y,b)} \wedge \texttt{SameElec(x,y)} \wedge \texttt{RegCoalition(a,b)} \wedge \neg\texttt{pos(x)} \rightarrow \neg\texttt{pos(y)}$
$\texttt{Manifesto(x)} \wedge \texttt{Party(x,a)} \wedge \texttt{Manifesto(y)} \wedge \texttt{Party(y,b)} \wedge \texttt{Recent(x,y)} \wedge \texttt{EUCoalition(a,b)} \wedge \texttt{pos(x)} \rightarrow \texttt{pos(y)}$
$\texttt{Manifesto(x)} \wedge \texttt{Party(x,a)} \wedge \texttt{Manifesto(y)} \wedge \texttt{Party(y,b)} \wedge \texttt{Recent(x,y)} \wedge \texttt{EUCoalition(a,b)} \wedge \neg\texttt{pos(x)} \rightarrow \neg\texttt{pos(y)}$
Transitivity
$\texttt{Manifesto(x)} \wedge \texttt{Party(x,a)} \wedge \texttt{Manifesto(y)} \wedge \texttt{Party(y,b)} \wedge \texttt{Manifesto(z)} \wedge \texttt{Party(z,c)} \wedge \texttt{SameElec(x,y)} \wedge \texttt{SameElec(y,z)} \wedge \texttt{RegCoalition(a,b)} \wedge \texttt{RegCoalition(b,c)} \wedge \texttt{pos(x)} \rightarrow \texttt{pos(z)}$
$\texttt{Manifesto(x)} \wedge \texttt{Party(x,a)} \wedge \texttt{Manifesto(y)} \wedge \texttt{Party(y,b)} \wedge \texttt{Manifesto(z)} \wedge \texttt{Party(z,c)} \wedge \texttt{SameElec(x,y)} \wedge \texttt{SameElec(y,z)} \wedge \texttt{RegCoalition(a,b)} \wedge \texttt{RegCoalition(b,c)} \wedge \neg\texttt{pos(x)} \rightarrow \neg\texttt{pos(z)}$
$\texttt{Manifesto(x)} \wedge \texttt{Party(x,a)} \wedge \texttt{Manifesto(y)} \wedge \texttt{Party(y,b)} \wedge \texttt{Manifesto(z)} \wedge \texttt{Party(z,c)} \wedge \texttt{Recent(x,y)} \wedge \texttt{Recent(y,z)} \wedge \texttt{EUCoalition(a,b)} \wedge \texttt{EUCoalition(b,c)} \wedge \texttt{pos(x)} \rightarrow \texttt{pos(z)}$
$\texttt{Manifesto(x)} \wedge \texttt{Party(x,a)} \wedge \texttt{Manifesto(y)} \wedge \texttt{Party(y,b)} \wedge \texttt{Manifesto(z)} \wedge \texttt{Party(z,c)} \wedge \texttt{Recent(x,y)} \wedge \texttt{Recent(y,z)} \wedge \texttt{EUCoalition(a,b)} \wedge \texttt{EUCoalition(b,c)} \wedge \neg\texttt{pos(x)} \rightarrow \neg\texttt{pos(z)}$
PSL$_{esim}$ — Similarity-based relational feature
$\texttt{Manifesto(x)} \wedge \texttt{Manifesto(y)} \wedge \texttt{Similarity(x,y)} \wedge \texttt{Recent(x,y)} \wedge \texttt{pos(x)} \rightarrow \texttt{pos(y)}$
$\texttt{Manifesto(x)} \wedge \texttt{Manifesto(y)} \wedge \texttt{Similarity(x,y)} \wedge \texttt{Recent(x,y)} \wedge \neg\texttt{pos(x)} \rightarrow \neg\texttt{pos(y)}$
PSL$_{ploc}$ — Right–left ratio
$\texttt{Manifesto(x)} \wedge \texttt{LwRightLeftRatio(x)} \rightarrow \texttt{pos(x)}$
$\texttt{Manifesto(x)} \wedge \neg\texttt{LwRightLeftRatio(x)} \rightarrow \neg\texttt{pos(x)}$
PSL$_{temp}$— Temporal Dependency
$\texttt{Manifesto(x)} \wedge \texttt{Party(x,a)} \wedge \texttt{PreviousManifesto(x,a,t)} \wedge \texttt{pos(t)} \rightarrow \texttt{pos(x)}$
$\texttt{Manifesto(x)} \wedge \texttt{Party(x,a)} \wedge \texttt{PreviousManifesto(x,a,t)} \wedge \neg\texttt{pos(t)} \rightarrow \neg\texttt{pos(x)}$

Table 1: PSL Model: Values for `Similarity`, `LwRightLeftRatio` and `pos` are obtained from the joint-structured model (Figure 1). Except for `pos`, other values are fixed in the network. Domain (`y`) for $\texttt{SameElec(x,y)}$ is within the country, and for $\texttt{Recent(x,y)}$ covers all the countries. $\neg$ denotes negation. Distance to satisfaction for each ground rule is obtained using a hinge-loss potential, which is then used inside the HL-MRF model (Equation (5)), where `pos` is $\mathbf{Y}$.

(`EUCoalition`).[6] We set the scope of interaction between manifestos (`x` and `y`) from a country to the same election (`SameElec`). For manifestos across countries, we consider only the most recent manifesto (`Recent`) from each party (`y`), released within 4 years relative to `x`. We use a logistic transformation of the number of times two parties have been in a coalition in the past (to get a value between 0 and 1), for both `RegCoalition` and `EUCoalition`. We also construct rules based on transitivity for both the relational features, i.e., parties which have had common coalition partners, even if they were not allies themselves, are likely to have similar policy positions.

Manifesto similarity: Manifestos that are similar in content are expected to have similar RILE scores (and associated sentence-level label distributions), similar to the modeling intuition captured by Burford et al. (2015) in the context of congressional debate vote prediction. For a pair of recent manifestos (`Recent`) we use the cosine similarity (`Similarity`) between their respective document vectors $\mathbf{V_d}$ (Figure 1).

Right–left ratio: For a given manifesto, we compute the ratio of sentences categorized under RIGHT to OTHERS ($\frac{\#\text{ RIGHT}}{\#\text{ RIGHT}+\#\text{ LEFT}+\#\text{ NEUTRAL}}$), where the categorization for sentences is obtained using the joint-structured model (Equation (4)). We also encode the location of sentence l_s in a document, by weighing the count of sentences for each class C by its location value $\sum_{s \in C} \log(l_s + 1)$ (referred to as *loc_lr*). The intuition here is that the beginning parts of a manifesto tends to contain generic information such as preamble, compared to

[6] http://www.europarl.europa.eu

later parts which are more policy-dense. We perform a logistic transformation of *loc_lr* to derive the `LwRightLeftRatio`.

Temporal dependency: We capture the temporal dependency between a party's current manifesto position and its previous manifesto position (`PreviousManifesto`).

Other than for the look-up based random variables, the network is instantiated with predictions (for `Similarity`, `LwRightLeftRatio` and `pos`) from the joint-structured model (Figure 1). All the random variables, except `pos` (which is the target variable), are fixed in the network. These values are then used inside a PSL model for collective probabilistic reasoning, where the first-order logic given in Table 1 is used to define the graphical model (HL-MRF) over the random variables detailed above. Inference on the HL-MRF is used to obtain the most probable interpretation such that it satisfies most ground rule instances, i.e., considering the relational and temporal dependencies.

5 Evaluation

5.1 Experimental Setup

As our dataset, we use manifestos from CMP for European countries only, as in Section 5.5 we will validate the manifesto's overall position on the left-right spectrum, using the Chapel Hill Expert Survey (CHES), which is only available for European countries (Bakker et al., 2015). In this, we sample 1004 manifestos from 12 European countries, written in 10 different languages — Danish (Denmark), Dutch (Netherlands), English (Ireland, United Kingdom), Finnish (Finland), French (France), German (Austria, Germany), Italian (Italy), Portuguese (Portugal), Spanish (Spain), and Swedish (Sweden). Out of the 1004 manifestos, 272 are annotated with both sentence-level labels and RILE scores, and the remainder only have RILE scores (see Table 2 for further statistics).

There are (less) scenarios where a natural sentence is segmented into sub-sentences and annotated with different classes (Däubler et al., 2012). Hence we use NLTK sentence tokenizer followed by heuristics from Däubler et al. (2012) to obtain sub-sentences. Consistent with previous work (Subramanian et al., 2017), we present results with manually segmented and annotated test documents.

Lang.	# Docs (Anntd.)	# Sents (Anntd.)
Danish	175 (36)	29694 (8762)
Dutch	107 (48)	132524 (70559)
English	117 (27)	86603 (34512)
Finnish	97 (16)	17979 (8503)
French	53 (10)	22747 (5559)
German	117 (46)	111376 (73652)
Italian	98 (15)	41455 (5154)
Portuguese	60 (9)	40922 (11077)
Spanish	85 (50)	145355 (93964)
Swedish	95 (15)	19551 (7938)
Total	1004 (272)	648206 (319680)

Table 2: Statistics of dataset ("Anntd." refers to the number of documents with sentence annotations in the second column, and the number of sentences with annotations in the third column).

5.2 Baseline Approaches

Sentence-level baseline approaches include:

- ***BoW-NN***: TF-IDF-weighted unigram bag-of-words representation of sentences (Biessmann, 2016), and monolingual training using a multi-layer perceptron ("MLP") model.
- ***BoT-NN***: Similar to above, but trigram bag-of-words.
- ***AE-NN***: MLP model with average multilingual word embeddings as the sentence representation (Subramanian et al., 2017).
- ***CNN***: Convolutional neural network ("CNN": Glavaš et al. (2017a)) with multilingual word embeddings.
- ***Bi-LSTM***: Simple bi-LSTM over multilingual word embeddings, last hidden units are concatenated to form the sentence representation, and fed directly into a softmax sentence-level layer. We evaluate two scenarios: (1) with a trainable embedding matrix W_e (***Bi-LSTM(+up)***); and (2) without a trainable W_e.

Document-level baseline approaches include:

- ***BoC***: Bag-of-centroids (BoC) document representation based on clustering the word embeddings (Lebret and Collobert, 2014), fed into a neural network regression model.
- ***HCNN***: Hierarchical CNN, where we encode both the sentence and document using stacked CNN layers.

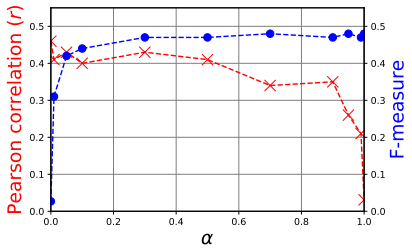
- ***HNN***: State-of-the-art hierarchical neural network model of Subramanian et al. (2017), based on average embedding representations for sentences and the document.

We present results evaluated under two different settings: (a) 80–20% random split averaged across 10 runs to validate the hierarchical model (Section 5.3 and Section 5.4); and (b) temporal setting, where train- and test-set are split chronologically, to validate both the hierarchical deep model and the PSL approach especially, since we encode temporal dependencies (Section 5.5).

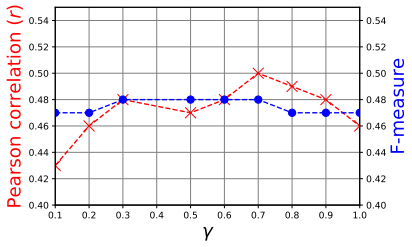
5.3 Hierarchical Sentence- and Document-level Model

We present sentence-level results with a 80–20% random split in Table 3, stratified by country, averaged across 10 runs. For *Bi-LSTM*, we found the setting with a trainable embedding matrix (*Bi-LSTM(+up)*) to perform better than the non-trainable case (*Bi-LSTM*). Hence we use a similar setting for *Joint* and $Joint_{struc}$. We show the effect of α (from Equation (2)) in Figure 2a, based on which we set $\alpha = 0.3$ hereafter. With the chosen model, we study the effect of the structured loss (Equation (4)), by varying γ with fixed $\beta = 0.1$, as shown in Figure 2b. We observe that $\gamma = 0.7$ gives the best performance, and varying β with γ at 0.7 does not result in any further improvement (see Figure 2c). Sentence-level results measured using F-measure, for baseline approaches and the proposed models selected from Figure 2a (*Joint*), Figures 2b and 2c ($Joint_{struc}$) are given in Table 3. We also evaluate the special case of $\alpha = 1$, in the form of sentence-only model $Joint_{sent}$. For the document-level task, results for overall manifesto positioning measured using Pearson's correlation (r) and Spearman's rank correlation (ρ) are given in Table 4. We also evaluate the hierarchical bi-LSTM model with document-level objective only, $Joint_{doc}$.

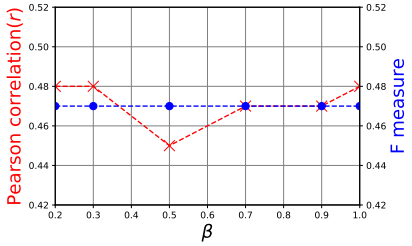
We observe that hierarchical modeling ($Joint_{sent}$, *Joint* and $Joint_{struc}$) gives the best performance for sentence-level classification for all the languages except Portuguese, on which it performs slightly worse than *Bi-LSTM(+up)*. Also, $Joint_{struc}$, does not improve over $Joint_{sent}$. We perform further analysis to see the effect of joint-structured model on the sentence-level task under sparsely-labeled conditions in Section 5.4. On the other hand, for the document-level task, the joint model (*Joint*) performs better than $Joint_{doc}$ and all the baseline approaches. Lastly, the joint-structured model ($Joint_{struc}$) provides further improvement over *Joint*.



(a) Effect of α in equation 2.



(b) Effect of γ in equation 4.



(c) Effect of β in equation 4.

Figure 2: Effect of hyper-parameters on sentence- and document-level performance. • denotes F-measure (right axis) and × denotes Pearson correlation (left axis).

5.4 Analysis of Joint-Structured Model for Sentence-level task

To understand the utility of joint modeling, especially given that there are more manifestos with document-level labels only than both sentence- and document-level labels, we compare the following two settings: (1) $Joint_{struc}$, which uses additional manifestos with document-level supervision (RILE); and (2) $Joint_{sent}$, which uses manifestos with sentence-level supervision only. We vary the proportion of labeled documents at the sentence-level, from 10% to 80%, to study the effect under sparsely-labeled conditions. Note that 80% is the maximum labeled training data under the cross-validation setting. In other cases, a subset (say 10%) is randomly sampled for train-

Lang.	*BoW-NN*	*BoT-NN*	*AE-NN*	*CNN*	*Bi-LSTM*	*Bi-LSTM(+up)*	$Joint_{sent}$	*Joint*	$Joint_{struc}$
Danish	0.35	0.33	0.35	0.31	0.38	0.38	**0.44**	0.40	0.43
Dutch	0.41	0.41	0.40	0.34	0.39	0.43	**0.52**	0.50	0.50
English	0.39	0.43	0.43	0.40	0.45	0.47	0.49	**0.50**	0.49
Finnish	0.30	0.34	0.33	0.30	0.38	0.39	**0.44**	0.41	0.42
French	0.36	0.37	0.36	0.37	0.42	0.44	0.48	**0.49**	0.48
German	0.33	0.35	0.37	0.35	0.40	0.41	0.45	0.45	**0.46**
Italian	0.33	0.38	0.37	0.31	0.37	0.39	0.49	**0.52**	**0.52**
Portuguese	0.32	0.38	0.31	0.28	0.43	**0.46**	0.44	0.44	0.43
Spanish	0.38	0.39	0.39	0.35	0.42	0.41	**0.50**	0.49	**0.50**
Swedish	0.46	0.42	0.36	0.36	0.41	0.44	**0.49**	0.46	0.46
Avg.	0.36	0.38	0.38	0.35	0.40	0.42	**0.48**	0.47	**0.48**

Table 3: Micro-Averaged F-measure for sentence classification. Best scores are given in bold.

Approach	r	ρ
BoC	0.18	0.20
HCNN	0.24	0.26
HNN	0.28	0.32
$Joint_{doc}$	0.30	0.37
Joint	0.46	0.54
$Joint_{struc}$	**0.50**	**0.63**

Table 4: RILE score prediction performance. Best scores are given in bold.

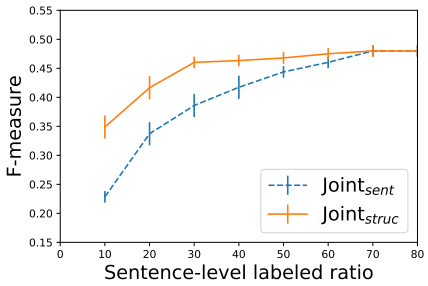


Figure 3: F-measure for $Joint_{struc}$ vs. $Joint_{sent}$ across different ratios of sentence-level labeled manifestos (averaged over 10 runs, with standard deviation)

ing. From Figure 3, having more manifestos with document-level supervision demonstrates the advantage of semi-supervised learning, especially when the sentence-level supervision is sparse ($\leq$ 40%)— $Joint_{struc}$ performs better than $Joint_{sent}$.

5.5 Manifesto Position Re-ranking using PSL

Finally, we present the results using PSL, which calibrates the overall manifesto position on the left–right spectrum, obtained using the joint-structured model ($Joint_{struc}$). As we evaluate the effect of temporal dependency, we use manifestos before 2008-09 for training (868 in total) and the later ones (until 2015, 136 in total) for testing. This test set covers one recent set of election manifestos for most countries, and two for the Netherlands, Spain and United Kingdom. To avoid variance in right-to-left ratio and the target variable (`pos`, initialized using $Joint_{struc}$) between the training and test sets, we build a stacked network (Fast and Jensen, 2008), whereby we estimate values for the training set using cross-validation across the training partition, and estimate values for the test-set with a model trained over the entire training data. Note that we build the $Joint_{struc}$ model afresh using the chronologically split training set, and the parameters are tuned again using an 80-20 random split of the training set. For a consistent view of results for both the tasks (and stages), we provide micro-averaged results for sentence-classification with the competing approaches (from Table 3): *AE-NN* (Subramanian et al., 2017), *Bi-LSTM(+up)*, and $Joint_{struc}$. Results are presented in Table 5, noting that the results for a given method will differ from earlier due to the different data split.

Approach	F-measure
AE-NN	0.31
Bi-LSTM(+up)	0.36
$Joint_{struc}$	**0.42**

Table 5: Micro-averaged F-measure for manifestos released after 2008-09. Best scores are given in bold.

For the document-level regression task, we also evaluate other approaches based on manifesto similarity and automated scaling with sentence-level policy positions:

- **Cross-lingual scaling (*CLS*)**: A recent unsupervised approach for crosslingual political speech text scoring (Glavaš et al., 2017b), based on TF-IDF weighed average word-embeddings to represent documents, and a graph constructed using pair-wise document

	RILE		CHES	
	r	ρ	r	ρ
CLS	0.11	0.10	0.09	0.07
PCA	0.26	0.17	0.01	−0.02
$Joint_{struc}$	0.46	0.42	0.42	0.42
PSL_{coal}	0.51	0.45	0.49	0.45
$PSL_{coal\ +\ esim}$	0.52	0.47	0.50	0.46
$PSL_{coal\ +\ esim\ +\ ploc}$	**0.54**	0.56	0.53	0.56
$PSL_{coal\ +\ esim\ +\ ploc\ +\ temp}$	**0.54**	**0.57**	**0.55**	**0.61**

Table 6: Manifesto regression task using the two-stage approach. Best scores are given in bold.

similarity. Given two pivot texts (for left and right), label propagation approach is used to position other documents.

- **PCA**: Apply principal component analysis (Gabel and Huber, 2000) on the distribution of sentence-level policy positions (56 classes, without 000), and use the projection on its principal component to explain maximum variance in its sentence-level positions, as a latent manifesto-level position score.

- ***Joint*$_{struc}$**: We evaluate the scores obtained using $Joint_{struc}$, which we calibrate using PSL.

We validate the calibrated position scores using both RILE and CHES[7] scores. We use CHES 2010-14, and map the manifestos to the closest survey year (wrt its election date). CHES scores are used only for evaluation and not during training. We provide results in Table 6 by augmenting features for the PSL model (Table 1) incrementally. We observed that the coalition-based feature, and polarity of sentences with its position information improves the overall ranking (r, ρ). Document similarity based relational feature provides only mild improvement (similarly to Burford et al. (2015)), and temporal dependency provides further improvement against CHES. That is, combining content, network and temporal features provides the best results.

6 Conclusion and Future Work

This work has been targeted at both fine- and coarse-grained manifesto text position analysis. We have proposed a two-stage approach, where in the first step we use a hierarchical multi-task deep model to handle the sentence- and document-level tasks together. We also utilize additional information on label structure, to augment an auxiliary structured loss. Since the first step places the manifesto on the left–right spectrum using text only, we leverage context information, such as coalition and temporal dependencies to calibrate the position further using PSL. We observed that: (a) a hierarchical bi-LSTM model performs best for the sentence-level classification task, offering a 10% improvement over the state-of-art approach (Subramanian et al., 2017); (b) modeling the document-level task jointly, and also augmenting the structured loss, gives the best performance for the document-level task and also helps the sentence-level task under sparse supervision scenarios; and (c) the inclusion of a calibration step with PSL provides significant gains in performance against both RILE and CHES, in the form of an increase from $\rho = 0.42$ to 0.61 wrt CHES survey scores.

There are many possible extensions to this work, including: (a) learning multilingual word embeddings with domain information; and (b) modeling other policy related scores from text, such as "support for EU integration".

[7] https://www.chesdata.eu/

Acknowledgements

We thank the anonymous reviewers for their insightful comments and valuable suggestions. This work was funded in part by the Australian Government Research Training Program Scholarship, and the Australian Research Council.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. EMNLP, pages 2289–2294.

S. H. Bach, B. Huang M. Broecheler, and L. Getoor. 2015. Hinge-loss markov random fields and probabilistic soft logic. *CoRR* abs/1505.04406.

Stephen H. Bach, Bert Huang, Ben London, and Lise Getoor. 2013. Hinge-loss markov random fields: Convex inference for structured prediction. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*.

Ryan Bakker, Catherine De Vries, Erica Edwards, Liesbet Hooghe, Seth Jolly, Gary Marks, Jonathan

Polk, Jan Rovny, Marco Steenbergen, and Milada Anna Vachudova. 2015. Measuring party positions in europe: The chapel hill expert survey trend file, 1999–2010. *Party Politics* 21(1):143–152.

Kenneth Benoit and Michael Laver. 2007. Estimating party policy positions: Comparing expert surveys and hand-coded content analysis. *Electoral Studies* 26(1):90–107.

Felix Biessmann. 2016. Automating political bias prediction. *CoRR* abs/1608.02195.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5:135–146.

B. Bruinsma and K. Gemenis. 2017. Validating Wordscores. *CoRR* abs/1707.04737.

I Budge, H.D Klingemann, A Volkens, J Bara, E Tannenbaum, R Fording, D Hearl, H.M Kim, M McDonald, and S Mendes. 2001. *Mapping Policy Preferences: Parties, Electors and Governments*. Oxford University Press.

Clint Burford, Steven Bird, and Timothy Baldwin. 2015. Collective document classification with implicit inter-document semantic relationships. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics (*SEM 2015)*. Denver, USA, pages 106–116.

Thomas Däubler and Kenneth Benoit. 2017. Estimating better left-right positions through statistical scaling of manual content analysis. *Retrieved from* http://kenbenoit.net/pdfs/text_in_context_2017.pdf .

Thomas Däubler, Kenneth Benoit, Slava Mikhaylov, and Michael Laver. 2012. Natural sentences as valid units for coded political texts. *British Journal of Political Science* 42(4):937–951.

Andrew Fast and David Jensen. 2008. Why stacked models perform effective collective classification. In *Proceedings of the Eighth International Conference on Data Mining*. IEEE, pages 785–790.

Matthew J Gabel and John D Huber. 2000. Putting parties in their place: Inferring party left-right ideological positions from party manifestos data. *American Journal of Political Science* pages 94–103.

Goran Glavaš, Federico Nanni, and Simone Paolo Ponzetto. 2017a. Cross-lingual classification of topics in political texts. In *Proceedings of the Second Workshop on NLP and Computational Social Science*. ACL, pages 42–46.

Goran Glavaš, Federico Nanni, and Simone Paolo Ponzetto. 2017b. Unsupervised cross-lingual scaling of political texts. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. volume 2, pages 688–693.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Proceedings of the international conference on Acoustics, speech and signal processing (ICASSP)*. IEEE, pages 6645–6649.

Zachary Greene. 2016. Competing on the issues: How experience in government and economic conditions influence the scope of parties policy messages. *Party Politics* 22(6):809–822.

Justin Grimmer and Brandon M Stewart. 2013. Text as data: The promise and pitfalls of automatic content analysis methods for political texts. *Political analysis* 21(3):267–297.

Frederik Georg Hjorth, Robert Tranekær Klemmensen, Sara Binzer Hobolt, Martin Ejnar Hansen, and Peter Kurrild-Klitgaard. 2015. Computers, coders, and voters: Comparing automated methods for estimating party positions. *Research and Politics* 2(2):1–9.

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

Kristen Johnson, Di Jin, and Dan Goldwasser. 2017. Leveraging behavioral and social information for weakly supervised collective classification of political discourse on twitter. In *Proceedings of the Association for Computational Linguistics*. ACL, pages 741–752.

Willy Jou and Russell J. Dalton. 2017. Left-right orientations and voting behavior. *Oxford Research Encyclopedia of Politics* .

Hans-Dieter Klingemann, Andrea Volkens, Judith Bara, Ian Budge, and Michael McDonald. 2006. *Mapping Policy Preferences II. Estimates for Parties, Electors, and Governments in Eastern Europe, European Union, and OECD*. Oxford University Press.

David Lazer, Alex Sandy Pentland, Lada Adamic, Sinan Aral, Albert Laszlo Barabasi, Devon Brewer, Nicholas Christakis, Noshir Contractor, James Fowler, Myron Gutmann, et al. 2009. Life in the network: the coming age of computational social science. *Science (New York, NY)* 323(5915):721.

Rémi Lebret and Ronan Collobert. 2014. N-gram-based low-dimensional representation for document classification. *CoRR* abs/1412.6277.

Jiwei Li, Luong Minh-Thang, and Jurafsky Dan. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. ACL, pages 1106–1115.

James Lo, Sven-Oliver Proksch, and Thomas Gschwend. 2013. A common left-right scale for voters and parties in europe. *Political Analysis* 22(2):205–223.

Will Lowe, Kenneth Benoit, Slava Mikhaylov, and Michael Laver. 2011. Scaling policy preferences from coded political texts. *Legislative studies quarterly* 36(1):123–155.

Ryan McDonald, Kerry Hannan, Tyler Neylon, Mike Wells, and Jeff Reynar. 2007. Structured models for fine-to-coarse sentiment analysis. In *Proceedings of the 45th annual meeting of the association of computational linguistics*. ACL, pages 432–439.

Stefano Menini, Federico Nanni, Simone Paolo Ponzetto, and Sara Tonelli. 2017. Topic-based agreement and disagreement in us electoral manifestos. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. EMNLP, pages 2938–2944.

Nicolas Merz, Sven Regel, and Jirka Lewandowski. 2016. The manifesto corpus: A new resource for research on political parties and quantitative text analysis. *Research & Politics* 3(2).

Slava Mikhaylov, Michael Laver, and Kenneth R Benoit. 2012. Coder reliability and misclassification in the human coding of party manifestos. *Political Analysis* 20(1):78–91.

Saif M Mohammad, Parinaz Sobhani, and Svetlana Kiritchenko. 2017. Stance and sentiment in tweets. *ACM Transactions on Internet Technology (TOIT)* 17(3):26.

Jonathan B Slapin and Sven-Oliver Proksch. 2008. A scaling model for estimating time-series party positions from texts. *American Journal of Political Science* 52(3):705–722.

Samuel L Smith, Turban David HP, Hamblin Steven, and Hammerla Nils Y. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*.

Dhanya Sridhar, Lise Getoor, and Marilyn Walker. 2014. Collective stance classification of posts in online debate forums. In *Proceedings of the Joint Workshop on Social Dynamics and Personal Attributes in Social Media*. ACL, pages 109–117.

Shivashankar Subramanian, Trevor Cohn, Timothy Baldwin, and Julian Brooke. 2017. Joint sentence-document model for manifesto text analysis. In *Proceedings of the 15th Annual Workshop of The Australasian Language Technology Association*. ALTA, pages 25–33.

Suzan Verberne, Eva Dhondt, Antal van den Bosch, and Maarten Marx. 2014. Automatic thematic classification of election manifestos. *Information Processing & Management* 50(4):554–567.

Andrea Volkens, Judith Bara, Budge Ian, and Simon Franzmann. 2013. Understanding and validating the left-right scale (RILE). In *Mapping Policy Preferences From Texts: Statistical Solutions for Manifesto Analysts*, Oxford University Press, chapter 6.

Andrea Volkens, Pola Lehmann, Theres Matthie, Nicolas Merz, Sven Regel, and Bernhard Weels. 2017. *The Manifesto Data Collection. Manifesto Project (MRG/CMP/MARPOR). Version 2017b*. Wissenschaftszentrum Berlin für Sozialforschung, Berlin, Germany.

Cäcilia Zirn, Goran Glavaš, Federico Nanni, Jason Eichorts, and Heiner Stuckenschmidt. 2016. Classifying topics and detecting topic shifts in political manifestos. In *Proceedings of the International Conference on the Advances in Computational Analysis of Political Text*. PolText, pages 88–93.

Behavior Analysis of NLI Models: Uncovering the Influence of Three Factors on Robustness

V. Ivan Sanchez Carmona and **Jeff Mitchell** and **Sebastian Riedel**
University College London
Department of Computer Science
{i.sanchezcarmona, j.mitchell, s.riedel}@cs.ucl.ac.uk

Abstract

Natural Language Inference is a challenging task that has received substantial attention, and state-of-the-art models now achieve impressive test set performance in the form of accuracy scores. Here, we go beyond this single evaluation metric to examine robustness to semantically-valid alterations to the input data. We identify three factors - *insensitivity*, *polarity* and *unseen pairs* - and compare their impact on three SNLI models under a variety of conditions. Our results demonstrate a number of strengths and weaknesses in the models' ability to generalise to new in-domain instances. In particular, while strong performance is possible on unseen hypernyms, unseen antonyms are more challenging for all the models. More generally, the models suffer from an insensitivity to certain small but semantically significant alterations, and are also often influenced by simple statistical correlations between words and training labels. Overall, we show that evaluations of NLI models can benefit from studying the influence of factors intrinsic to the models or found in the dataset used.

1 Introduction

The task of Natural Language Inference (NLI)[1] has received a lot of attention and has elicited models which have achieved impressive results on the Stanford NLI (SNLI) dataset (Bowman et al., 2015). Such results are impressive due to the linguistic knowledge required to solve the task (LoBue and Yates, 2011; Maccartney, 2009). However, the ever-growing complexity of these models inhibits a full understanding of the phenomena that they capture.

As a consequence, evaluating these models purely on test set performance may not yield enough insight into the complete repertoire of abilities learned and any possible abnormal behaviors (Kummerfeld et al., 2012; Sammons et al., 2010). A similar case can be observed in models from other domains; take as an example an image classifier that predicts based on the image's background rather than on the target object (Zhao et al., 2017; Ribeiro et al., 2016), or a classifier used in social contexts that predicts a label based on racial attributes (Crawford and Calo, 2016). In both examples, the models exploit a bias (an undesired pattern hidden in the dataset) to enhance accuracy. In such cases, the models may appear to be *robust* to new and even challenging test instances; however, this behavior may be due to spurious factors, such as biases. Assessing to what extent the models are robust to these contingencies just by looking at test accuracy is, therefore, difficult.

In this work we aim to study how certain factors affect the robustness of three pre-trained NLI models (a conditional encoder, the DAM model (Parikh et al., 2016), and the ESIM model (Chen et al., 2017)). We call these target factors *insensitivity* (not recognizing a new instance), *polarity* (a word-pair bias), and *unseen pairs* (recognizing the semantic relation of new word pairs). We became aware of these factors based on an exploration of the models' behavior, and we hypothesize that these factors systematically influence the behavior of the models.

In order to systematically test if the above factors affect robustness, we propose a set of challenging instances for the models: We sample a set of instances from SNLI data, we apply a transformation on this set that yields a new set of instances, and we test both how well the models

[1] Also known as Recognizing Textual Entailment.

Proceedings of NAACL-HLT 2018, pages 1975–1985
New Orleans, Louisiana, June 1 - 6, 2018.

classify these new instances and whether the target factors influence the models' behavior. The transformation (swapping a pair of words between premise and hypothesis sentences) is intended to yield both easy and difficult instances to challenge the models, but easy for a human to annotate them.

We draw motivation to study the robustness of NLI models from previous work on evaluating complex models (Isabelle et al., 2017; White et al., 2017). Furthermore, we base our approach on the discipline of behavioral science which provides methodologies for analyzing how certain factors influence the behavior of subjects under study (Epling and Pierce, 1986).

We aim to answer the research questions: How robust is the predictive behavior of the pre-trained models under our transformation to input data? Do the target factors (insensitivity, polarity, and unseen pairs) influence the prediction of the models? Are these factors common across models?

Our results show that the models are robust mainly where the semantics of the new instances do not change significantly with respect to the sampled instances and thus the class labels remain unaltered; i.e., the models are insensitive to our transformation to input data. However, when the class labels change, the models significantly drop accuracy. In addition, the models exploit a bias, polarity, to stay *robust* when facing new instances. We also find that the models are able to cope with unseen word pairs under a hypernym relation, but not with those under an antonym relation, suggesting their inability to learn a symmetric relation.

2 Related Work

2.1 Analysis of Complex Models

Previous works in ML and NLP have analyzed different aspects of complex models using a variety of approaches; for example, understanding input-output relationships by approximating the local or global behavior of the model using an interpretable model (Ribeiro et al., 2016; Craven and Shavlik, 1996), or analyzing the output of the model under lesions of its internal mechanism (Li et al., 2016). Another line of work has analyzed the robustness of NLP models both via controlled experiments to complement the information from the test set accuracy and test abilities of the models (Isabelle et al., 2017; B. Hashemi and Hwa, 2016; White et al., 2017) and via adversarial instances to expose weaknesses (Jia and Liang, 2017). In addition, work has been done to uncover and diminish *gender* biases in datasets captured by structured prediction models (Zhao et al., 2017) and word embeddings (Bolukbasi et al., 2016). However, to the best of our knowledge, there is no previous work to study the robustness of NLI models while analyzing factors affecting their predictions.

2.2 Behavior Analysis

Previous work on behavioral science has focused on understanding how environmental factors influence behaviors in both human (Soman, 2001) and animal (Mench, 1998) subjects with the objective of predicting behavioral patterns or analyzing environmental conditions. This methodology also helps to identify and understand abnormal behaviour by collecting behavioral data without the need to reach any internal component of the subject (Birkett and Newton-Fisher, 2011).

We base our approach in the discipline of behavioral science since some of our research questions and objectives align to those from this discipline; in addition, its methodology to study how factors effect on the subjects' behavior provides statistical guarantees.

3 Background

3.1 Natural Language Inference

NLI, or RTE, is the task of inferring whether a natural language sentence (hypothesis) is entailed by another natural language sentence (premise) (Maccartney, 2009; Dagan et al., 2009; Dagan and Glickman, 2004). More formally, given a pair of natural language sentences $i = (premise, hypothesis)$, a model classifies the type of relation such sentences fall in from three possible classes, *entailment*, where the hypothesis is necessarily true given the premise, *neutral*, where the hypothesis may be true given the premise, and *contradiction*, where the hypothesis is necessarily false given the premise. Solving this task is challenging since it requires linguistic and semantic knowledge, such as co-reference, hypernymy, and antonymy (LoBue and Yates, 2011), as well as pragmatic knowledge and informal reasoning (Maccartney, 2009).

3.2 Behavior Analysis

Behavior analysis seeks to account for the role that factors (independent variables) play in the behavior (dependent variable) of subjects. Testing for

the influence of a factor on the subject's behavior can be done via statistical tests: A null hypothesis states no association between a target factor and behavior, whereas the alternative hypothesis states an association (McDonald, 2014).

4 Dataset and Models

4.1 SNLI Dataset

The Stanford NLI dataset (Bowman et al., 2015) was created with the purpose of training deep neural models while providing human-annotated data. Each instance was created by providing a premise sentence, harvested from a pre-existing dataset, to a crowdsource worker who was instructed to produce three hypothesis sentences, one for each NLI class (entailment, neutral, contradiction). This process yielded a balanced dataset containing around 570K instances.

4.2 Models

Conditional Encoder We use two bidirectional LSTMs; the first LSTM encodes the premise sentence into a fixed-size vector embedding by sequentially reading on a word basis, while the second LSTM encodes the hypothesis sentence conditioned on the representation of the premise sentence. At the final layer we used a softmax over the class labels on top of a 3-layer MLP. All embeddings, of dimensionality $d = 100$, were randomly initialized and learned during training. Accuracy on SNLI's dev set is 0.782.

Decomposable Attention Model DAM (Parikh et al., 2016) consists of 2-layer multilayer-perceptrons (MLPs) factorized in a 3-step process. First, a soft-alignment matrix is created for all the words in both the premise and hypothesis. Then, each word of the premise is paired with the soft-alignment representation of the hypothesis sentence and fed into an MLP, and similarly for each word in the hypothesis with the soft-alignment of the premise. The resulting representations are then aggregated where the vector representations of the premise are summed up and the same for those of the hypothesis; the new representations are then fed to an MLP, followed by a linear layer and a softmax whose output is a class label. We use $d = 300$ dimensional GloVe embeddings (not updated at training time). All layers use the ReLU function. Accuracy on SNLI's dev set is 0.854.

Enhanced Sequential Information Model ESIM (Chen et al., 2017) performs inference in three stages. First, Input Encoding uses BiLSTMs to produce representations of each word in its context within premise or hypothesis. Then, Local Inference Modelling constructs new word representations for each hypothesis (premise) by summing over the BiLSTM hidden states for the premise (hypothesis) words using weights from a soft attention matrix. Additionally, these representations are enhanced with element-wise products and differences of the original hidden states vectors and the new attention based vectors. Finally, Inference Composition uses a BiLSTM, average and max pooling and an MLP output layer to produce predicted labels. Accuracy on SNLI's dev set is 0.882.

5 Methods

We test our main hypothesis (Section 1) by perturbing instances in a controlled, simple, and meaningful way. This alteration, at the instance level, yields new sets of instances which range from *easy* (the semantics and the label of the new instance are the same to those of the original instance) to *challenging* (both semantics and label of the new instance change with respect to those of the original instance), but all of them remain easy to annotate for a human.

To examine how the models generalize from seen instances to transformed instances, we sample our original instances from the SNLI training set, which we refer to as control instances from now on. We then produce new instances which differ either minimally from the control instances, by changing only a single word in the premise and hypothesis, or more substantially, by copying the same sentence structure into the premise and hypothesis with a single word changed. In this way, we produce instances that contain only words seen at training time, within sentence structures also seen at training time. Thus, our evaluation sets are as in-domain as possible, and control for factors associated with novel sentential contexts and vocabulary.

5.1 Basic Procedure and Statistical Analyses

We first sample an instance from the SNLI dataset according to a given criterion, namely we look for a specific word pair in the instance; then, we apply our transformation over the word pair. This pro-

cedure generates a new instance. After that, the models label the new instance, and we statistically analyze which target factors influenced the models to respond in such a way via chi-square (McNemar's, independence, and homogeneity) tests (McDonald, 2014; Alpaydin, 2010). When the sample size is too small we apply Yate's correction or a Fisher test. We use the StatsModels (Seabold and Perktold, 2010) and SciPy (Oliphant, 2007) packages. The level of significance is $p < 0.0001$, unless otherwise stated.[2] This procedure is applied in four experiments, where we study the effect of different word pairs (hypernym, hyponym, and antonyms) and the effect of two types of context words surrounding the word pairs which we refer to as *in situ* and *ex situ* (explained in Section 5.3).

5.2 Transformation and Word Pairs

Given a set of word pairs of the form $W = (w_1, w_2)$, where w_1 and w_2 hold under a semantic relation $s \in \{antonymy, hypernymy, hyponymy\}$, we look through the training set for instances $i_k = (p_k, h_k)$, where p_k and h_k are premise and hypothesis sentences, respectively, such that $w_1 \in p_k$ and $w_2 \in h_k$. For each instance i_k we apply transformation T: we swap w_1 with w_2; this transformation yields an instance $i_m = (p_m, h_m)$ where $w_2 \in p_m, w_1 \in h_m$ and $w_1 \notin p_m, w_2 \notin h_m$.[3]

An example of transformation T on a *contradiction* instance i_k is the following:

(1) p_k : A soccer game occurring at sunset.
h_k : A basketball game is occurring at sunrise.

Where the word pair $(sunset, sunrise)$ are antonyms. After applying transformation T, we obtain the new *contradiction* instance i_m:

(2) p_m : A soccer game occurring at sunrise.
h_m : A basketball game is occurring at sunset.

Consider now the following instance i_l (class label *entailment*):

(3) p_l : A little girl hugs her brother on a footbridge in a forest.
h_l : A pair of siblings are on a bridge.

If we now apply transformation T on the hypernym word pair $(footbridge, bridge)$ we derive the new instance i_n (class *neutral*):

(4) p_n : A little girl hugs her brother on a bridge in a forest.
h_n : A pair of siblings are on a footbridge.

Since swapping word pairs under hypernymy or hyponymy relations may yield a different class label for the new instance, we manually annotate all the instances in the new sample, discarding those that are semantically incoherent.

5.3 Experimental Conditions

We consider two types of sentential context for the word pairs, namely *in situ* and *ex situ*. Examples of instances under the *in situ* condition are Examples 1, 2, 3, and 4 in Section 5.2. The name *in situ* refers to the fact that we analyze the effect of the transformation T within the original context of the premise and hypothesis sentences. This allows to control for confounding factors, such as sentence length and order of the context words.

We also consider an *ex situ* condition in which we remove the word pair from the original premise and hypothesis and analyze the effect of the transformation T within a simplified sentential context which is the same in premise and hypothesis. Specifically, we randomly select either the premise or hypothesis context from the original instance and copy it into both positions. In this way, we obtain a sentence pair where the only difference between the premise and hypothesis is the word pair, which allows us to isolate the effect of this pair from its interaction with the surrounding context; this condition thus allows to control for context words. This process yields a new set of instances, which we refer to as E.

An example of an *ex situ* instance can be constructed from Example 1 (Section 5.2). If the premise sentence is selected, then after performing the procedure described above, the following sentence pair e_k is generated:

(5) p_k : A soccer game occurring at sunset.
h_k : A soccer game occurring at sunrise.

Given a sample E, we apply the transformation T in order to generate a transformed sample E_T where the word pairs are swapped, similar to the procedure applied in Section 5.2 on SNLI control instances in order to generate their transformed instances counterpart. In the latter case, we say that

[2] We apply a Bonferroni correction.

[3] If a word w_1 or w_2 appears more than once, we replace all the appearances with its corresponding pair, w_2 or w_1.

		Whole sample			Subset 1: Gold label changes			Subset 2: Unseen word pairs			Subset 3: Polarity $\neq$ gold label		
Exp	sample	ESIM	DAM	CE	ESIM	DAM	CE	ESIM	DAM	CE	ESIM	DAM	CE
1	I_A	0.970	0.946	0.820							0.900	0.900	0.750
	I_{TA1}	0.933	0.946	0.732				0.600	0.500	0.400	0.681	0.637	0.536
	I_{TA2}	0.721	0.771	0.645	0.554	0.653	0.476						
	I_{TA3}	0.722	0.745	0.646	0.568	0.630	0.535						
2	E_A	0.953	0.958	0.508							0.400	0.500	0.450
	E_{TA}	0.933	0.929	0.480				0.575	0.500	0.175	0.565	0.492	0.260
3	I_H	0.898	0.819	0.828							0.836	0.701	0.733
	I_{TH}	0.648	0.691	0.543	0.315	0.509	0.271	0.694	0.777	0.555	0.719	0.697	0.586
4	E_H	0.771	0.849	0.742							0.715	0.707	0.461
	E_{TH}	0.576	0.788	0.534	0.551	0.783	0.516	0.527	0.666	0.472	0.631	0.674	0.507

Table 1: Accuracy scores of all models. *Exp*: experiment number. *Whole sample*: accuracy scores on the whole sample. *Subset 1*: subset of transformed instances that have different gold label with respect to the control instances they were generated from. *Subset 2*: subset of transformed instances that contain word pairs unseen at training time. *Subset 3*: subset of control or transformed instances containing word pairs whose polarity does not match the instance's gold label.

given a sample of control instances I we generate a transformed sample I_T.

As an example of obtaining a transformed *ex situ* instance, we apply T to $(sunset, sunrise)$ in Example 5 to obtain the new instance e_m:

(6) p_m : A soccer game occurring at sunrise.
h_m : A soccer game occurring at sunset.

We note that for both conditions, *in situ* and *ex situ*, the same word pairs are swapped, so the differences are the surrounding context words and the factors being controlled.

5.4 Test Sets

In each experiment we use two sets of instances in order to measure the robustness of the models and analyze our target factors: 1) The control instances where the target word pair is in its original position and 2) the transformed instances generated after applying transformation T. The name of each set corresponds with the experimental setting it is used in. Samples used in *in situ* experiments are named as I, and E for *ex situ*. Subscripts distinguish both the type of word pairs (A for antonyms and H for hypernym/hyponym) and the type of set (control or transformed). For example, I_A refers to the control *in situ* set whose instances contain antonym word pairs, whereas E_{TH} refers to the *ex situ* transformed test set containing hypernym/hyponym swapped word pairs.

We clarify: a) the sets I_A and I_H are sampled from the SNLI dataset; b) transformed test sets are generated from control sets containing control instances; c) we refer to the sets E_A and E_H as control test sets because the target word pairs are in their original position, and we apply T on them in order to obtain the transformed samples E_{TA} and E_{TH}, respectively.

Details about the sets: In order to build set I_A, we sample only *contradiction* instances (instances in E_A are also *contradictions*). We use the antonym word pairs from (Mohammad et al., 2013) to yield the sets I_{TA1} and E_{TA}, which also only contain *contradictions* since the relation of antonymy is symmetric.[4] We build two more sets, I_{TA2} and I_{TA3} (explained in Section 6.1). Sets I_H, E_H, I_{TH}, and E_{TH} contain instances with any class label. In order to generate sets I_{TH} and E_{TH}, we use the hypernym word pairs from (Baroni et al., 2012). We manually annotate these transformed sets and discard incoherent instances.

5.5 Factors Under Study

We describe the three target factors that we hypothesize that affect the models' response.

Insensitivity is the name we give to the tendency of a model to predict the original label on a transformed instance that is similar to a control instance. Thus a model would be insensitive if, for example, it incorrectly predicts the same class label for both the control instance in Example 3

[4]The word pair $(sunset, sunrise)$ holds in an antonymy relation regardless of the position of the words in premise and hypothesis sentences.

and the transformed instance in Example 4 just because they closely resemble each other. A simple measure of the impact of this effect is to look at the accuracy on the subset of instances in which the gold label was changed by the transformation. We show this effect by statistically correlating the rate of correct predictions with changes in the labels predicted.

Unseen Word Pairs are another factor we can use to evaluate robustness. In this case, we are interested in the subset of transformed instances where the swapped word pair is now in an order within premise and hypothesis that was unseen in the training data. An example is Example 2 which contains the unseen word pair $(sunrise, sunset)$; i.e., no instance in the training set contains the word *sunrise* in the premise and the word *sunset* in the hypothesis. Poor performance on this subset reflects an inability to exploit the symmetry (antonym pairs) or anti-symmetry (hypernym pairs) of the word pairs involved. We show models' abilities to cope with unseen pairs by statistically associating proportions of instances containing unseen pairs with incorrect predictions rates.

Polarity is the name we give to the association between a word pair and the most frequent class it is found in across training instances. For example, we associate the word pair $(sunset, sunrise)$ with polarity *contradiction* because it mainly appears on training instances with label *contradiction*. We define four main categories of polarity: *neutral*, *contradiction*, *entailment*, and *none* for unseen word pairs.[5] Accuracy on the subset of instances where polarity and gold label disagree is an indicator of the extent to which a model is influenced by this factor. For example, a model incorrectly predicting label *entailment* for the instance in Example 4 (class *neutral*) based on the polarity of class *entailment* of its word pair $(bridge, footbridge)$ indicates that the model is influenced by this factor. We show this influence by statistically correlating labels predicted with polarities.

6 Experiments and Results

Table 1 presents the performance of the models across the different test sets. In general, DAM and ESIM seem to be more robust than CE, with the latter's accuracy degrading to essentially random performance on the most challenging subsets. However, this general trend is reversed in a single row of the table. On E_{TH}, ESIM shows a comparable performance to CE. And on Subset 3 of I_H, DAM appears to rely on a bias (polarity) in the same way as CE. Overall, all models are affected by the three target factors, dropping performance up to 0.25, 0.20, and 0.28 for ESIM, DAM, CE, respectively, just by virtue of our simple transformation of swapping words.

6.1 Experiment 1: Swapping Antonyms in *In Situ* Instances

In this experiment we use sets I_A and I_{TA1}. Swapping antonyms seems to have no effect on the overall performance of the DAM model on I_{TA1} when compared to I_A, and little effect on ESIM. Thus these two models appear to be robust to this transformation. Nonetheless, further analysis will not support the conclusion that both models have learned that antonymy is symmetric, and we will show that this seemingly robust behavior is due to confounding factors and not due to inference abilities. Accuracy scores of CE model seem to reveal that it is much less robust to the antonym swap, with performance significantly dropping by roughly 10.5% according to a McNemar's test.

Insensitivity Because instances in I_{TA1} are *contradiction*, we perform a proxy experiment to understand the models' sensitivity. From I_A, we substitute one of the antonyms in each word pair (in each instance) with a hyponym, hypernym, or synonym[6] of the other. Doing this on both the premise and hypothesis yields two new samples, I_{TA2} and I_{TA3}, which we manually annotate.

Examples of control (Example 7) and transformed (Example 8) instances are given below, showing the replacement of *young*, in the hypothesis, with *aged*, a synonym of *elderly* from the premise. This transformation changes gold-label from *contradiction* to *neutral*. Approximately, half the sample yields such changes in gold-label.

(7) p_k : An elderly woman sitting on a bench.
h_k : A young mother sits down.

(8) p_m : An elderly woman sitting on a bench.
h_m : An aged mother sits down.

[5] We also define categories when a word pair appears the same number of times in two classes, such as *entailment-neutral*, though these cases are rare.

[6] We manually select these from WordNet such that it appears at least $t = 10$ times in the training set on either the premise sentences or the hypothesis sentences.

This transformation leads to a considerable drop in overall performance for all models when accuracy scores on sets I_{TA2} and I_{TA3} are compared to the accuracy on the control instances in I_A: up to 0.175 (CE), 0.201 (DAM), and 0.24 (ESIM) points (Table 1). To test if insensitivity to the transformation is associated with these behaviors, we measure accuracy only on those instances that changed gold-label (Subset 1 from the sets I_{TA2} and I_{TA3}), where we see a further reduction in performance for all models. 2-way tests of independence provide strong evidence for the insensitivity of the models (CE: $\chi^2(1) = 73.33$, DAM: $\chi^2(1) = 108.30$, ESIM: $\chi^2(1) = 175.34$).

Table 2 shows the case for ESIM: most of its incorrect predictions are due to predicting the same label on both control and transformed instances when these two type of instances have different gold labels. Paradoxically, this effect works in the models' favour in the antonym swapping case (I_{TA1}) because all the gold-labels remain as *contradiction*. Thus ignoring the transformation will avoid any loss in performance.

	Distribution of predictions	
Labels predicted	correct	incorrect
change	155	31
no change	8	100

Table 2: Contingency table for ESIM: Predictions on transformed instances with different gold labels from those of the control instances.

Unseen Word Pairs The results in the column Subset 2 of I_{TA1} (Table 1) suggest that performance on unseen word pairs is weak. However, only 40 instances within I_{TA1} contain unseen antonym pairs; thus the impact of this result may be limited. 2-way tests of homogeneity show that the difference in accuracy of predictions in instances containing seen or unseen word pairs is nonetheless significant for all models (CE: $\chi^2(1) = 19.46$, DAM: $\chi^2(1) = 74.16$, ESIM: $\chi^2(1) = 39.33$). In other words, the models struggle to recognize the reversed antonym pairs, even though they were all seen in their original order at training time. This effect can be seen, for example, in the contingency table for DAM in Table 3.

Polarity Only 11% of the instances in the transformed sample I_{TA1} contain word pairs that have polarity other than *contradiction*. Thus, a model relying only on this factor could achieve an accuracy of 89%. We investigate if the predicted labels on instances in I_{TA1} are associated with the polarity of the transformed word pair. For all models, independence tests are highly significant (CE: $\chi^2(6) = 30.69$, DAM: $\chi^2(6) = 101.26$, ESIM: $\chi^2(6) = 64.40$). Table 4 shows that the predictions of DAM change according to the polarity of the word pairs. For example, when the polarity is *contradiction*, around 98.5% of the predictions are *contradictions*; however, this figure changes when the polarity is *neutral* where the rate of correct predictions (*contradictions*) fall to 80.7%, and a more dramatic fall is observed when the word pairs are unseen (polarity *none*) where only 50% of the predictions are correct. This is strong evidence that the models learned to rely on polarity.

We note that a model with perfect accuracy on I_{TA1}, would lead to a statistic that does not reject the null hypothesis, showing in this case that the predictions are independent of polarity.

	Word pairs	
Predictions	seen	unseen
correct	567	20
incorrect	13	20

Table 3: Contingency table for DAM: Predictions distributed according to instances containing a seen or an unseen antonym word pair.

Polarity \ Prediction	Neutral	Contradiction	Entailment
Neutral	5	21	0
Contradiction	5	543	3
Entailment	0	3	0
None	8	20	12

Table 4: Contingency table for DAM: Predictions distributed according to the polarity of target word pairs found in the transformed instances.

6.2 Experiment 2: Swapping Antonyms in *Ex Situ* Instances

In this experiment, we use samples E_A and E_{TA}. Swapping antonyms has little effect on the performance of all models, where the biggest drop comes from DAM (0.029 points). However, the CE model performs quite poorly at both samples (0.508 and 0.48 accuracy points on E_A and E_{TA});

this drop in performance, with respect to the *in situ* condition, suggests that the repeated sentence context is too different from the structure of the training instances for the CE model to generalize effectively.

In this condition, we refrain from analyzing the effect of insensitivity, since doing so would require a transformation similar to that in the *in situ* condition, which might add an extra layer of change and the results may turn difficult to interpret.

Unseen Word Pairs Accuracy scores strongly suggest that the models are weak at dealing with unseen antonym pairs (Subset 2 of E_{TA} in Table 1); drops in performance on this subset range from 0.315 up to 0.429 points across the three models. Tests of homogeneity show strong evidence of this weakness for all models (CE: $\chi^2(1) = 15.91$, DAM: $\chi^2(1) = 59.17$, ESIM: $\chi^2(1) = 44.72$). Comparing results on this subset with those of Subset 2 in I_{TA1}, we notice that ESIM and DAM keep similar behavior, but CE seems to be strongly affected by this context type.

Polarity All models perform poorly in the subset of instances where polarity disagrees with gold label of the instance (Subset 3 of E_{TA}), showing that the models' behavior rely on this bias. These results are highly significant (CE: $\chi^2(6) = 34.37$, DAM: $\chi^2(6) = 136.99$, ESIM: $\chi^2(6) = 103.47$). This is further evidence that the models get *confused* with a simple reversal of an antonym pair.

6.3 Experiment 3: Swapping Hypernyms and Hyponyms in *In Situ* Instances

We now study the effect on the robustness of the systems when we swap hypernym and hyponym word pairs in *in situ* instances. Whole sample accuracy scores in Table 1 significantly drop, according to McNemar's tests, by 0.25 (ESIM), 0.285 (CE), and 0.128 (DAM) points when we compare scores on control instances (I_H) with those on transformed instances (I_{TH}). We investigate the role of our target factors on these behaviors.

Insensitivity Around 42% of the instances in I_{TH} (Subset 1) have different gold label from those in I_H. On these instances, the models' results are severely impaired: CE and ESIM models' performances drop to close-to-random (0.271 and 0.315), while DAM decreases by 0.18 points. All models' errors on this subset are strongly associated with failure to change the predicted class (CE:$\chi^2(1) = 90.73$, DAM:$\chi^2(1) = 101.52$, ESIM:$\chi^2(1) = 150.92$). In contrast to the case in Experiment 1, insensitivity acts in detriment of the models' robustness when gold labels change after the transformation.

Unseen Word Pairs Whereas model performance was significantly worse on unseen antonym pairs, this effect is not obvious on the hyponym-hypernym results (Subset 2 of I_{TH}). In fact, all models have a slightly higher accuracy on this subset than overall. Homogeneity tests find no evidence of an association between unseen word pairs and incorrect predictions for any model (CE:$\chi^2(1) = 0.00036, p = 0.98$, DAM:$\chi^2(1) = 0.98, p = 0.32$, ESIM:$\chi^2(1) = 0.178, p = 0.67$). This effect may be explained by the models exploiting information from word embeddings. It has been shown that word embeddings are able to capture hypernymy (Sanchez and Riedel, 2017); thus the models may use this information to generalize to unseen hypernym pairs.

Polarity We find very strong evidence for an association between polarity and class label predicted on sample I_H for all models (CE:$\chi^2(10) = 168.40$, DAM:$\chi^2(10) = 182.76$, ESIM:$\chi^2(10) = 157.76$). However, for sample I_{TH}, only DAM keeps this strong correlation ($\chi^2(14) = 47.71$). In the case of CE, we find weak evidence in favour of this correlation on instances of I_{TH} ($\chi^2(14) = 25.27, p = 0.03$). For ESIM we find no evidence of correlation ($\chi^2(14) = 22.72, p = 0.06$), thus we do not reject the null hypothesis. Polarity's influence can be observed in Subset 3 of I_H (Table 1), where we observe a drop in accuracy for instances whose gold labels do not match the polarity of the word pairs, compared to the accuracy of the whole sample; this means that when the models have polarity as a cue, they improve performance.

6.4 Experiment 4: Swapping Hypernyms and Hyponyms in *Ex Situ* Instances

All models' performance significantly drop ($p < 0.01$) after our transformation by 0.208 (CE), 0.061 (DAM) and 0.195 (ESIM) points, where performance of ESIM is comparable to that of CE on both samples, E_H and E_{TH}. Compared to the *in situ* condition, DAM's performance improves, opposite to CE's and ESIM's behavior.

Insensitivity The drop in performance described above can be partially explained by insensitivity to changes in gold label, since around 93% of the instances in E_{TH} changed gold-label with respect to E_H. We find strong statistical evidence for this hypothesis (CE:$\chi^2(1) = 175.19$, DAM:$\chi^2(1) = 158.62$, ESIM:$\chi^2(1) = 252.27$). However, in the case of DAM, this factor seems to play a small role on its behavior as seen when we compare accuracy on Subset 1 with that of the whole transformed sample.

Insensitivity seems to have a bigger influence on the models when the transformed instances are closer to the training set: Accuracy scores on Subset 1 from I_{TH} are smaller than those on Subset 1 from E_{TH}.

Unseen Word Pairs Similar to the *in situ* condition, our homogeneity tests show no evidence for incorrect predictions being due to unseen word pairs (CE:$\chi^2(1) = 0.35, p = 0.55$, DAM:$\chi^2(1) = 2.43, p = 0.11$, ESIM:$\chi^2(1) = 0.183, p = 0.66$). We posit the same explanation as before: Models may use hypernymy information contained in the embeddings.

Polarity We find statistically high correlation of the models' predictions with the polarity of the word pairs in the instances from both samples, E_H (CE:$\chi^2(10) = 261.77$, DAM:$\chi^2(10) = 312.67$, ESIM:$\chi^2(10) = 176.38$) and E_{TH} (CE:$\chi^2(14) = 56.52$, DAM:$\chi^2(14) = 258.09$, ESIM:$\chi^2(10) = 105.70$). This evidence indicates that all models use, to some extent, the polarity as a feature for predicting class labels.

7 Discussion and Conclusions

Although all three models achieve strong results on the original SNLI development set (CE: 0.782, DAM: 0.854, ESIM: 0.882), each model exhibits particular weaknesses on the transformed training instances. Notably, all perform poorly on I_{TH} instances in which the gold label is changed, with ESIM and CE performing below the level of chance. Thus, on these instances, the models tend to predict the label of the original unaltered training instance and inference in this case is similar to nearest-neighbour prediction.

On the other hand, much better performance is obtained for the DAM and ESIM models on I_{TH} instances containing unseen word pairs, indicating these models have learned to infer hypernym/hyponym relations from information in the pre-trained word embeddings. In contrast, performance on the unseen word pairs in I_{TA1} and E_{TA} suggests that inferring antonymy from the embeddings is more difficult.

Weak performance is seen again on the E_A and E_{TA} instances where the polarity of the antonym pair is not consistent with the gold label. For these cases, the only difference between premise and hypothesis is the antonym pair, and the models tend to fall back on predicting the most frequent gold label seen for that word pair.

One result that remains anomalous is the overall performance of the ESIM model on the whole E_{TH} sample. While this sample contains unseen word pairs and instances in which the gold label changes or is inconsistent with polarity, these effects do not by themselves explain the poor performance overall. Neither is this weakness explained by the *ex situ* structure, in which premise and hypothesis differ by only one word, as performance on the control *ex situ* sample, E_H, is much stronger. The effect, then, appears to be due to an interaction of the *ex situ* structure in combination with the transformation.

In the present work, we have limited ourselves to examining single influences independently. However, there are undoubtedly manifold interactions contributing to model performance. In fact, the complexities of these models (LSTMs, attention mechanisms and MLPs) are specifically intended to capture the interactions between the words in the premise and hypothesis. Further work is required to understand what these interactions are and how they contribute to performance. Fully uncovering these factors in current NLI datasets is a pre-requisite for the construction of more effective resources in the future.

Acknowledgments

We thank Raul Ortiz Pulido and Erick Sanchez Carmona for insightful discussions, Pasquale Minervini for providing the implementations of DAM and ESIM, Pontus Stenetorp for providing valuable feedback on the manuscript, and Johannes Welbl for insightful comments. The first author was recipient of a scholarship from CONACYT. This work was supported by an Allen Distinguished Investigator Award and the EU H2020 SUMMA project (grant agreement number 688139).

References

Ethem Alpaydin. 2010. *Introduction to Machine Learning*. The MIT Press, 2nd edition.

Homa B. Hashemi and Rebecca Hwa. 2016. An evaluation of parser robustness for ungrammatical sentences. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1765–1774. https://aclweb.org/anthology/D16-1182.

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Avignon, France, pages 23–32. http://www.aclweb.org/anthology/E12-1004.

Lucy P. Birkett and Nicholas E. Newton-Fisher. 2011. How abnormal is the behaviour of captive, zoo-living chimpanzees? *PLOS ONE* 6(6):1–7. https://doi.org/10.1371/journal.pone.0020101.

Tolga Bolukbasi, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam T. Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, Curran Associates, Inc., pages 4349–4357.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 632–642. http://aclweb.org/anthology/D15-1075.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1657–1668. http://aclweb.org/anthology/P17-1152.

Mark Craven and Jude W. Shavlik. 1996. Extracting tree-structured representations of trained networks. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, MIT Press, pages 24–30.

Kate Crawford and Ryan Calo. 2016. There is a blind spot in ai research. *Nature* 538(7625).

Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. 2009. Recognizing textual entailment: Rational, evaluation and approaches. *Natural Language Engineering* 15(4):i–xvii. https://doi.org/10.1017/S1351324909990209.

Ido Dagan and Oren Glickman. 2004. Probabilistic textual entailment: generic applied modeling of language variability. In *PASCAL Workshop on Learning Methods for Text Understanding and Mining*. Grenoble, France.

W. Frank Epling and W. David Pierce. 1986. The basic importance of applied behavior analysis. *The Behavior Analyst* 9(1):89–99. https://doi.org/10.1007/BF03391932.

Pierre Isabelle, Colin Cherry, and George Foster. 2017. A challenge set approach to evaluating machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 2476–2486. https://www.aclweb.org/anthology/D17-1262.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 2011–2021. https://www.aclweb.org/anthology/D17-1214.

Jonathan K. Kummerfeld, David Hall, James R. Curran, and Dan Klein. 2012. Parser showdown at the wall street corral: An empirical investigation of error types in parser output. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, Jeju Island, Korea, pages 1048–1059. http://www.aclweb.org/anthology/D12-1096.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. Understanding neural networks through representation erasure. *CoRR* abs/1612.08220. http://arxiv.org/abs/1612.08220.

Peter LoBue and Alexander Yates. 2011. Types of common-sense knowledge needed for recognizing textual entailment. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*. Association for Computational Linguistics, Stroudsburg, PA, USA, HLT '11, pages 329–334. http://dl.acm.org/citation.cfm?id=2002736.2002805.

Bill Maccartney. 2009. *Natural Language Inference*. Ph.D. thesis, Stanford, CA, USA. AAI3364139.

J.H. McDonald. 2014. *Handbook of Biological Statistics (3rd ed.)*. Sparky House Publishing, Baltimore, Maryland.

Joy Mench. 1998. Why it is important to understand animal behavior. *ILAR Journal* 39(1):20–26. https://doi.org/10.1093/ilar.39.1.20.

Saif M. Mohammad, Bonnie J. Dorr, Graeme Hirst, and Peter D. Turney. 2013. Computing lexical contrast. *Computational Linguistics* 39(3):555–590.

T. E. Oliphant. 2007. Python for scientific computing. *Computing in Science Engineering* 9(3):10–20. https://doi.org/10.1109/MCSE.2007.58.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 2249–2255. https://aclweb.org/anthology/D16-1244.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA, KDD '16, pages 1135–1144. https://doi.org/10.1145/2939672.2939778.

Mark Sammons, V. G. Vinod Vydiswaran, and Dan Roth. 2010. "ask not what textual entailment can do for you...". In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '10, pages 1199–1208. http://dl.acm.org/citation.cfm?id=1858681.1858803.

Ivan Sanchez and Sebastian Riedel. 2017. How well can we predict hypernyms from word embeddings? a dataset-centric analysis. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, Valencia, Spain, pages 401–407. http://www.aclweb.org/anthology/E17-2064.

Skipper Seabold and Josef Perktold. 2010. Statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*.

Dilip Soman. 2001. Effects of payment mechanism on spending behavior: The role of rehearsal and immediacy of payments. *Journal of Consumer Research* 27(4):460–474.

Aaron Steven White, Pushpendre Rastogi, Kevin Duh, and Benjamin Van Durme. 2017. Inference is everything: Recasting semantic resources into a unified evaluation framework. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Asian Federation of Natural Language Processing, Taipei, Taiwan, pages 996–1005. http://www.aclweb.org/anthology/I17-1100.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2017. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 2979–2989. https://www.aclweb.org/anthology/D17-1323.

Assessing Language Proficiency from Eye Movements in Reading

Yevgeni Berzak
MIT BCS
berzak@mit.edu

Boris Katz
MIT CSAIL
boris@mit.edu

Roger Levy
MIT BCS
rplevy@mit.edu

Abstract

We present a novel approach for determining learners' second language proficiency which utilizes behavioral traces of eye movements during reading. Our approach provides standalone eyetracking based English proficiency scores which reflect the extent to which the learner's gaze patterns in reading are similar to those of native English speakers. We show that our scores correlate strongly with standardized English proficiency tests. We also demonstrate that gaze information can be used to accurately predict the outcomes of such tests. Our approach yields the strongest performance when the test taker is presented with a suite of sentences for which we have eyetracking data from other readers. However, it remains effective even using eyetracking with sentences for which eye movement data have not been previously collected. By deriving proficiency as an automatic byproduct of eye movements during ordinary reading, our approach offers a potentially valuable new tool for second language proficiency assessment. More broadly, our results open the door to future methods for inferring reader characteristics from the behavioral traces of reading.

1 Introduction

It is currently estimated that over 1.5 billion people are learning English as a Second Language (ESL) worldwide. Their learning progress is commonly evaluated with classroom tests prepared by language instructors, quizzes in language learning software such as Duolingo and Rosetta Stone, and by official standardized language proficiency tests such as TOEFL, IELTS, MET and others. In "high stakes" scenarios, official language proficiency tests are the de-facto standards for language assessment; they are accepted by educational and professional institutions, and are taken by millions of language learners every year (for example, in 2016 over three million people took the IELTS test (IELTS, 2017)). These tests probe language proficiency based on performance on various linguistic tasks, including grammar and vocabulary exams, reading and listening comprehension questions, as well as essay writing and speaking assignments.

Despite their ubiquity, traditional approaches to language proficiency testing have several drawbacks. First, such tests are typically prepared manually and require extensive resources for test development. Moreover, their validity can be undermined by test specific training, prior knowledge of the evaluation mechanisms (Powers et al., 2002), as well as plain cheating via unauthorized access to test materials. Further, the utilized testing and evaluation methodologies vary across different tests, and test materials are in most cases inaccessible to the research community. Perhaps most crucially, the reliance of these tests on the end products of linguistic tasks makes it challenging to study learners' language processing patterns and the difficulties they encounter in real time.

In this work we propose a novel methodology for language proficiency assessment which marks a significant departure from traditional language proficiency tests and addresses many of their drawbacks. In our approach, we determine language proficiency from broad coverage analysis of *eye movements during reading of free-form text in a foreign language*, a special case of the general problem of inferring comprehender characteristics and cognitive state from the measurable traces of real-time language processing. Our framework does not require the test taker to prepare for the test or to perform any hand-crafted linguistic tasks, but simply to attentively read an arbitrary set of sentences. To the best of our knowledge, this work is the first to propose and implement such an approach, yielding a novel language proficiency evaluation scheme which relies solely

Proceedings of NAACL-HLT 2018, pages 1986–1996
New Orleans, Louisiana, June 1 - 6, 2018.

on ordinary reading.

Our framework builds on previous research in psycholinguistics demonstrating that the eyetracking record reflects how readers interact with the text and how language processing unfolds over time (Frazier and Rayner, 1982; Rayner, 1998; Rayner et al., 2012). In particular, it has been shown that key aspects of the reader's characteristics and cognitive state, such as mind wandering during reading (Reichle et al., 2010), dyslexia (Rello and Ballesteros, 2015) and native language (Berzak et al., 2017) can be inferred from their gaze record. Despite these advances, the potential of the rich and highly informative behavioral signal obtainable from human reading for automated inference about readers, and specifically about their linguistic proficiency has thus far been largely unutilized.

Here, we first introduce EyeScore, an independent measure of ESL proficiency which reflects the extent to which a learner's English reading patterns resemble those of native speakers. Second, we present a regression model which uses gaze features to predict the learner's scores on specific external proficiency tests. We address each of our tasks in two data regimes: *Fixed Text*, which requires eyetracking training data for the specific sentences presented to the test taker, as well as the more general and challenging *Any Text* regime, where the test taker is presented with arbitrary sentences for which no previous eyetracking data is available. To enable prediction mechanisms in both regimes, we utilize previously proposed gaze features, and develop new linguistically and psychologically motivated feature sets which capture the interaction between eye movements and linguistic properties of the text.

We demonstrate the effectiveness of our approach via score comparison to standardized English proficiency tests. Our primary benchmark test, taken in lab by 145 ESL participants, are the grammar and listening sections of the Michigan English Test (MET) whose scores range from 0 to 50. EyeScore yields 0.5 Pearson's correlation to MET in the Fixed Text regime, and 0.48 in the Any Text regime. Our regression model for predicting MET scores from eye movement features obtains a correlation of 0.7 and a Mean Absolute Error (MAE) of 3.31 points in the Fixed Text regime, and 0.49 correlation and 4.11 MAE in the Any Text regime. Our results are substantially stronger compared to a baseline using only raw reading speed, and are reasonably close to correlations among traditional proficiency tests. These outcomes confirm the promise of the proposed methodology to reliably measure language proficiency.

This paper is structured as follows. Section 2 describes the data and the experimental setup. In section 3 we delineate our feature sets for charactering eye movements in human reading. Section 4 introduces EyeScore, a second language proficiency metric which is based on similarity of reading patterns to native speakers. In section 5 we use eyetracking patterns to predict scores on MET and TOEFL. In section 6 we survey related work. Finally, we conclude and discuss future work in section 7.

2 Experimental Setup

Our study uses the dataset of eye movement records and English proficiency scores introduced in Berzak et al. (2017)[1], which we describe here in brief. The dataset contains gaze recordings of 37 native English speakers and 145 ESL speakers belonging to four native language backgrounds: 36 Chinese, 36 Japanese, 36 Portuguese and 37 Spanish. Participants were presented with free-form English sentences appearing as one-liners. To encourage attentive reading each sentence was followed by a yes/no comprehension question. During the experiment participants held a controller with buttons for indicating sentence reading completion and answering the sentence comprehension questions. Participants' eye movements were recorded using a desktop mount EyeLink 1000 eyetracker (SR Research) at a sampling rate of 1000Hz.

2.1 Procedure and Reading Materials

An experimental trial for a sentence starts with a presentation of a target circle at the center left of a blank screen. A 300ms fixation on this circle triggers a one-liner sentence on a new screen starting at the same location. After completing reading the sentence, participants are presented with the letter Q on a blank screen. A 300ms fixation on this letter triggers a question about the sentence on a new screen. Participants provide a yes/no answer to the question and are subsequently informed if

[1]The data was collected under IRB approval, and all the participants provided written informed consent.

they answered correctly. The first trial of the experiment was presented to familiarize participants with the experimental setup, and is discarded from the analysis.

Each participant read a total of 156 English sentences, randomly drawn from the Wall Street Journal Penn Treebank (WSJ-PTB) (Marcus et al., 1993). The maximal sentence length was set to 100 characters, yielding an average sentence length of 11.4 words. All the sentences include the manual PTB annotations of POS tags (Santorini, 1990) and phrase structure trees, as well as Google universal POS tags (Petrov et al., 2012) and dependency trees obtained from the Universal Dependency Treebank (UDT) (McDonald et al., 2013).

2.2 Experimental Regimes

Half of the 156 sentences presented to each participant belong to the *Fixed Text* regime, and the other half belong to the *Any Text* regime. Sentences from the two regimes were interleaved randomly and presented to all participants in the same order.

Fixed Text In this regime, all the participants read the same suite of 78 pre-selected sentences (900 words). The Fixed Text regime supports *token-level* comparisons of reading patterns for specific words in the same contexts across readers. It enables the construction of a proficiency test which relies on a fixed battery of reading materials for which previous eyetracking data was collected.

Any Text In the second, Any Text regime, different participants read different sets of 78 sentences each (880 words on average). This regime generalizes the Fixed Text scenario; predicting reader characteristics in this regime requires formulating *type-level* abstractions that would allow meaningful comparisons of reading patterns across different sentences. It corresponds to a proficiency test in which the sentences presented to the test taker are completely arbitrary, and no prior eyetracking data is available for them.

2.3 Standardized English Tests

We use participants' performance on the Michigan English Test (MET) and TOEFL as external benchmarks of their English proficiency.

Michigan English Test (MET) Our primary indicator of English proficiency is the listening and grammar sections of the MET (Form-B), which were administered by Berzak et al. (2017) in-lab, and taken by all the 145 non-native participants upon completion of the reading experiment. The test has a total of 50 multiple choice questions, comprising 20 listening comprehension questions and 30 written grammar questions. The test score is computed as the number of correct answers for these questions, with possible scores ranging from 0 to 50. The mean MET score in the dataset is 41.46 (std 6.27).

TOEFL Berzak et al. (2017) also collected self-reported scores on the most recently taken official English proficiency test, which we use here as a secondary evaluation benchmark. We focus on the most commonly reported test, the TOEFL-iBT whose scores range from 0 to 120. We take into account only test results obtained less than four years prior to the experiment, yielding 33 participants. We sum the scores of the reading and listening sections of test, with a total possible score range of 0 to 60. In cases where participants reported only the overall score, we divided that score by two. We further augment this data with 20 participants who took the TOEIC Listening and Reading test within the same four years range, resulting in a total of 53 external proficiency scores. The TOEIC scores were converted to the TOEFL scale by fitting a third degree polynomial on an unofficial score conversion table[2] between the tests. The converted scores were then divided by two. Henceforth we refer to both TOEFL-iBT and TOEIC scores converted to TOEFL-iBT scale as TOEFL scores. The mean TOEFL score is 47.6 (std 9.55). The Pearson's r correlation between the TOEFL and MET scores in the dataset is 0.74.

2.4 Data Split

We divide the ESL speakers into training/development and test sets in the following manner. For MET, we split our 145 ESL participants into a training/development set of 88 participants and a test set of 57 participants. The test set consists of an entire held out native language – 36 speakers of Portuguese – as well as 7 participants randomly sampled from each of the remaining three native languages. Our test set is thus particularly challenging due to the large fraction of participants belonging to the held out language, a design which emphasizes

[2] http://theedge.com.hk/conversion-table-for-toefl-ibt-pbt-cbt-tests/ Although both TOEFL and TOEIC are administered by the same company (ETS), to the best of our knowledge there is no publicly available official conversion table between the two tests.

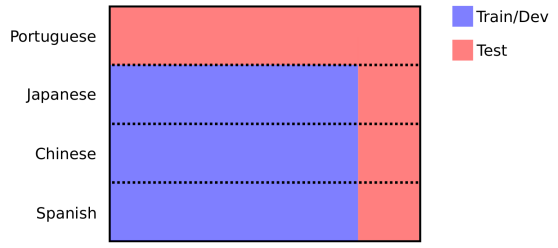


Figure 1: Illustration of the data split for MET into a training/development set (88 participants) and a test set (57 participants).

generalization to language learner populations which are not part of the training set. Figure 1 presents a schematic overview of our MET split. For TOEFL, due to the limited available data, in Section 4 we report EyeScore correlations for all the 53 test takers, and in Section 5 we perform regression experiments using leave-one-out cross validation.

3 Eye Movement Features

In order to capture behavioral psycholinguistic traces of language proficiency we utilize several linguistically and psychologically motivated feature representations of eye movements in reading. We include features introduced in prior work (see Words in Fixed Context and Syntactic Clusters (Berzak et al., 2017)) as well as newly developed feature sets (see Word Property Coefficients and Transitions). All our features rely on the well established division of gaze trajectories into fixations (stops) and saccades (movements between fixations) that characterizes human reading (Rayner, 1998).

Our fixation based features make use of several standard metrics of fixation times, defined below.

- *First Fixation duration (FF)* Duration of the first fixation on a word.
- *First Pass duration (FP)* Time spent from first entering a word to first leaving it (including re-fixations within the word).
- *Total Fixation duration (TF)* The sum of all fixation times on a word.
- *Regression Path duration (RP)* Time from first entering a word until proceeding to its right.

Our feature sets are divided into two groups. The first group consists of type-level features, applicable both in the Any Text and Fixed Text regimes. The second group of feature sets is token-based and can be extracted only in the Fixed Text regime, because it presupposes the same textual input for all participants.

3.1 Type-Level Features

Word Property Coefficients (WP-Coefficients)

This new feature set quantifies the influence of three key word characteristics on reading times of individual readers: word length, word frequency and surprisal. The last measures the difficulty of processing a word in a sentence (Hale, 2001; Levy, 2008), and is defined as its negative log probability given a sentential context:

$$surprisal(w_i|w_{1...i-1}) = -\log(w_i|w_{1...i-1}) \quad (1)$$

In the reading literature, these three characteristics were suggested as the most prominent linguistic factors influencing word reading times (e.g. Inhoff and Rayner, 1986; Rayner and Well, 1996; Pollatsek et al., 2008; Kliegl et al., 2004; Rayner et al., 2004, 2011; Smith and Levy, 2013; Luke and Christianson, 2016); whereby longer, less frequent and contextually less predictable words are fixated longer.

To derive this feature set, we measure length as the number of characters in the word. Word (log) frequencies are obtained from the BLLIP-WSJ corpus (Charniak et al., 2000). Estimates of surprisal are obtained from a trigram language model with Chen and Goodman's modified Kneser-Ney smoothing trained on the BLLIP-WSJ using SRILM (Stolcke et al., 2002). We then fit for each participant four regression models that use these three word characteristics to predict the word's raw FF, FP, TF and RP durations. The regression models are fitted using Ordinary Least Squares (OLS). We also train a logistic regression model for predicting word skips. Finally, we extract the weights and intercepts of these models and encode them as features. As each of the five models has three coefficients and one intercept term, the resulting WP-Coefficients feature set has 20 features.

Syntactic Clusters (S-Clusters)

Following Berzak et al. (2017), we extract average word reading times clustered by POS tags and

syntactic functions. We utilize three metrics of reading times, FF, FP and TF durations. We then cluster words according to three types of syntactic criteria, Google Universal POS tags, PTB POS tags, and the syntactic function label of the word to its head word. To derive the feature set, we average the word fixation times of each cluster. An example of an S-Cluster feature is the average TF duration for words with the PTB POS tag DT. We take into account only cluster labels that appear at least once in the reading input of all the participants, yielding a total of 312 S-Clusters features in the Fixed Text regime. In the Any Text regime we obtain 156 S-Clusters features for MET and 165 S-Clusters features for TOEFL.

3.2 Token-Level Features

Transitions

Transitions is a new feature set which summarizes the sequence of saccades between words in a sentence. Given a sentence with n words, we construct an $n \times n$ matrix T. A matrix entry $t_{i,j}$ records the number of saccades whose launch site falls within word i and landing site falls within word j. With a total of 11,616 possible transitions in the Fixed Text sentences, the resulting feature set contains 9,077 features with a non-zero value for at least one participant for MET, and 8,132 such features for TOEFL.

Words in Fixed Context (WFC)

This feature set was previously used in Berzak et al. (2017) and consists of reading times for words within fixed contexts. We extract FP and TF durations for the 900 words in the Fixed Text sentences, resulting in a total of 1,800 WFC features.

4 English Proficiency Scoring Based on Eye Movements in Reading

We hypothesize that language proficiency influences the way that learners process a second language, which in turn will be reflected in eye movement patterns in reading. Specifically, we propose to examine whether the more proficient is an ESL learner, the more similar are their reading patterns to those of native English speakers. We operationalize the notion of native-like reading in the following manner. First, given a feature representation of choice and a dataset D comprising ESL learners D_{L2} and native speakers D_{L1} we Z score each feature in D using a Z scaler derived from D_{L2}. We then obtain a prototype feature vector of native reading v_{L1} by averaging the feature vectors of the native speakers.

$$v_{L1} = \frac{1}{|D_{L1}|} \sum_{y \in D_{L1}} v_y \tag{2}$$

Finally, we obtain an eyetracking based proficiency score of an ESL learner by computing the cosine similarity of their feature vector to the native reading prototype. Hereafter we refer to this measure as EyeScore.

$$EyeScore_{y \in D_{L2}} = \frac{v_y \cdot v_{L_1}}{\|v_y\| \|v_{L1}\|} \tag{3}$$

Reading Speed Normalization To reduce bias towards fast readers, the feature representations used for Eyescore are normalized to be invariant to the reading speed of the participant. Specifically, for the S-Clusters and WFC feature sets we follow the normalization procedure of Berzak et al. (2017), where for a given participant, the reading time of a word w_i according to a fixation metric M is normalized by $S_{M,C}$, the metric's fixation time per word in the linguistic context C:

$$S_{M,C} = \frac{1}{|C|} \sum_{w \in C} M_w \tag{4}$$

The linguistic context is defined as the surrounding sentence in the Fixed Text regime, and the entire textual input in the Any Text regime. The normalized fixation time is then obtained as:

$$Mnorm_{w_i} = \frac{M_{w_i}}{S_{M,C}} \tag{5}$$

For the WC-Coefficients features we take into account only the 15 model coefficients, and omit the 5 intercept features which capture the reading speed of the participant. Finally, we also normalize the Transitions features matrix T by the total number of saccades in the sentence to obtain T_{norm} in which $\sum_{i,j} t_{norm_{i,j}} = 1$.

4.1 Correlation with MET and TOEFL

We evaluate the ability of EyeScore to capture language proficiency by comparing it against our two external proficiency tests, MET and TOEFL. Table 1 presents the Pearson's r correlation of EyeScore with MET and TOEFL for the feature sets described in section 3 using the MET training/development set and all the participants who took TOEFL.

	MET		TOEFL	
Features	Fixed	Any	Fixed	Any
Reading Speed	0.28	0.27	0.15	0.13
WP-Coefficients	0.38	0.37	0.21	0.13
S-Clusters	0.45	**0.48**	0.50	**0.45**
Transitions	0.45	NA	0.44	NA
WFC	**0.50**	NA	**0.54**	NA

Table 1: Pearson's r of EyeScore for different feature sets with MET (training/development set, 88 participants) and TOEFL (all 53 participants). Fixed denotes the Fixed Text regime in which all the participants read the same sentences, and Any denotes the Any Text regime where different readers read different sentences.

The strongest correlations, 0.5 for MET and 0.54 for TOEFL, are obtained in the Fixed Text regime using the WFC features. This outcome confirms the effectiveness reading time comparisons when the presented sentences are shared across participants. To illustrate the quality of this result, Figure 2 presents a comparison of EyeScore and MET scores in the Fixed Text and WFC features setup. We further note good performance of the Transitions and S-Clusters features in this regime across both proficiency tests. The strongest performance in the Any Text regime is obtained using the S-Clusters features, yielding 0.48 correlation with MET and 0.45 correlation with TOEFL. These results are competitive with the WFC feature set in the Fixed Text regime, suggesting that reliable EyeScores can be obtained even when no prior eyetracking data is available for the sentences presented to the test taker.

In order to contextualize the correlations obtained with the EyeScore approach, we first compare our results to raw reading speed, an informative baseline which does not rely on eyetracking. EyeScore substantially outperforms this baseline for nearly all the feature sets on both MET and TOEFL, clearly showing the benefit of eye movement information for our task. Next, we consider possible upper bounds for our correlations. While obtaining such upper bounds is challenging, we can use correlations between different traditional standardized proficiency tests as informative reference points. First, as mentioned previously, in our dataset the MET and reported TOEFL scores have a Pearson's r correlation of 0.74. We further note an external study conducted by the testing company Education First (EF) which measured the correlation of their flagship standardized English

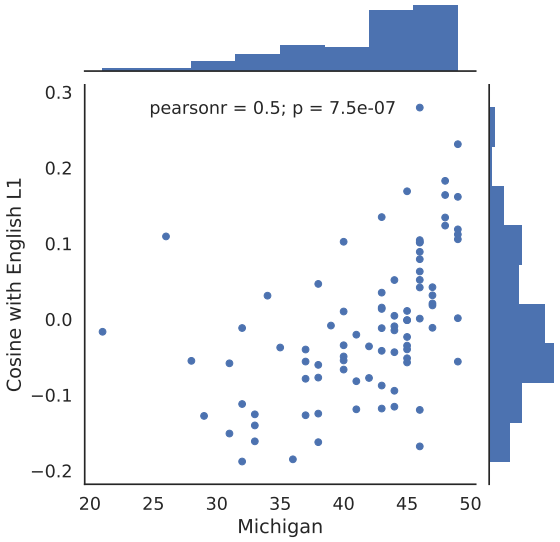


Figure 2: Comparison of MET (training/development set, 88 participants) with EyeScore using Words in Fixed Context (WFC) features in the Fixed Text regime.

proficiency test EFSET-PLUS with TOEFL-iBT (Luecht, 2015). Using 384 participants who took both tests, the study found a Pearson's r of 0.63 for the reading comprehension and 0.69 for the listening comprehension sections of these tests. Despite the radical difference of our testing methodology, our strongest feature sets obtain rather competitive results relative to these correlations, further strengthening the evidence for the ability of our approach to capture language proficiency.

5 Predicting Performance on MET and TOEFL

In section 4 we introduced EyeScore as an independent metric of language proficiency which is based on eye movements during reading. Here, we examine whether eye movements can also be used to explicitly *predict* the performance of participants on specific external standardized language proficiency tests. This task is of practical value for development of predictive tools for standardized proficiency tests, and constitutes an alternative framework for studying the relevance of eye movement patterns in reading to language proficiency.

To address this task, we use Ridge regression to predict overall scores on an external proficiency test from eye movement features in reading. The model parameters θ are obtained by minimizing

	MET				TOEFL			
	Fixed		Any		Fixed		Any	
Features	r	MAE	r	MAE	r	MAE	r	MAE
Reading Speed	0.27	4.58	0.24	4.62	0.09	7.92	0.06	7.96
WP-Coefficients	0.43	4.11	0.44	4.14	0.34	7.76	0.31	**7.49**
S-Clusters	0.56	3.87	**0.49**	**4.11**	**0.55**	7.45	**0.50**	7.76
Transitions	0.52	3.93	NA	NA	0.38	7.11	NA	NA
WFC	**0.70**	**3.31**	NA	NA	0.50	**6.68**	NA	NA

Table 2: Pearson's r and Mean Absolute Error (MAE) for prediction of MET scores (test set, 57 participants) and TOEFL scores (leave-one-out cross validation, all 53 participants) from eye movement patterns in reading. We consider two baselines which do not use eyetracking information: (1) the average proficiency score in the training set, which yields 4.82 MAE on MET and 8.29 MAE on TOEFL, and (2) the reading speed of the participant.

the following loss objective:

$$\sum_i (y_i - \theta \cdot f(x_i))^2 + \lambda \|\theta\|_2^2 \qquad (6)$$

where y_i is a participant's test score, x_i is their eye movement record, and $f(x_i)$ are the extracted eye movement features. To calibrate the model with respect to native English speakers, we augment each training set $D_{L2_{tr}}$ with the group of 37 native speakers D_{L1} whose proficiency scores are assigned to the maximum grade of the respective test (50 for MET and 60 for TOEFL)[3]. Based on MET performance on the train/dev set, the features used for predicting scores on both tests are not normalized for speed[4]. As a preprocessing step, we fit a Z scaler for each feature using the ESL participants in the training set, and apply it to all the participants in the training and test sets.

Results

We evaluate prediction accuracy using Pearson's r and Mean Absolute Error (MAE) from the true proficiency test scores. The λ parameter for MET is optimized for MAE on 10 fold cross validation within the training/development set. For TOEFL, which has a relatively small number of participants, we report results on leave-one-out cross validation with λ set to 1.

Table 2 presents the results for both proficiency tests. We consider two baselines; the first is assigning all test set participants with the average score of the training participants. This baseline yields an MAE of 4.82 on MET and 8.29 on TOEFL. The second baseline uses reading speed as the sole feature for prediction. In all cases, our eyetracking based features outperform the average score and reading speed baselines.

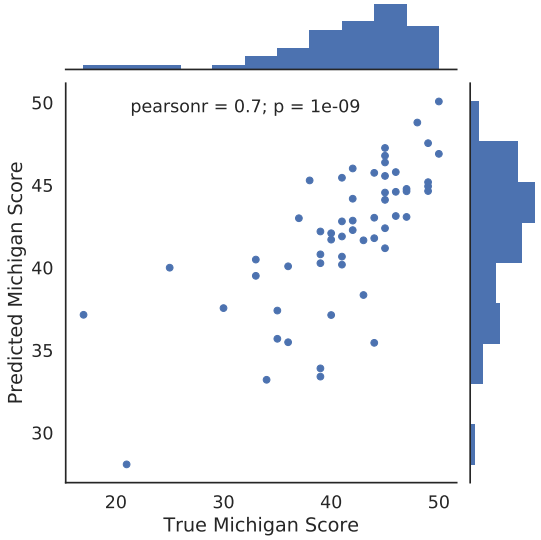


Figure 3: Comparison of MET scores (test set, 57 participants) with predicted MET scores using Words in Fixed Context (WFC) eye movement features in the Fixed Text regime.

The performance of the different feature sets is in most cases consistent across the two proficiency tests and is largely in line with the correlations of EyeScore reported in Table 1. Similarly to the EyeScore outcomes, the best performance in the Fixed Text regime is obtained using the WFC feature set, with a Pearson's r of 0.7 and MAE of 3.31 for MET. This result is highly competitive with correlations between different standardized English proficiency tests. Figure 3 depicts a com-

[3]Our experiments on the training/development set indicate that this training data augmentation step leads in most cases to improved regression performance.

[4]We note that in line with the low correlation of reading speed with TOEFL, speed normalized features tend to be better predictors of TOEFL scores, obtaining r 0.59 and MAE 6.47 with WFC features in the Fixed Text regime, and r 0.58 and MAE 7.19 with S-Clusters in the Any Text regime.

parison between MET scores and our MET predictions in this setup. On TOEFL, WFC features obtain the strongest MAE of 6.68, while S-Clusters have a higher r coefficient of 0.55.

In the Any Text regime, differently from EyeScore, we obtain comparable results for the S-Clusters and WP-Coefficients feature sets. Overall, the improvements of both feature sets over the baselines in the Any Text regime further support the ability of type-level features to generalize the task of language proficiency prediction to arbitrary sentences.

6 Related Work

Our work lies on the intersection of language proficiency assessment, second language acquisition (SLA), the psychology of reading and NLP. Automated language proficiency assessment from free-form linguistic performance has been studied mainly in language *production* (Dikli, 2006; Williamson, 2009; Shermis and Burstein, 2013). Over the past several decades, multiple essay and speech scoring systems have been developed for learner language using a wide range of linguistically motivated feature sets (e.g. Lonsdale and Strong-Krause, 2003; Landauer, 2003; Xi et al., 2008; Yannakoudakis et al., 2011). Some of these systems have been deployed in official language proficiency tests, for example the *e-rater* essay scoring system (Attali and Burstein, 2004) used in TOEFL (Ramineni et al., 2012). While this line of work focuses on assessment of language production, here we introduce and address for the first time automated language assessment during online language *comprehension*.

In SLA, there has been considerable interest in eyetracking, where studies have mostly focused on controlled experiments examining processing of specific linguistic phenomena such as syntactic ambiguities, cognates and idioms (Dussias, 2010; Roberts and Siyanova-Chanturia, 2013). A notable exception is (Cop et al., 2015) who used free-form reading to study differences in fixation times and saccade lengths between native and non-native readers. Our work also adopts broad coverage analysis of reading patterns, which we use to formulate predictive models of language proficiency.

Our study draws on a large body of work in the psychology of reading (see Rayner, 1998; Rayner et al., 2012, for overview) which has suggested that eye movement patterns during reading are systematically influenced by a broad range of linguistic characteristics of the text, and reflect how readers mentally engage with the text (Frazier and Rayner, 1982; Rayner and Frazier, 1989; Reichle et al., 1998; Engbert et al., 2005; Demberg and Keller, 2008; Reichle et al., 2009; Levy et al., 2009, among many others). Prior work on reading has also demonstrated that gaze provides valuable information about various characteristics of the reader and their cognitive state. For example, Reichle et at. (2010) have shown that eye movement patterns are categorically different in attentive versus mindless reading. In Rello and Ballesteros (2015) eye movements were used to distinguish between readers with and without dyslexia. Berzak et al. (2017) collected the dataset used in our work and used it to predict the first language of non-native English readers from gaze. We build on these studies to motivate our task and design feature representations which encode linguistic factors known to affect the human reading process.

Related work in NLP developed predictive models of reading times in reading of free-form text (e.g. Nilsson and Nivre, 2009; Hara et al., 2012; Hahn and Keller, 2016). In a complementary vein, eyetracking signal has been used for linguistic annotation tasks such as POS tagging (Barrett and Søgaard, 2015a; Barrett et al., 2016) and prediction of syntactic functions (Barrett and Søgaard, 2015b). Both lines of investigation provide further evidence for the tight interaction between eye movements and linguistic properties of the text, which we leverage in our work for inference about the linguistic knowledge of the reader.

7 Conclusion and Discussion

We present a novel approach for automated assessment of language proficiency which relies on eye movements during reading of free-form text. Our EyeScore test captures the similarity of language learners' gaze patterns to those of native speakers, and correlates well with the standardized tests MET and TOEFL. A second variant of our approach accurately predicts participants' scores on these two tests. To the best of our knowledge, the proposed framework is the first proof-of-concept for a system which utilizes eyetracking to measure linguistic ability.

In future work, we plan to extend the analysis of the validity and consistency of our approach, and further explore its applications for language

proficiency evaluation. In particular, we will examine the impact of factors that can undermine the validity of language proficiency tests, such as test specific training, familiarity with the evaluation system's features (Powers et al., 2002), and cheating via unauthorized prior access to test materials. Since participants are less likely to be able to manipulate their eye movements in an informed and systematic manner—readers are generally not even aware that their eye movements are saccadic—and since our test can be performed on arbitrary sentences, we expect it to be robust to prior exposure to the test materials and testing methodology. We will further study the consistency of our scores for repeated tests by the same participants. A preliminary split-half analysis indicates that eyetracking based scores are expected to be highly consistent across tests. Finally, our approach can be combined with traditional proficiency testing methodologies, whereby gaze will be recorded while the participant is taking a standardized language proficiency test. This will enable developing novel approaches to language proficiency assessment which will integrate task based performance with real time monitoring of cognitive and linguistic processing.

Acknowledgments

This material is based upon work supported in part by the Center for Brains, Minds, and Machines (CBMM), funded by NSF STC award CCF-1231216.

References

Yigal Attali and Jill Burstein. 2004. Automated essay scoring with e-rater® v. 2.0. *ETS Research Report Series* 2004(2).

Maria Barrett, Joachim Bingel, Frank Keller, and Anders Søgaard. 2016. Weakly supervised part-of-speech tagging using eye-tracking data. In *ACL*. volume 2, pages 579–584.

Maria Barrett and Anders Søgaard. 2015a. Reading behavior predicts syntactic categories. In *CoNLL*. pages 345–349.

Maria Barrett and Anders Søgaard. 2015b. Using reading behavior to predict grammatical functions. In *Proceedings of the Sixth Workshop on Cognitive Aspects of Computational Language Learning*. pages 1–5.

Yevgeni Berzak, Chie Nakamura, Suzanne Flynn, and Boris Katz. 2017. Predicting native language from gaze. In *ACL*. pages 541–551.

Eugene Charniak, Don Blaheta, Niyu Ge, Keith Hall, John Hale, and Mark Johnson. 2000. BLLIP 1987-89 WSJ corpus release 1. *Linguistic Data Consortium, Philadelphia* 36.

Uschi Cop, Denis Drieghe, and Wouter Duyck. 2015. Eye movement patterns in natural reading: A comparison of monolingual and bilingual reading of a novel. *PLOS ONE* 10(8):1–38.

Vera Demberg and Frank Keller. 2008. Data from eye-tracking corpora as evidence for theories of syntactic processing complexity. *Cognition* 109(2):193–210.

Semire Dikli. 2006. An overview of automated scoring of essays. *The Journal of Technology, Learning and Assessment* 5(1).

Paola E Dussias. 2010. Uses of eye-tracking data in second language sentence processing research. *Annual Review of Applied Linguistics* 30:149–166.

Ralf Engbert, Antje Nuthmann, Eike M Richter, and Reinhold Kliegl. 2005. Swift: a dynamical model of saccade generation during reading. *Psychological Review* 112(4):777.

Lyn Frazier and Keith Rayner. 1982. Making and correcting errors during sentence comprehension: Eye movements in the analysis of structurally ambiguous sentences. *Cognitive Psychology* 14(2):178–210.

Michael Hahn and Frank Keller. 2016. Modeling human reading with neural attention. In *EMNLP*. pages 85–95.

John Hale. 2001. A probabilistic Earley parser as a psycholinguistic model. In *NAACL*. Association for Computational Linguistics, pages 1–8.

Tadayoshi Hara, Daichi Mochihashi, Yoshinobu Kano, and Akiko Aizawa. 2012. Predicting word fixations in text with a CRF model for capturing general reading strategies among readers. In *Proceedings of the 1st Workshop on Eye-tracking and Natural Language Processing*. pages 55–70.

IELTS. 2017. International English language testing system — Wikipedia, the free encyclopedia. Online; accessed November 2017. https://en.wikipedia.org/wiki/International_English_Language_Testing_System.

Albrecht Werner Inhoff and Keith Rayner. 1986. Parafoveal word processing during eye fixations in reading: Effects of word frequency. *Perception & Psychophysics* 40(6):431–439.

Reinhold Kliegl, Ellen Grabner, Martin Rolfs, and Ralf Engbert. 2004. Length, frequency, and predictability effects of words on eye movements in reading. *European Journal of Cognitive Psychology* 16(1-2):262–284.

Thomas K Landauer. 2003. Automated scoring and annotation of essays with the Intelligent Essay Assessor. *Automated essay scoring: A crossdisciplinary perspective* .

Roger Levy. 2008. Expectation-based syntactic comprehension. *Cognition* 106(3):1126–1177.

Roger Levy, Klinton Bicknell, Tim Slattery, and Keith Rayner. 2009. Eye movement evidence that readers maintain and act on uncertainty about past linguistic input. *Proceedings of the National Academy of Sciences* 106(50):21086–21090.

Deryle Lonsdale and Diane Strong-Krause. 2003. Automated rating of ESL essays. In *Proceedings of the HLT-NAACL Workshop on Building Educational Applications using Natural Language Processing - Volume 2*. Association for Computational Linguistics, pages 61–67.

Richard M Luecht. 2015. EFSET PLUS - TOEFL iBT correlation study report. https://www.efset.org/research/~/media/centralefcom/efset/pdf/EFSET_TOEFL_correlational_report_Sep_v1.pdf.

Steven G Luke and Kiel Christianson. 2016. Limits on lexical prediction during reading. *Cognitive Psychology* 88:22–60.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* 19(2):313–330.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, et al. 2013. Universal dependency annotation for multilingual parsing. In *ACL*.

Mattias Nilsson and Joakim Nivre. 2009. Learning where to look: Modeling eye movements in reading. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, pages 93–101.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *LREC*.

Alexander Pollatsek, Barbara J Juhasz, Erik D Reichle, Debra Machacek, and Keith Rayner. 2008. Immediate and delayed effects of word frequency and word length on eye movements in reading: a reversed delayed effect of word length. *Journal of Experimental Psychology: Human Perception and Performance* 34(3):726.

Donald E Powers, Jill C Burstein, Martin Chodorow, Mary E Fowles, and Karen Kukich. 2002. Stumping e-rater: challenging the validity of automated essay scoring. *Computers in Human Behavior* 18(2):103–134.

Chaitanya Ramineni, Catherine S Trapani, David M Williamson, Tim Davey, and Brent Bridgeman. 2012. Evaluation of the e-rater® scoring engine for the TOEFL® independent and integrated prompts. *ETS Research Report Series* 2012(1).

Keith Rayner. 1998. Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin* 124(3):372.

Keith Rayner, Jane Ashby, Alexander Pollatsek, and Erik D Reichle. 2004. The effects of frequency and predictability on eye fixations in reading: implications for the ez reader model. *Journal of Experimental Psychology: Human Perception and Performance* 30(4):720.

Keith Rayner and Lyn Frazier. 1989. Selection mechanisms in reading lexically ambiguous words. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 15(5):779.

Keith Rayner, Alexander Pollatsek, Jane Ashby, and Charles Clifton Jr. 2012. *Psychology of reading*. Psychology Press.

Keith Rayner, Timothy J Slattery, Denis Drieghe, and Simon P Liversedge. 2011. Eye movements and word skipping during reading: effects of word length and predictability. *Journal of Experimental Psychology: Human Perception and Performance* 37(2):514.

Keith Rayner and Arnold D Well. 1996. Effects of contextual constraint on eye movements in reading: A further examination. *Psychonomic Bulletin & Review* 3(4):504–509.

Erik D Reichle, Alexander Pollatsek, Donald L Fisher, and Keith Rayner. 1998. Toward a model of eye movement control in reading. *Psychological Review* 105(1):125.

Erik D Reichle, Andrew E Reineberg, and Jonathan W Schooler. 2010. Eye movements during mindless reading. *Psychological Science* 21(9):1300–1310.

Erik D Reichle, Tessa Warren, and Kerry McConnell. 2009. Using ez reader to model the effects of higher level language processing on eye movements during reading. *Psychonomic Bulletin & Review* 16(1):1–21.

Luz Rello and Miguel Ballesteros. 2015. Detecting readers with dyslexia using machine learning with eye tracking measures. In *Proceedings of the 12th Web for All Conference*. ACM, page 16.

Leah Roberts and Anna Siyanova-Chanturia. 2013. Using eye-tracking to investigate topics in L2 acquisition and L2 processing. *Studies in Second Language Acquisition* 35(02):213–235.

Beatrice Santorini. 1990. Part-of-speech tagging guidelines for the Penn Treebank project (3rd revision). *Technical Reports (CIS)* .

Mark D Shermis and Jill Burstein. 2013. *Handbook of automated essay evaluation: Current applications and new directions*. Routledge.

Nathaniel J Smith and Roger Levy. 2013. The effect of word predictability on reading time is logarithmic. *Cognition* 128(3):302–319.

Andreas Stolcke et al. 2002. SRILM - an extensible language modeling toolkit. In *Interspeech*. volume 2002, page 2002.

David M Williamson. 2009. A framework for implementing automated scoring. In *Annual Meeting of the American Educational Research Association and the National Council on Measurement in Education, San Diego, CA*.

Xiaoming Xi, Derrick Higgins, Klaus Zechner, and David M Williamson. 2008. Automated scoring of spontaneous speech using SpeechRaterSM v1. 0. *ETS Research Report Series* 2008(2).

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading ESOL texts. In *ACL*. pages 180–189.

Comparing Theories of Speaker Choice Using a Model of Classifier Production in Mandarin Chinese

Meilin Zhan
Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, MA 02139
meilinz@mit.edu

Roger Levy
Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, MA 02139
rplevy@mit.edu

Abstract

Speakers often have more than one way to express the same meaning. What general principles govern speaker choice in the face of optionality when near semantically invariant alternation exists? Studies have shown that optional reduction in language is sensitive to contextual predictability, such that the more predictable a linguistic unit is, the more likely it is to get reduced. Yet it is unclear to what extent these cases of speaker choice are driven by audience design versus toward facilitating production.

Here we argue that for a different optionality phenomenon, namely classifier choice in Mandarin Chinese, Uniform Information Density and at least one plausible variant of availability-based production make opposite predictions regarding the relationship between the predictability of the upcoming material and speaker choices. In a corpus analysis of Mandarin Chinese, we show that the distribution of speaker choices supports the availability-based production account, and not Uniform Information Density.

1 Introduction

The expressivity of natural language often gives speakers multiple ways to convey the same meaning. Meanwhile, linguistic communication takes place in the face of environmental and cognitive constraints. For instance, language users have limited memory and cognitive resources, the environment is noisy, and so forth. What general principles govern speaker choice in the face of alternations that are (nearly) semantically invariant? To the extent that we are able to provide a general answer to this question it will advance our fundamental knowledge of human language production.

Studies have shown that alternations are very often sensitive to contextual predictability. For well-studied cases of optional REDUCTION in language, the following trend is widespread: the more predictable a linguistic unit is, the more likely it is to get reduced. Predictable words are phonetically reduced (Jurafsky et al., 2001; Bell et al., 2009; Seyfarth, 2014) and have shorter lexical forms (Piantadosi et al., 2011), and optional function words are more likely to be omitted when the phrase they introduce is predictable (Jaeger, 2010). Yet it is unclear to what extent speakers' choices when faced with an alternation are made due to audience design or to facilitate production. For example, the above pattern of predictability sensitivity in optional reduction phenomena is predicted by both the Uniform Information Density (UID) hypothesis (Levy and Jaeger, 2007), a theory which that the speaker aims to convey information at a relatively constant rate and which can be motivated via considerations of optimality from the comprehender's perspective (e.g., Smith and Levy, 2013), and by the speaker-centric availability-based production hypothesis (Bock, 1987; Ferreira and Dell, 2000), which hypothesizes that the dominant factor in determining speaker choice is that the speaker uses whatever material is readily available when it comes time to convey a particular part of a planned message.

Here we argue that for a different optionality phenomenon, namely classifier choice in Mandarin Chinese, UID and availability-based production make opposite predictions regarding the relationship between the predictability of upcoming material and speaker choice. In a corpus analysis of Mandarin Chinese, we show that the distribution of speaker choices supports the availability-based production account, and not UID.

Proceedings of NAACL-HLT 2018, pages 1997–2005
New Orleans, Louisiana, June 1 - 6, 2018.

2 Uniform Information Density and Availability-based Production

In Sections 2 and 3, we explain why the UID and availability-based production accounts make the same predictions in many cases, but can be potentially disentangled using Chinese classifier choice. Here we exemplify predictions of these two accounts in the case of optional function word omission.

For optional function word omission such as *that*-omission ((1) and (2)), predictability effects have been argued to be consistent with both the speaker-oriented account of AVAILABILITY-BASED PRODUCTION (Bock, 1987; Ferreira and Dell, 2000) and the potentially audience-oriented account of UNIFORM INFORMATION DENSITY (Levy and Jaeger, 2007). On both accounts, but for different reasons, the less predictable the clause introduced by the functional word, the more likely the speaker will be to produce the function word *that*.

(1) The student (that) you tutored graduated.

(2) The woman thought (that) we were crazy.

The UID hypothesis claims that within boundaries defined by grammar, when multiple options are available to encode a message, speakers prefer the variant that distributes information density most uniformly, thus lowering the chance of information loss or miscommunication (Levy and Jaeger, 2007; Jaeger, 2010). In (1), if the function word *that* is omitted, the first word of the relative clause *you* serves two purposes: signaling the onset of the relative clause, and conveying part of the contents of the relative clause itself. These both contribute to the information content of the first relative clause-internal word. If one or both is high-surprisal, then the first relative clause-internal word might be a peak in information density, as illustrated in Figure 1 (top left). If instead the function word *that* is produced, *that* signals the onset of the relative clause, and *you* only communicates part of the content of the relative clause itself. This could help eliminate any sharp peak in information density, as illustrated in Figure 1 (bottom left). Thus, if the speaker's goal is to transfer information as smoothly as possible, the less predictable the upcoming clause, the more inclined the speaker would be to produce the function word *that*.

On the availability-based production hypothesis, speaker choice is governed by the relationship by the relative time-courses of (i) when a part of a message needs to be expressed within an utterance, and (ii) when the linguistic material to encode that part of the message becomes available for production. If material that specifically encodes a part of the message is available when it comes time to convey that part of the message, it will be used—that is the PRINCIPLE OF IMMEDIATE MENTION of Ferreira and Dell (2000). If, on the other hand, that material is not yet available, then other available material consistent with the grammatical context produced thus far and that does not cut off the speaker's future path to conveying the desired content will be used. In (1), assuming the function word *that* is always available when the speaker plans to produce a relative clause, the speaker will produce *that* when the upcoming relative clause or the first part of its contents are not yet available. If phrase structures and phrase contents take longer to become available when they are lower-predictability—an assumption consistent with the literatures on picture naming (Oldfield and Wingfield, 1965) and word naming (Balota and Chumbley, 1985)—then the less predictable the relative clause, the lower the probability that its first word, w_1, will be available when the time comes to begin the relative clause, as illustrated in Figure 2 (left). Under these circumstances, the speaker would choose to produce other available material, namely function word *that*. If, in contrast, the upcoming relative clause is predictable, then w_1 will be more likely to be available, and the speaker would be more likely to omit the function word *that* and immediately proceed with w_1.

While these two accounts differ at many levels, they make the same prediction for function word omission in syntactic reduction such as (1) and (2). It is difficult to disentangle these accounts empirically.[1] Below we will show that for a different optionality phenomenon, namely classifier choice in Mandarin, these accounts may make different predictions.

[1] Prior work (Jaeger, 2010) acknowledged this entanglement of the predictions of these accounts, and attempted to tease the accounts apart via joint modeling using logistic regression. The present study builds on these efforts by exploring a case involving a starker disentanglement of the accounts' predictions.

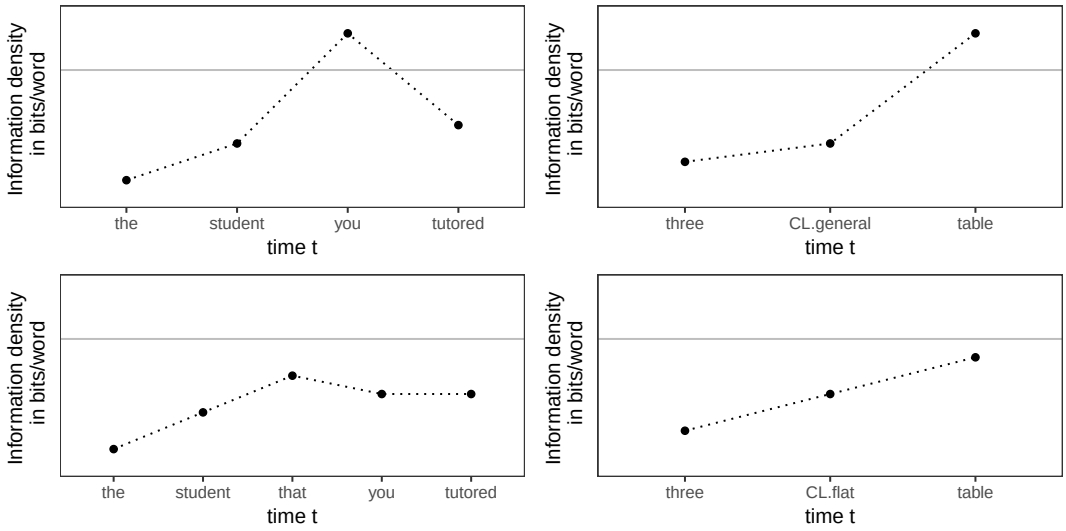


Figure 1: Schematic illustrations of Uniform Information Density in the context of relative clause (left) and classifier choice (right). The grey lines indicate a hypothetical channel capacity.

3 Classifiers in Mandarin Chinese

Languages in the world can be broadly grouped into classifier languages and non-classifier languages. In non-classifier languages, such as English and other Indo-European languages, a numeral modifies a noun directly: e.g., *three tables, two projects*. In Mandarin Chinese and other classifier languages, a numeral classifier is obligatory when a noun is to be preceded with a numeral (and often obligatory with demonstratives): e.g., *san zhang zhuozi* "three CL.flat table", *liang xiang gongcheng* "two CL.item project". Although it has been hypothesized that numeral classifiers play a functional role analogous to that of the singular–plural distinction in other languages (Greenberg, 1972), it is not clear whether there is a meaningful correlation between the presence of numeral classifiers and plurality among the languages of the world (Dryer and Haspelmath, 2013).

In Mandarin, classifiers, together with their associated numeral or demonstrative, precede the head noun of a noun phrase. There are about 100 individual numeral classifiers (Ma, 2015). While different nouns are compatible with different SPECIFIC classifiers, there is a GENERAL classifier *ge*(个) that can be used with most nouns. In some cases, the alternating options between using a general or a specific classifier with the same noun are almost semantically invariant. Table 1 shows examples of classifier options in fragments of naturally occuring texts.

Yet these options have different effects on the information densities of the following nouns. A specific classifier is more likely to reduce the information density of the upcoming noun than a general classifier because a specific classifier constrains the space of possible upcoming nouns more tightly (Klein et al., 2012). Consider the following pair of classifier examples (3) and (4).

(3) 我 买了 三 张 桌子
wo mai-le san zhang zhuozi
I bought three CL.flat table ("I bought three tables")

(4) 我 买了 三 个 桌子
wo mai-le san ge zhuozi
I bought three CL.general table ("I bought three tables")

As shown in Figure 1 (top right), while a general classifier has some information (e.g., signaling there will be a noun), it has relatively low information density—it is the most frequent and generally the highest-probability classifier in many contexts. In comparison, as illustrated in Figure 1 (bottom right), a specific classifier has higher information density—specific classifiers are less frequent than the general classifier and typically lower-predictability—but, crucially, it constrains the hypothesis space for the identity of the upcoming noun, since the noun's referent must meet certain semantic requirement that the classifier is associated with. The UID hypothesis predicts that speakers choose a **specific** classifier more often when the predictability of the noun would other-

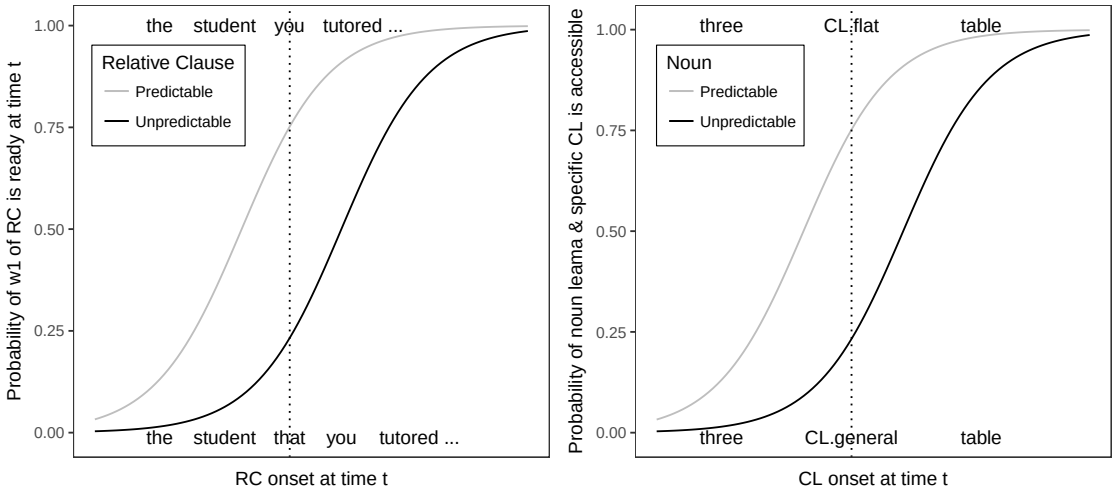


Figure 2: Schematic illustrations of availability-based production in the context of relative clause (left) and classifier choice (right). X axis presents the progression of time. The dotted lines indicate onset times for relative clause and classifier respectively.

wise be low.

Availability-based production, provided three plausible assumptions, makes different predictions than UID. The first assumption is that a speaker must access a noun lemma in order to access its appropriate specific classifier. The second assumption is that unpredictable noun lemmas are harder and/or slower to access (as described in Section 2, this assumption is supported by findings from the naming literature). The third assumption is that the general classifier is *always* available, regardless of the identity of the upcoming noun, as it is compatible with virtually every noun. Under these assumptions, for unpredictable nouns, specific classifiers will less often be available to the speaker when the time comes to initiate production of classifier, as shown in Figure 2 (right). Since noun lemmas need to be accessed before their associated specific classifiers, the less predictable the noun, the less likely the noun lemma and hence the associated specific classifier is to be available by the classifier onset time t. The general classifier, in contrast, is always accessible. Under these assumptions, the availability-based production hypothesis thus predicts that speakers choose a **general** classifier more often when the following noun is less predictable.

4 Data and Processing

To provide data for this study, we created a corpus of naturally occurring classifier-noun pairs from SogouCS, a collection of online news texts from various channels of Sohu News (Sogou, 2008). The deduplicated version of the corpus (see Section 4.1 for deduplication details) has 11,548,866 sentences. To parse and annotate the data, we built a pipeline to 1) clean and deduplicate the data, 2) part-of-speech tag and syntactically parse the clean text, and 3) extract and filter classifier-noun pairs from the parsed text. We are aware that a spoken corpus would be ideal to investigate speaker choice, but nothing this big is available. Instead we used SogouCS to approximate the language use of native speakers.

4.1 Cleaning and deduplication

Since the data contain web pages, many snippets are not meaningful content but automatically generated text such as legal notices. To use this corpus as a reasonable approximation of language experience of speakers, we performed deduplication on the data, following similar practice adopted by other work dealing with web-based corpora (Buck et al., 2014). After cleaning the text, we removed repeated lines in the corpus.

4.2 Word segmentation, POS-tagging and syntactic parsing

We used the Stanford CoreNLP toolkit for word segmentation, part-of-speech tagging, and syntactic parsing (Manning et al., 2014). We used CoreNLP's Shift-Reduce model for parsing (Zhu et al., 2013). We also got dependency parsing results as part of the Stanford CoreNLP output.

Noun	个 (ge, CL.general)	项 (xiang, CL.item)	张 (zhang, CL.flat)
公告 announcement	一 口 气 发布 11 个 公告 a CL breath release 11 CL announcement “release 11 announcements at one go”	连续 发布 三 项 公告 consecutively release three CL announcement “release three announcements in a” row”	门口 贴了 一 张 公告 door paste a CL announcement “there is an announcement on the door”
账单 bill	女儿 拿着 一 个 账单 就 过来了 daughter carry a CL bill at once come “daughter came with a bill at once”	not co-occurring	在 一 张 账单 上 解决 所有 收费 问题 on a CL bill solve all charge problem “solve all charge problems on a bill”
工程 project	跟 圆明园 有关 的 一 个 工程 to Yuanmingyuan related de a CL project “a project related to Yuanmingyuan”	抓好 六 项 重点 工程 grasp six CL key project “manage six key projects”	not co-occurring
活动 activity	昨天 我 参加了 一 个 活动 yesterday I attend a CL activity “yesterday I attended an activity”	广州市 今天 开展 的 一 项 活动 Guangzhou today hold de a CL activity “an activity held by Guangzhou today”	not co-occurring

Table 1: Examples from development set of available classifier options that are semantically (near-)invariant

4.3 Extracting and filtering classifier-noun pairs

From the parsed corpus, we extracted all observations where the head noun has a `nummod` relation with a numeral and the numeral has a `mark:clf` relation with a classifier. Figure 3 illustrates two such examples. We included classifiers in the list of 105 individual classifiers identified by Ma (2015) that are identified by the Stanford CoreNLP toolkit. For the purpose of restricting our data to cases of (nearly) semantically invariant alternation, we excluded classifiers such as *zhong* (“CL.kind”) that would introduce a clear truth-conditional change in utterance meaning, compared with the general classifier *ge*. We did further filtering to get nouns that can be used with both the general classifier and at least one specific classifier. This left us 1,479,579 observations of classifier-noun pairs.

To construct the development set, we randomly sampled about 10% of the noun types (1,179) and extracted all observations with of these noun types. We manually checked and filtered applicable classifiers for these noun types and we ended up with 713 noun types for the development set. For the test set, we also randomly sampled about 10% of the noun types (1,093) and extracted all observations with these noun types. We did not perform manual filtering of the test set. We reserve the remaining 80% for future work.

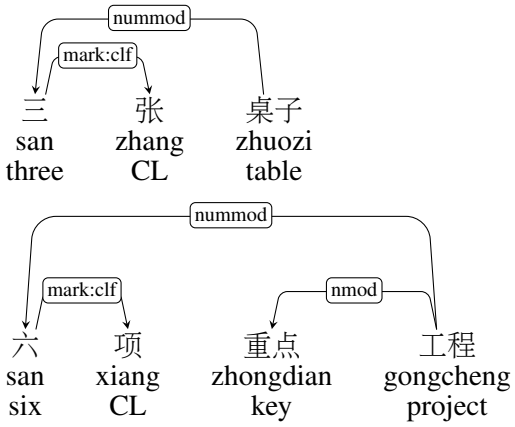


Figure 3: Classifier examples where the head noun has a nummod relation with a numeral and the numeral has a `mark:clf` relation with the classifier

5 Model estimation

We use SURPRISAL, the negative log probability of the word in the context (Hale, 2001; Levy, 2008; Demberg and Keller, 2008; Frank and Bod, 2011; Smith and Levy, 2013), generated from a language model to estimate noun predictability. Since classifiers occur before their corresponding nouns, to avoid circularity, we mapped all target classifiers to the same token, `CL`, in the segmented text for language modeling, analogous to the procedure used in (Levy and Jaeger, 2007) and similar studies. We implemented 5gram modified Kneser-Ney smoothed models with the SRI Lan-

guage Modeling toolkit (Stolcke, 2002) and performed ten-fold cross-validation to estimate noun surprisal.

We used a mixed-effect logit model to investigate the relationship between noun predictability and classifier choice. The dependent variable was the binary outcome of whether a general or a specific classifier was used. For each noun type, we also identified its most frequently used specific classifier. We included two predictors in the analysis: noun surprisal and noun log frequency.[2] We included noun frequency as a control factor for two reasons. First, noun frequency has shown effects on many aspects of speaker behavior. Second, surprisal and frequency of a word are intrinsically correlated. Taken together, these two reasons make noun frequency an important potential confound to be controlled for in investigating any potential effect of noun surprisal on classifier choice.

We included noun and potential specific classifier as random factors, both with random intercepts and random slopes for noun surprisal. This random effect structure is maximal with regard to testing effects of noun surprisal, which varies within noun and within classifier (Barr et al., 2013). We then applied the model to the test set. The full formula in the style of `R`'s `lme4` package (Bates et al., 2014) is:

```
cl_choice~noun_surprisal+log_noun_freq
+(1+noun_surprisal|noun)
+(1+noun_surprisal|potential_spec_cl)
```

We used Markov chain Monte Carlo (MCMC) methods in the `R` package `MCMCglmm` (Hadfield et al., 2010) for significance testing, an based our p-values on the posterior distribution of regression model parameters using an uninformative prior and determining the largest possible symmetric posterior confidence interval on one side of zero, as is common for MCMC-based mixed model fitting (Baayen et al., 2008).

6 Results

In both the development set and the test set, overall we saw more observations with a specific classifier than with a general classifier (55.4% vs. 44.6% in the development set, 63.1% vs. 36.9% in the test set). For the development set, we find that the less predictable the noun, the less likely a specific classifier is to be used ($\beta = -0.038,\ p < 0.001$, Figure 4). There was no effect of noun frequency ($\beta = 0.018,\ p = 0.51$, Figure 5). For the test set, the result of noun predictability replicates ($\beta = -0.059,\ p < 0.001$, Figure 6).[3] In the test set but not in the development set, we also found an effect of noun frequency ($\beta = -0.11,\ p < 0.001$, Figure 7): the more frequent the noun, the less likely a specific classifier is to be used. Further analysis suggests that this effect of noun frequency in the test set is likely to be an artifact of incorrect noun–classifier associations that would disappear were we to filter the test set in the same way as we filtered the development set.[4] The consistent effect of noun surprisal on classifier choice in both our development and test sets supports the availability-based production hypothesis, and is inconsistent with the predictions of UID.

One potential concern regarding the above conclusion that noun predictability drives classifier choice is that it might not fully take into account effects of the frequencies of classifiers themselves on availability. The availability-based production hypothesis does not exclude the possibility that a classifier's accessibility is substantially dependent on its frequency, and the general classifier is indeed the most frequently used classifier. However, if specific classifier frequency were confounding the apparent effect of noun surprisal that we see in our analysis, there would have to be a correlation in our dataset between specific classifier frequency and noun surprisal. Our inclusion of a by-specific-classifier random intercept largely rules out the possibility that even a correlation that the above-mentioned one could be driving our effect. To be thorough, we tried a version of our regression analysis that also include a fixed effect for the log frequency of potential specific classifier as a control. We did not find any qualitative change to

[2] We used base 2 here to be consistent with the base used in noun surprisal.

[3] As can be seen in Figure 6, there is a bump at bin 27 in the rate of using a specific classifier. We consider this likely to be due to data sparsity: the number of observations is small in the last two bins of noun surprisal ($n = 27$ and $n = 3$), and there is no such bump in the development set.

[4] We found a marginal effect of noun frequency in our unfiltered development set, where the more frequent the noun was, the less likely it was used with a specific classifier. We did further analysis with the dev set and found that the"nouns" (some of them were misclassified as nouns from the results of the automatic parsing) that were excluded tend to have a higher frequency compared to the ones that were included, and the excluded ones also had a lower rate of concurring with a specific classifier. This tendency suggests that in the unfiltered test set, illegible nouns may contribute at least partially to the noun frequency effect.

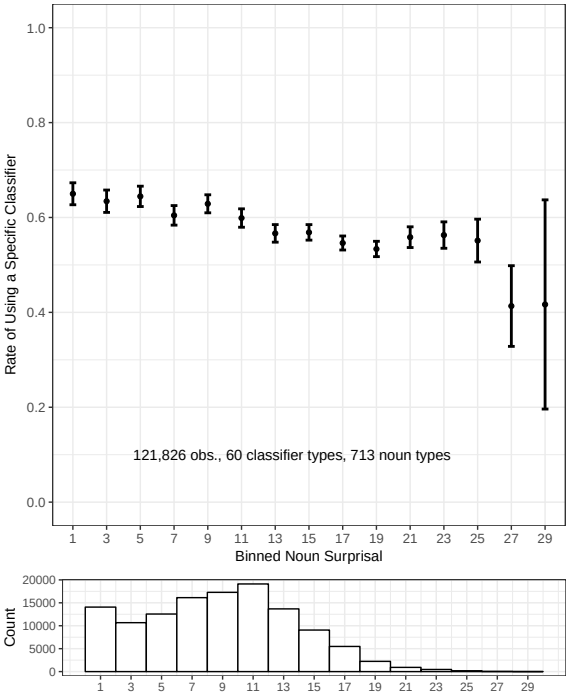


Figure 4: Dev set: N-gram estimated noun surprisal and the rate of using a specific classifier (as opposed to the general classifier *ge*).

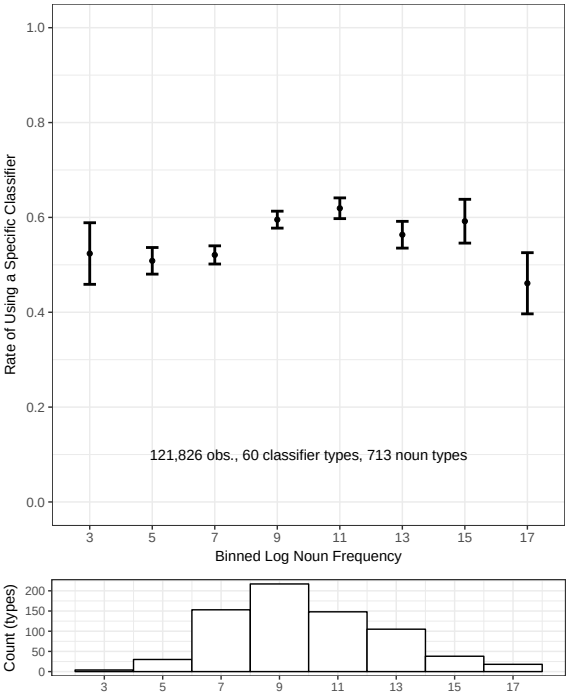


Figure 5: Dev set: Noun frequency (log scale) and the rate of using a specific classifier (as opposed to the general classifier *ge*).

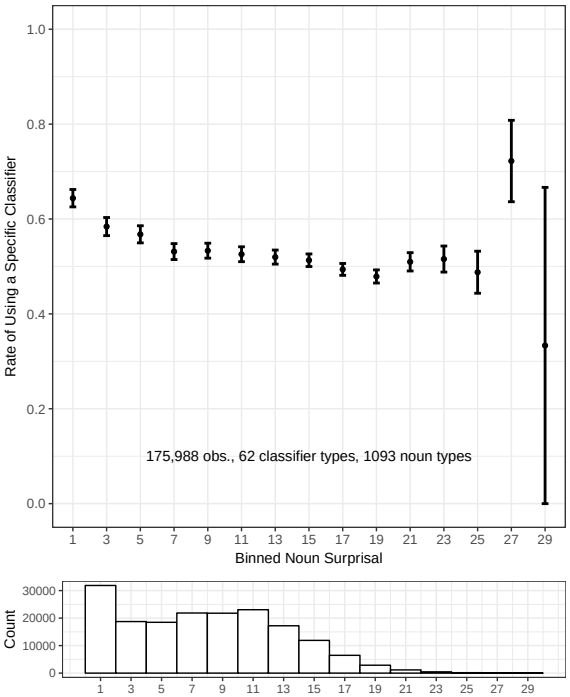


Figure 6: Test set: N-gram estimated noun surprisal and the rate of using a specific classifier.

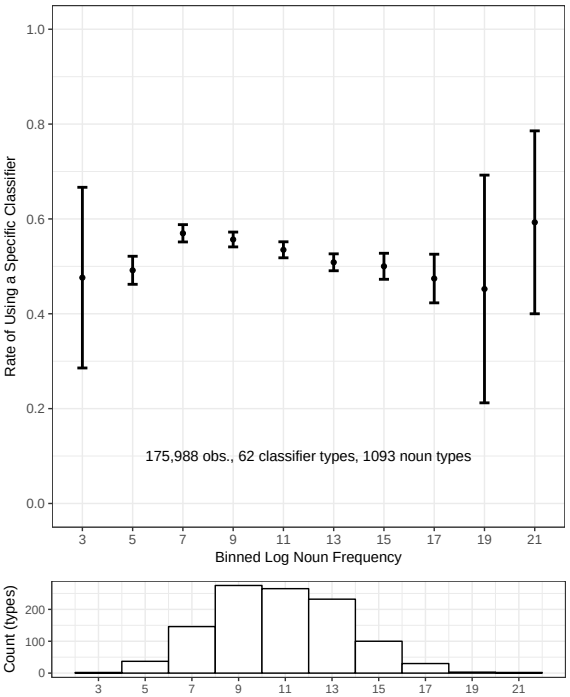


Figure 7: Test set: Noun frequency (log scale) and the rate of using a specific classifier.

the results: the effect of noun surprisal on specific classifier choice remains the same. We also note that in this new analysis, we do not find a significant effect of specific classifier log frequency on classifier choice ($p = 0.629$ for the dev set and $p = 0.7$ for the test set). This additional analysis suggests that it is unlikely that the effect of specific classifier frequency to be driving the effect of noun surprisal.

Overall, we did not find evidence for the UID hypothesis at the level of alternating options with different information density, in our case, a specific classifier versus a general classifier. We demonstrate that within the scope of near semantically invariant alternation, classifier choice is modulated by noun predictability with the tendency to facilitate speaker production. Our results lend support to an availability-based production model. We did not find consistent evidence for the effect of noun frequency on classifier choice. The effect of noun frequency remains unclear and we will need to test it with a larger sample of noun types.

7 Conclusion

Though it has proven difficult to disentangle UID and availability-based production through optional word omission phenomena, we have demonstrated here that the two accounts can potentially be distinguished through at least one word alternation phenomenon. The UID hypothesis predicts that predictable nouns favor the *general* classifier whereas availability-based production predicts that predictable nouns favor a *specific* classifier. Our empirical results favor the availability-based production account.

To the best of our knowledge, this is the first study that demonstrates contextual predictability is correlated with classifier choice. This study provides a starting point to understand the cognitive mechanisms governing speaker choices as manifested in various language optionalities. Ultimately we plan to complement our corpus analysis with real-time language production experiments to more throughly test hypotheses about speaker choice.

Acknowledgments

We gratefully acknowledge valuable feedback from Naomi Feldman, members of MIT's Computational Psycholinguistics Laboratory, three anonymous reviewers, technical advice for data processing from Wenzhe Qiu, and support from NSF grants BCS-1456081 and BCS-1551866 to RPL, and an MIT Henry E. Singleton (1940) Fellowship to MZ.

References

R Harald Baayen, Douglas J Davidson, and Douglas M Bates. 2008. Mixed-effects modeling with crossed random effects for subjects and items. *Journal of Memory and Language* 59(4):390–412.

David A Balota and James I Chumbley. 1985. The locus of word-frequency effects in the pronunciation task: Lexical access and/or production? *Journal of Memory and Language* 24(1):89–106.

Dale J Barr, Roger Levy, Christoph Scheepers, and Harry J Tily. 2013. Random effects structure for confirmatory hypothesis testing: Keep it maximal. *Journal of Memory and Language* 68(3):255–278.

Douglas Bates, Martin Maechler, Ben Bolker, Steven Walker, et al. 2014. lme4: Linear mixed-effects models using eigen and s4. *R package version* 1(7).

Alan Bell, Jason M Brenier, Michelle Gregory, Cynthia Girand, and Dan Jurafsky. 2009. Predictability effects on durations of content and function words in conversational English. *Journal of Memory and Language* 60(1):92–111.

Kathryn Bock. 1987. An effect of the accessibility of word forms on sentence structures. *Journal of Memory and Language* 26(2):119–137.

Christian Buck, Kenneth Heafield, and Bas Van Ooyen. 2014. N-gram counts and language models from the common crawl. In *LREC*. volume 2, page 4.

Vera Demberg and Frank Keller. 2008. Data from eye-tracking corpora as evidence for theories of syntactic processing complexity. *Cognition* 109(2):193–210.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig. http://wals.info/.

Victor S Ferreira and Gary S Dell. 2000. Effect of ambiguity and lexical availability on syntactic and lexical production. *Cognitive Psychology* 40(4):296–340.

Stefan L Frank and Rens Bod. 2011. Insensitivity of the human sentence-processing system to hierarchical structure. *Psychological Science* 22(6):829–834.

Joseph H Greenberg. 1972. Numeral classifiers and substantival number: Problems in the genesis of a linguistic type. *Working Papers on Language Universals* 9.

Jarrod D Hadfield et al. 2010. MCMC methods for multi-response generalized linear mixed models: the MCMCglmm R package. *Journal of Statistical Software* 33(2):1–22.

John Hale. 2001. A probabilistic Earley parser as a psycholinguistic model. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*. Association for Computational Linguistics, pages 1–8. http://aclweb.org/anthology/N/N01/N01-1021.pdf.

T. Florian Jaeger. 2010. Redundancy and reduction: Speakers manage syntactic information density. *Cognitive Psychology* 61(1):23–62.

Daniel Jurafsky, Alan Bell, Michelle Gregory, and William D Raymond. 2001. Probabilistic relations between words: Evidence from reduction in lexical production. *Typological studies in language* 45:229–254.

Natalie M Klein, Greg N Carlson, Renjie Li, T Florian Jaeger, and Michael K Tanenhaus. 2012. Classifying and massifying incrementally in chinese language comprehension. *Count and mass across languages* pages 261–282.

Roger Levy. 2008. Expectation-based syntactic comprehension. *Cognition* 106(3):1126–1177.

Roger P. Levy and T. Florian Jaeger. 2007. Speakers optimize information density through syntactic reduction. In *Advances in Neural Information Processing Systems*. pages 849–856.

Aimin Ma. 2015. *Hanyu geti liangci de chansheng yu fazhan[The Emergence and Development of Chinese Individual Classifiers]*. China Social Sciences Press.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL (System Demonstrations)*. pages 55–60. http://www.aclweb.org/anthology/P14-5010.

R.C. Oldfield and A. Wingfield. 1965. Response latencies in naming objects. *Quarterly Journal of Experimental Psychology* 17(4):273–281.

Steven T Piantadosi, Harry Tily, and Edward Gibson. 2011. Word lengths are optimized for efficient communication. *Proceedings of the National Academy of Sciences* 108(9):3526–3529.

Scott Seyfarth. 2014. Word informativity influences acoustic duration: Effects of contextual predictability on lexical representation. *Cognition* 133(1):140–155.

Nathaniel J Smith and Roger Levy. 2013. The effect of word predictability on reading time is logarithmic. *Cognition* 128(3):302–319.

Sogou. 2008. Sogou lab data: Sohu news corpus 2008 version. http://www.sogou.com/labs/resource/cs.php. Accessed: 2017-05-30.

Andreas Stolcke. 2002. SRILM-an extensible language modeling toolkit. In *Interspeech*. volume 2002, page 2002.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *ACL (1)*. pages 434–443. http://www.aclweb.org/anthology/P13-1043.

Spotting Spurious Data with Neural Networks

Hadi Amiri, Timothy A. Miller, Guergana Savova
Boston Children's Hospital Informatics Program, Harvard Medical School
{firstname.lastname}@childrens.harvard.edu

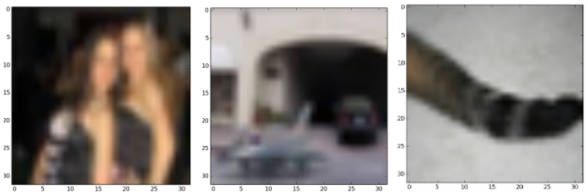
Abstract

Automatic identification of spurious instances (those with potentially wrong labels in datasets) can improve the quality of existing language resources, especially when annotations are obtained through crowdsourcing or automatically generated based on coded rankings. In this paper, we present an effective approach inspired by queueing theory and psychology of learning to automatically identify spurious instances in datasets. Our approach discriminates instances based on their "difficulty to learn," determined by a downstream learner. Our method can be applied to *any* dataset assuming the existence of a neural network model for the target task of the dataset. Our best approach outperforms competing state-of-the-art baselines and has a MAP of 0.85 and 0.22 in identifying spurious instances in synthetic and carefully-crowdsourced real-world datasets respectively.

1 Introduction

The importance of error-free language resources cannot be overstated as errors can inversely affect interpretations of the data, models developed from the data, and decisions made based on the data. Although the quality of language resources can be improved through good annotation guidelines, test questions, etc., annotation noise still exists (Gupta et al., 2012; Lasecki et al., 2013). For example, Figure 1 shows sample spurious instances (those with potentially wrong labels) in CIFAR-10 (Krizhevsky, 2009) which is a benchmark dataset for object classification. Spurious instances can mislead systems, and, if available in test data, lead to unrealistic comparison among competing systems.

Previous works either directly identify noise in datasets (Hovy et al., 2013; Dickinson and Meurers, 2003; Eskin, 2000; Loftsson, 2009), or develop models that are more robust against noise (Guan et al., 2017; Natarajan et al., 2013; Zhu et al., 2003; Zhu and Wu, 2004). Furthermore, recent works on adversarial perturbation have tackled this problem (Goodfellow et al., 2015; Feinman et al., 2017). However, most previous approaches require either annotations generated by each individual annotator (Guan et al., 2017), or both task-specific and instance-type (genuine or adversarial) labels for training (Hendrik Metzen et al., 2017; Zheng et al., 2016), or noise-free data (Xiao et al., 2015). Such information is often not available in the final release of most datasets.



(a) Truck (b) Airplane (c) Cat

Figure 1: Spurious instances in CIFAR-10. (a) not a truck, (b) missing annotation for car as another object category, (c) incomplete image of a cat.

Current approaches utilize prediction probability/loss of instances to tackle the above challenges in identifying spurious instances. This is because prediction probability/loss of spurious instances tend to be lower than that of genuine instances (He and Garcia, 2009). In particular, the Bayesian Uncertainty model (Feinman et al., 2017) defines spurious instances as those that have greater uncertainty (variance) in their stochastic predictions, and the Variational Inference model (Rehbein and Ruppenhofer, 2017; Hovy et al., 2013) expects greater posterior entropy in predictions made for spurious instances.

In this paper, our hypothesis is that spurious instances are *frequently* found to be difficult to

Proceedings of NAACL-HLT 2018, pages 2006–2016
New Orleans, Louisiana, June 1 - 6, 2018.

learn during training process. This difficulty in learning stems from the intrinsic discrepancy between spurious and the cohort of genuine instances which frequently makes a learner less confident in predicting the *wrong* labels of spurious instances. Based on this hypothesis, we present two frameworks which are inspired by findings in queueing theory and psychology, namely Leitner queue network (Leitner, 1974) and Curriculum Learning (Bengio et al., 2009). Our frameworks can be considered as schedulers that schedule instances to train a downstream learner (e.g. a neural network) with respect to "easiness"/"difficulty" of instances - determined by the extent to which the learner can correctly label (e.g. classify) instances during the training process. The two frameworks, however, differ in their views on the theory of learning as we describe below:

Curriculum learning is inspired by the learning principle that humans can learn more effectively when training starts with *easier* concepts and gradually proceeds with more difficult ones (Bengio et al., 2009). On the other hand, Leitner system is inspired by *spaced repetition* (Dempster, 1989; Cepeda et al., 2006), the learning principle that effective and efficient learning can be achieved by working more on difficult concepts and less on easier ones. Both frameworks are effective, conceptually simple, and easy to implement.

The contributions of this paper are as follows: (a) we develop a cognitively-motivated and effective algorithm for identifying spurious instances in datasets, (b) our approach can be applied to *any* dataset without modification if there exists a neural network architecture for the target task of the dataset, and (c) we release a tool that can be easily used to generate a ranked list of spurious instances in datasets.[1] Our tool requires a dataset and its corresponding network architecture to generate a ranked list of spurious instances in the dataset.

Our best approach (Leitner model) has a mean average precision (MAP) of 0.85 and 0.22 in identifying spurious instances on real-world and synthetic datasets and outperforms competing state-of-the-art baselines.

2 Method

We assume that our learner is a neural network which trains for k iterations until convergence. Furthermore, we assume that spurious and genuine instances are mixed at training time and the network is only provided with task-specific but not genuine/spurious labels for the instances.

[1] https://scholar.harvard.edu/hadi/spot

Algorithm 1. Curriculum Spotter

Input: $\mathbf{H}$: training data, $\mathbf{V}$: validation data, k : number of iterations
Output: Ranked list of spurious instances

$batch = \mathbf{H}$
$S^0[h_j] = 0$ for $h_j \in \mathbf{H}$
For $epoch = 1$ to k:
  $model = \mathbf{train}(batch, \mathbf{V})$
  $loss = \mathbf{Loss}(model, \mathbf{H})$
  $\lambda = \mathbf{compute\_lambda}(model, loss, \mathbf{H})$
  $easy\_batch = \mathbf{sample\_easy}(\lambda, loss, \mathbf{H})$
  $hard\_batch = \mathbf{H} - easy\_batch$
  $batch = easy\_batch + \mathbf{top}(\frac{epoch}{k}, hard\_batch)$
  $S^{epoch} = \mathbf{update\_stat}(S^{epoch-1}, hard\_batch, loss)$
End for
return $\mathbf{sort}(S^k, \mathbf{H}, loss)$

Figure 2: Curriculum Spotter. $\mathbf{Loss}(.)$ computes loss of a network with respect to given instances, $\mathbf{compute\_lambda}(.)$ computes the average loss of current model for correctly classified instances, $\mathbf{sample\_easy}(.)$ creates list of easy instances using current loss values, $\mathbf{top}(.)$ returns $epoch/k$ fraction of easiest hard instances, $\mathbf{update\_stat}(.)$ scores instances according to Eq. (1), and $\mathbf{sort}(.)$ ranks instances based on the resulting scores S updated by Eq. (2).

2.1 Curriculum Learning

Bengio et al. (2009) and Kumar et al. (2010) developed training paradigms which are inspired by the learning principle that humans can learn more effectively when training starts with easier concepts and gradually proceeds with more difficult ones. Since easiness of information is not readily available in most datasets, previous approaches used heuristic techniques (Spitkovsky et al., 2010; Basu and Christensen, 2013) or optimization algorithms (Jiang et al., 2015, 2014) to quantify easiness for instances. These approaches consider an instance as easy if its prediction loss is smaller than a threshold (λ). Given a neural network as the learner, we adopt curriculum learning to identify spurious instances as follows (see Figure 2):

At each iteration i, we divide all instances into *easy* and *hard* batches using the iteration-specific threshold λ_i and the loss values of instances at iteration i, obtained from the current partially-trained network. All instances with a loss smaller than λ_i are considered as easy and the rest are consid-

ered as hard. All easy instances in conjunction with $\delta_i \in [0, 1]$ fraction of easiest hard instances (those with smallest loss values greater than λ_i) are used for training at iteration i. We set each λ_i to the average[2] loss of training instances that are correctly classified by the current partially-trained network. Furthermore, at each iteration $i > 1$, we set $\delta_i = i/k$ where k is the total number of iterations. In this way, difficult instances are gradually introduced to the network at every new iteration.

The $\mathbf{update\_stat}(.)$ function in Figure 2 scores instances based on their frequency of occurrence in the hard batch. In particular, for each instance h_i:

$$S^e(h_i) = S^{e-1}(h_i) + \mathbb{1}_{hard\_batch^e}(h_i) \times \Big(\frac{1}{|hard\_batch^e|} + loss^e(h_i)\Big), \quad (1)$$

where $S^e(h_i)$ is the score of h_i at iteration e, $\mathbb{1}_Y(x)$ is an indicator function which is 1 when $x \in Y$ and otherwise 0, $hard\_batch^e$ indicates the set of hard instances at iteration e, and $loss^e(h_i)$ is the loss of the network for h_i at iteration e. The above function assigns higher scores to instances that are frequently considered as hard instances by the curriculum learning framework (such instances are ranked higher in the final ranked list of spurious instances). It also assigns a final score of $S^k(h_i) = 0$ to instances that are treated as easy instances throughout the training process, i.e. those that have a loss smaller than the iteration-specific threshold λ_i at each iteration i and, therefore, are always placed in the $easy\_batch$. To break the tie for these instances in the final ranking, we resort to their final loss values as follows:

$$S^k(h_i) = loss^k(h_i), \text{if } S^k(h_i) = 0. \quad (2)$$

2.2 Leitner System

The Leitner System is inspired by the broad evidence in psychology that shows human ability to retain information improves with repeated exposure and exponentially decays with delay since last exposure (Cepeda et al., 2006). Spaced repetition forms the building block of many educational devices, such as flashcards, in which small pieces of information are repeatedly presented to a learner on a schedule determined by a spaced repetition algorithm. Such algorithms show that human learners can learn efficiently and effectively by increasing intervals of time between subsequent reviews of previously learned materials (Dempster, 1989; Novikoff et al., 2012). We adopt the Leitner system to identify spurious instances as follows:

Algorithm 2. Leitner Spotter

Input: $\mathbf{H}$: training data, $\mathbf{V}$: validation data, k : number of iterations, n : number of queues
Output: Ranked list of spurious instances

$Q = [q_0, q_1, \ldots, q_{n-1}]$
$q_0 = [\mathbf{H}]$, $q_i = []$ for $i \in [1, n-1]$
$S^0[h_j] = 0$ for $h_j \in \mathbf{H}$
For $epoch = 1$ to k:
  $batch = []$
  For $i = 0$ to $n - 1$:
    If $epoch\%2^i == 0$:
      $batch = batch + q_i$
  End For
  $promos, demos, loss = \mathbf{train}(batch, \mathbf{V})$
  $\mathbf{update\_queue}(Q, promos, demos)$
  $S^{epoch} = \mathbf{update\_stat}(S^{epoch-1}, Q, loss)$
End for
return $\mathbf{sort}(S^k, \mathbf{H}, loss)$

q_0 epochs = $\{1, 2, 3, 4, 5, \ldots\}$
q_1 epochs = $\{2, 4, 6, 8, 10, \ldots\}$
q_2 epochs = $\{4, 8, 12, 16, 20, \ldots\}$
...

Figure 3: Leitner Spotter. The $\mathbf{train}(.)$ function trains the network using instances in the current $batch$, $\mathbf{update\_queue}(.)$ promotes the correctly classified instances–$promos$–to their next queues and demotes the wrongly classified ones–$demos$–to q_0, $\mathbf{update\_stat}(.)$ scores instances according to Eq. (3), and $\mathbf{sort}(.)$ ranks instances based on resulting scores S updated by Eq. (4).

Suppose we have n queues $\{q_0, q_1, \ldots, q_{n-1}\}$. The Leitner system initially places all instances in the first queue, q_0. As Figure 3 shows, the system trains with instances of q_i at every 2^i iterations. At each iteration, only instances in the selected queues will be used for training the network. During training, if an instance from q_i is correctly classified by the network, the instance will be "promoted" to q_{i+1}, otherwise it will be "demoted" to the first queue, q_0. Therefore, as the network trains through time, higher queues will accumulate easier instances which the network is most accurate about, while lower queues carry either hard or potentially spurious instances. This is because of the intrinsic discrepancy between spurious instances and the cohort of genuine instances which makes the network less confident in predicting the wrong labels of spurious instances. Figure 3 (bot-

[2] We also considered maximum and median loss, but average loss led to greater training gain in terms of effectiveness.

tom) provides examples of queues and their corresponding processing epochs.

The **update_stat**(.) function in Figure 3 scores instances based on their occurrence in q_0. In particular, for each instance h_i:

$$S^e(h_i) = S^{e-1}(h_i) + \mathbb{1}_{q_0^e}(h_i) \times \Big(\frac{1}{|q_0^e|} + loss^e(h_i)\Big), \quad (3)$$

where $|q_0^e|$ indicates the number of instance in q_0 at iteration e. The above function assigns higher scores to instances that frequently occur in q_0. It also assigns a final score of $S^k(h_i) = 0$ (at the last iteration) to instances that have never been demoted to q_0. To break the tie for such instances, we use their final loss value as follows:

$$S^k(h_i) = loss^k(h_i), \text{if } S^k(h_i) = 0. \quad (4)$$

3 Experiments

3.1 Evaluation Metrics

We employ a TREC-like evaluation setting to compare models against each other. For this, we create a pool of K most spurious instances identified by different models. If needed, e.g. in case of real-world datasets, we manually label all instances in the pool and come to agreement about their labels. Then, we compare the resulting labels with the original labels in the dataset to determine spurious/genuine instances. We compare models based on the standard TREC evaluation measures, namely mean average precision (MAP), precision after r instances are retrieved (P@r), and, only for synthetic data, precision after all spurious instances are retrieved (Rprec). We use the `trec-eval` toolkit to compute performance of different models.[3]

3.2 Datasets

We develop synthetic and real-world datasets for our experiments. Since, in contrast to real-world datasets, (most[4]) synthetic datasets do not contain any noisy instances, we can conduct large-scale evaluation by injecting spurious instances into such datasets. Table 1 shows detail information about our datasets.

[3] http://trec.nist.gov/trec_eval/

[4] Some synthetic datasets may contain noise, see sample inconsistencies that our model identified in the bAbi dataset (Weston et al., 2016) in Table 3.

Dataset	Train/Val	SpRatio	Input	Output
Synthetic Dataset				
Addition	10K/2K	$\alpha \in (0, 0.5]$	$(x,y \geq 0)$	$x+y$
Real-world Datasets				
Twitter	10K/1K	0.30	brand tweet	pos/neg
Reddit	4K/400	0.23	cancer post	rel/irel

Table 1: Dataset Information. "SpRatio" shows the fraction of spurious instances in the TREC pool; size of the pool for Addition is 10K instances with no limit on the top K retrieved instances; the corresponding value for Twitter and Reddit datasets are 198 and 152 posts respectively for top $K = 50$.

3.2.1 Synthetic Dataset

The Addition dataset, initially developed by Zaremba and Sutskever (2014), is a synthetic dataset in which an input instance is a pair of non-negative integers smaller than 10^l and the corresponding output is the arithmetic sum of the input; we set $l = 4$ in our experiments. Since this dataset contains only genuine instances, we create noisy datasets by injecting $\alpha \times N$ spurious instances into $(1 - \alpha) \times N$ genuine instances, where $N = 10K$ is the total number of training instances and $\alpha \leq 0.5$ indicates the noise level in the dataset. We create spurious instances as follows: given three random numbers $x_i, x_j, x_k \in [0, 10^l)$ such that $x_j \neq x_k$, the wrong sum (output) for the pair (x_i, x_j) is computed as:

$$\max(0, x_i + (-1)^o \times x_k),$$

where o is a random variable that takes values from $O = \{1, 2\}$ with equal probability.

3.2.2 Real-world Datasets

We crowdsource annotations for two real-world datasets, namely Twitter and Reddit posts (see Table 1). For quality control, we carefully develop annotation schemas as well as high quality test questions (see below) to minimize the chances of spurious labels in the resulting annotations.

The Twitter dataset contains tweets about a telecommunication brand. Tweets contain brand name or its products and services. Annotators are instructed to label tweets as positive/negative if they describe positive/negative sentiment about the target brand. We use 500 labeled instances for annotation quality assurance and ignore data generated by annotators who have less than 80% accuracy on these instances. The resulting Fleiss' kappa (Fleiss, 1971) is $\kappa = 0.66$ on our Twitter dataset which indicates substantial agreement.

The Reddit dataset includes posts about colon, breast, or brain cancer. These posts contain phrases like *colon cancer*, *breast cancer*, or *brain cancer*. Annotators are instructed to label a post as "relevant" if it describes a patient's experience (including sign and symptoms, treatments, etc.,) with respect to the cancer. In contrast, "irrelevant" posts are defined as generic texts (such as scientific papers, news, etc.,) that discuss cancer in general without describing a real patient experience. We use 300 labeled instances for annotation quality assurance and ignore annotations generated by users who have less than 80% accuracy on these instances. The resulting Fleiss' kappa is $\kappa = 0.48$ for the Reddit dataset which indicates moderate agreement.

3.3 Settings

For the synthetic Addition dataset, we set the size of the TREC pool to $K = 10,000$ (size of training data) which indicates there is no limitation on the number of spurious instances that a model can retrieve; note that we have a spurious/genuine label for each instance in the Addition dataset and therefore do not need to label the resulting TREC pool manually. Furthermore, we consider the LSTM network developed by Sutskever et al. (2014) as the downstream learner.[5] Without noise in data, this network obtains a high accuracy of 99.7% on the Addition task.

For the real-world datasets, we allow each model to submits its top 50 most spurious instances to the TREC pool (we have five models including our baselines). As mentioned before, we manually label these instances to determine their spurious/genuine labels. This leads to TREC pools of size 198 and 152 posts (with 59 and 35 spurious instances) for the Twitter and Reddit datasets respectively.

We use the MLP network `fastText` (Joulin et al., 2017) as the downstream learner - for more effective prediction, we add a `Dense` layer of size 512 before the last layer of `fastText`. This network obtains accuracy of 74.6% and 70.2% on Twitter and Reddit datasets respectively.

Finally, for the Leitner system, we experiment with different queue lengths, $n = \{3, 5, 7\}$, and set $n = 5$ in the experiments as this value leads to slightly better performance in our experiments.

[5] http://github.com/fchollet/keras/blob/master/examples/addition_rnn.py

3.4 Baselines

We consider the following baselines; each baseline takes a dataset and a model as input and generates a ranked list of spurious instances in the dataset:

• **Prediction Probability (PP)**: Since prediction loss of spurious instances tend to be higher than that of genuine ones (He and Garcia, 2009; Hendrycks and Gimpel, 2016), this baseline ranks instances in descending order of their prediction loss after networks are trained through standard (rote) training.

• **Variational inference (VI)** (Hovy et al., 2013; Rehbein and Ruppenhofer, 2017): This model approximates posterior entropy from several predictions made for each individual instance (see below).[6]

• **Bayesian Uncertainty (BU)** (Feinman et al., 2017): This model ranks instances with respect to the uncertainty (variance) in their stochastic predictions.[7]

BU estimates an uncertainty score for each individual instance by generating $T = 50$ predictions for the instance from a distribution of network configurations. The prediction disagreement tends to be common among spurious instances (high uncertainty) but rare among genuine instances (low uncertainty). Uncertainty of instance x with predictions $\{\mathbf{y}_1, \ldots, \mathbf{y}_T\}$ is computed as follows:

$$\frac{1}{T}\sum_{i=1}^{T}\mathbf{y}_i^\top\mathbf{y}_i - \Big(\frac{1}{T}\sum_{i=1}^{T}\mathbf{y}_i\Big)^\top\Big(\frac{1}{T}\sum_{i=1}^{T}\mathbf{y}_i\Big).$$

Variational inference (VI) (Rehbein and Ruppenhofer, 2017; Hovy et al., 2013) detects spurious instances by approximating the posterior $p(y|x)$ with a simpler distribution $q(y)$ (called variational approximation to the posterior) which models the prediction for each instance. The model jointly optimizes the two distributions through EM: in the E-step, q is updated to minimize the divergence between the two distributions, $D(q||p)$; in the M-step, q is kept fixed while p is adjusted. The two steps are repeated until convergence. Instances are then ranked based on their posterior entropies. Similar to BU, we generate $T = 50$ predictions for each instance.

For both BU and VI baselines, we apply a dropout rate of 0.5 after the first and last hidden

[6] http://isi.edu/publications/licensed-sw/mace/

[7] http://github.com/rfeinman/detecting-adversarial-samples

layers of our downstream networks to generate predictions. See (Gal and Ghahramani, 2016) for the ability of dropout neural networks in representing model uncertainty.

3.5 Experimental Results

The overall mean average precisions (MAPs) of different models on synthetic and real-world datasets are reported in Table 2. For the synthetic dataset (Addition), we report average MAP across all noise levels, and for real-world datasets (Twitter and Reddit), we report average MAP at their corresponding noise levels obtained from corresponding TREC pools. We use t-test for significance testing and asterisk mark (*) to indicate significant difference at $\rho = 0.05$ between top two competing systems.

The results show that Leitner (Lit) and Bayesian uncertainty (BU) models considerably outperform prediction probability (PP) and curriculum learning (CL) on both synthetic and real-world datasets. In case of real-world datasets, we didn't find significant difference between top two models perhaps because of the small size of corresponding TREC pools (198 Twitter posts and 152 Reddit posts, see Table 1). Overall, BU and Lit show average MAP of 0.81, and 0.85 on the synthetic dataset and 0.15, 0.22 on real-world datasets respectively. The higher performance of Lit indicates that spurious instances often appear in q_0. The lower performance of CL, however, can be attributed to its training strategy which may label spurious instances as easy instances if their loss values are smaller than the loss threshold (section 2.1). The large difference between the performances of Lit and CL (two methods based on repeated scoring across training epochs) shows that the way that repetition is utilized by different methods largely affects their final performance in spotting spurious instances. In addition, VI shows lower performance than BU and Lit on synthetic data, but comparable performance to BU on real-world datasets.

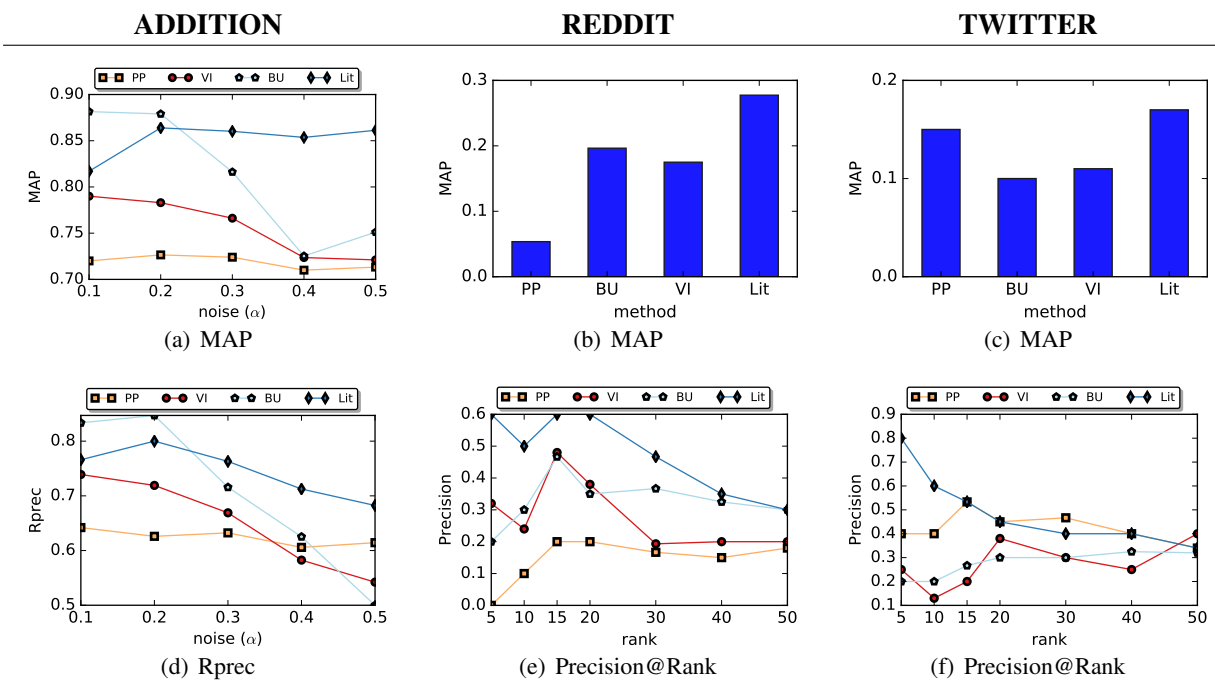
Furthermore, the results show that the performance of all models are considerably lower on real-world datasets than the synthetic dataset. This could be attributed to the more complex nature of our real-world datasets which leads to weaker generalizability of downstream learners on these datasets (see next section for discussion on training performance). This can in turn inversely affect the performance of different spotters, e.g. by encouraging most instances to be considered as hard and thus placed in lower queues of Lit or in the hard batch of CL, or by increasing the prediction uncertainty and entropy in case of BU and VI respectively. In addition, as we mentioned before, we carefully setup the annotation task to minimize the chances of spurious labels in the resulting annotations. Therefore, we expect a considerably smaller fraction of spurious instances in our real-world datasets.

	Synthetic noise = $[0.1, 0.5]$	**Real-world** noise = $\{0.28, 0.35\}$
PP	0.719	0.100
CL	0.718	0.067
VI	0.757	0.142
BU	0.811	0.148
Lit	**0.851***	**0.225**

Table 2: Average overall MAP performance across datasets and noise levels.

Figures 4(a) and 4(d) report MAP and precision after all spurious instances have been retrieved (Rprec) on Addition at different noise levels respectively; note that $\alpha = 0.5$ means equal number of spurious and genuine instances in training data (here, we do not report the performance of CL due to its lower performance and for better presentation). First, the results show that Lit and BU considerably outperform PP and VI. Furthermore, BU shows considerably high performance at lower noise levels, $\alpha \leq 0.2$, while Lit considerably outperforms BU at greater noise levels, $\alpha > 0.2$. The lower performance of BU at higher noise levels might be because of the poor generalizability of LSTM in the context of greater noise which may increase the variance in the prediction probabilities of most instances (see section 3.6 for our note on training performance). In terms of average Rprec, the overall performance of PP, CL, VI, BU, and Lit models is 0.62, 0.57, 0.65, 0.70, and 0.74 respectively on the Addition dataset across all noise levels (see the corresponding values for MAP in Table 2). The lower Rprec values than MAP indicate that some spurious instances are ranked very low by models. These are perhaps the most difficult spurious instances to identify.

For the real-world datasets, we only report MAP and P@r (precision at rank r) as spurious/genuine labels are only available for those instances that make it to the TREC pool but not for all instances. The results on Reddit, Figures 4(b) and 4(e) respectively, show that Lit outperforms other mod-



(a) MAP (b) MAP (c) MAP

(d) Rprec (e) Precision@Rank (f) Precision@Rank

Figure 4: Models performance across datasets. MAP indicates mean average precision, Rprec indicates precision after all spurious instances are retrieved, and P@r indicates precision after r instances are retrieved. In (a) and (b) performance values are reported at different noise levels $\alpha \in (0, 0.5]$.

els, but VI and BU show comparable MAP (in contrast to their performance on Addition). Furthermore, Figure 4(e) shows that Lit generates a more accurate ranked list of spurious instances and consistently outperforms other models at almost all ranks. In particular, it maintains a MAP of around 60% at rank 20, while other models have consistently lower MAP than 50% at all ranks.

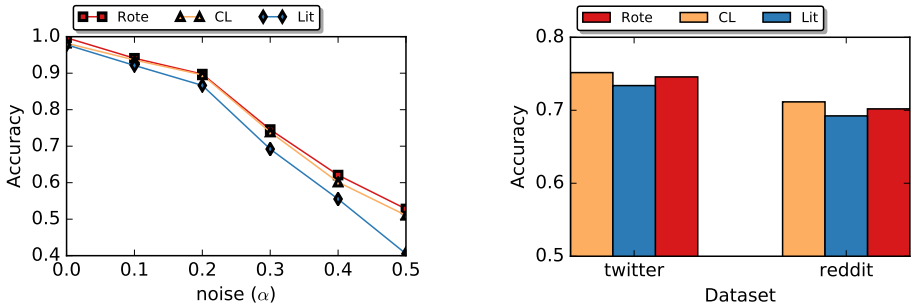
The results on the Twitter dataset, Figures 4(c) and 4(f), show that Lit outperforms other models. However, interestingly, PP outperforms BU in terms of both MAP and P@r across almost all ranks. This result could be attributed to the substantial annotation agreement on Twitter dataset (Fleiss' $\kappa = 0.66$) which could make network predictions/loss values more representative of gold labels. Figure 4(f) also shows that Lit is the most precise model in identifying spurious instances. Note that P@5 is an important metric in search applications and as Figures 4(e) and 4(f) show, at rank 5, Lit is 2-3 times more precise than the best-performing baseline on our real-world datasets.

Given *any* dataset and its corresponding neural network, our Leitner model simultaneously trains the network and generates a ranked list of spurious instances in the dataset. For this purpose, the model tracks loss values and occurrences of instances in the lower Leitner queue during training.

3.6 Notes on Training Performance

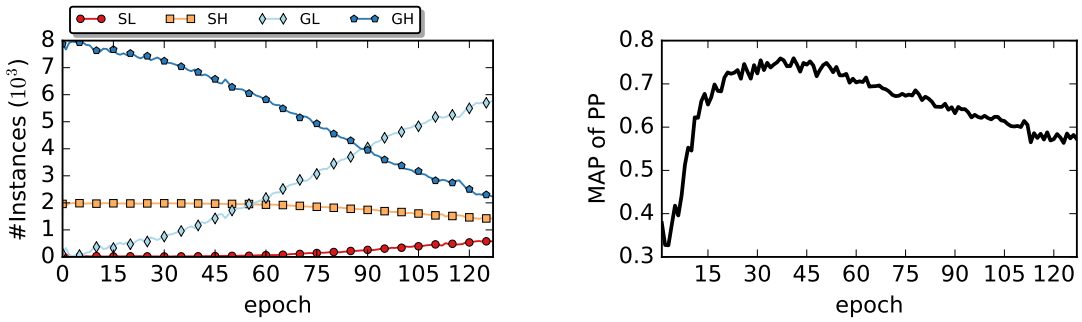
Figure 5(a) shows the accuracy of the LSTM network (Sutskever et al., 2014) trained with different training regimes on the validation data of Addition with different noise levels; note that Rote represents standard training where at each iteration all instances are used to train the network. As the results show, at lower noise levels, the training performance (i.e. the generalizability/accuracy of the LSTM network) is generally high and comparable across different training regimes, e.g. close to 100% at $\alpha = 0$. However, Lit leads to a slightly weaker training performance than CL and Rote as the noise level increases. This is because Lit learns from spurious instances more frequently than genuine ones. This may decrease the training performance of Lit, especially with greater amount of noise in data. However, this training strategy increases the spotting performance of Lit as spurious instances seem to occur in lower queues of Leitner more frequently, see Figure 4.

In addition, the accuracy of `fastText` (Joulin et al., 2017) is reported in Figure 5(b). The results show that different training regimes lead to comparable performance on both datasets (accuracy of around 75% and 70% on Twitter and Reddit respectively). The relatively lower training per-



(a) Addition/LSTM (b) Twitter and Reddit/`fastText`

Figure 5: Accuracy of LSTM/`fastText` trained with different schedulers. Rote: standard training.



(a) Spurious/genuine instances with low/high loss (b) MAP of prediction loss (PP)

Figure 6: Behavior of prediction loss across training epochs. (a) S: spurious, G: genuine; L: low, H: high.

formance on these datasets can contribute to the weaker performance of spotters on these datasets.

3.7 Discussion

We first report insights on why prediction loss alone is not enough to identify spurious instances. For this analysis, we track the loss of spurious and genuine instances at each training iteration.[8] Figure 6(a) shows the number of spurious/genuine instances with low/high loss at each epoch; where, we use the average loss of *correctly* classified training instances at each epoch as a pivot value to determine the low and high loss values for that epoch. Initially, almost all spurious and genuine instances have high loss values (see SH and GH in Figure 6(a)). However, the sheer imbalance of genuine instances relative to spurious instances means that there will still be a relatively large number of genuine instances with large loss - these are simply difficult instances. Furthermore, the number of spurious instances with lower loss values (SL) slowly increases as the network gradually learns the wrong labels of some spurious instances; this, in turn, decreases the expected loss of such instances. Since PP merely ranks instances based on loss values, the above two factors may cause some spurious instances to be ranked *lower* than genuine ones by PP; see Figure 6(b) for MAP of PP in detecting spurious instances at every iteration. Using queue information from the Leitner system adds information that loss alone does not; we suspect that the learner can find principled solutions that trade off losses between one difficult genuine instance and another (causing them to bounce between q_0 and higher queues) without harming total loss, but that the more random nature of spurious instances means that they are consistently misclassified, staying in q_0. Verifying this hypothesis will be the subject of future work.

For our second analysis, we manually inspect highly ranked instances in q_0 of Lit. We use the synthetic dataset bAbi (Weston et al., 2016) which is a systematically generated QA dataset for which the task is to generate an answer given a question and its corresponding story. As the learner, we use an effective LSTM network specifically developed for this task.[9] Table 3 shows sample instances from bAbi which are highly ranked by

[8] Here we use Addition with $N = 10$K training instances and noise level of $\alpha = 0.2$.

[9] https://github.com/fchollet/keras/blob/master/examples/babi_rnn.py

Story: Mary traveled to the garden. Daniel went to the garden. Mary journeyed to the kitchen. Mary went **back to the hallway**. Daniel traveled to the office. Daniel moved to the garden. Sandra went back to the kitchen. John traveled to the bathroom. **Question**: Where is Mary? **Answer**: hallway
Story: John went to the office. Daniel journeyed to the office. Sandra picked up the football **there**. Sandra went to the bedroom. Sandra left the football there. Sandra went **back to the kitchen**. Sandra traveled to the hallway. Sandra moved to the garden. **Question**: Where is the football? **Answer**: bedroom

Table 3: Sample inconsistencies in bAbi dataset.

Lit. We observe inconsistencies in the given stories. In the first case, the story contains the sentence "Mary went *back* to the hallway," while the previous sentences indicate that Mary was in the "garden/kitchen" but not "hallway" before. In the second case, the sentence "Sandra picked up the football *there*" is inconsistent with story because the word "there" doesn't refer to any specific location. We conjecture that these inconsistencies can mislead the learner or at least make the learning task more complex. Our model can be used to explore language resources for such inconsistencies.

4 Related Work

There is broad evidence in psychology that shows human ability to retain information improves with repeated exposure and exponentially decays with delay since last exposure. Ebbinghaus (1913, 2013), and recently Murre and Dros (2015), studied the hypothesis of the exponential nature of forgetting in humans. Three major indicators were identified that affect memory retention in humans: delay since last review of learning materials and strength of human memory (Ebbinghaus, 1913; Dempster, 1989; Wixted, 1990; Cepeda et al., 2006; Novikoff et al., 2012), and, more recently, difficulty of learning materials (Reddy et al., 2016).

The above findings show that human learners can learn efficiently and effectively by increasing intervals of time between subsequent reviews of previously learned materials (spaced repetition). In (Amiri et al., 2017), we built on these findings to develop efficient and effective training paradigms for neural networks. Previous research also investigated the development of cognitively-motivated training paradigms named curriculum learning for artificial neural networks (Bengio et al., 2009; Kumar et al., 2010). The difference between the above models is in their views to learning: curriculum learning is inspired by the learning principle that training starts with *easier* concepts and gradually proceeds with more difficult ones (Bengio et al., 2009). On the other hand, spaced repetition models are inspired by the learning principle that effective and efficient learning can be achieved by working more on difficult concepts and less on easier ones.

In this research, we extend our spaced repetition training paradigms to simultaneously train artificial neural networks and identify training instances with potentially wrong labels (spurious instances) in datasets. Our work is important because spurious instances may inversely affect interpretations of the data, models developed from the data, and decisions made based on the data. Furthermore, spurious instances lead to unrealistic comparison among competing systems if they exist in test data.

5 Conclusion and Future Work

We present a novel approach based on queueing theory and psychology of learning to identify spurious instances in datasets. Our approach can be considered as a scheduler that iteratively trains a downstream learner (e.g. a neural network) and detects spurious instances with respect to their difficulty to learn during the training process. Our approach is robust and can be applied to any dataset without modification given a neural network designed for the target task of the dataset.

Our work can be extended by: (a) utilizing several predictions for each training instance, (b) investigating the extent to which a more sophisticated and effective downstream learner can affect the performance of different spotters, (c) developing models to better distinguish hard genuine instances from spurious ones, and (d) developing ranking algorithms to improve the performance of models on real-world datasets.

Acknowledgments

We thank anonymous reviewers for their thoughtful comments. This work was supported by National Institutes of Health (NIH) grant R01GM114355 from the National Institute of General Medical Sciences (NIGMS). The content is solely the responsibility of the authors and does not represent the official views of the NIH.

References

Hadi Amiri, Timothy Miller, and Guergana Savova. 2017. Repeat before forgetting: Spaced repetition for efficient and effective training of neural networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 2401–2410.

Sumit Basu and Janara Christensen. 2013. Teaching classification boundaries to humans. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*. AAAI Press, pages 109–115.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*. pages 41–48.

Nicholas J Cepeda, Harold Pashler, Edward Vul, John T Wixted, and Doug Rohrer. 2006. Distributed practice in verbal recall tasks: A review and quantitative synthesis. *Psychological bulletin* 132(3):354–380.

Frank N Dempster. 1989. Spacing effects and their implications for theory and practice. *Educational Psychology Review* 1(4):309–330.

Markus Dickinson and W Detmar Meurers. 2003. Detecting errors in part-of-speech annotation. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1*. pages 107–114.

Hermann Ebbinghaus. 1913. *Memory: A contribution to experimental psychology*. 3. University Microfilms.

Hermann Ebbinghaus. 2013. Memory: A contribution to experimental psychology. *Annals of neurosciences* 20(4):155.

Eleazar Eskin. 2000. Detecting errors within a corpus using anomaly detection. In *Proceedings of North American chapter of the Association for Computational Linguistics conference*. pages 148–153.

Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. 2017. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410* .

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin* 76(5):378.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*. pages 1050–1059.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. *International Conference on Learning Representations* .

Melody Y. Guan, Varun Gulshan, Andrew M. Dai, and Geoffrey E. Hinton. 2017. Who said what: Modeling individual labelers improves classification. *CoRR* abs/1703.08774.

Aakar Gupta, William Thies, Edward Cutrell, and Ravin Balakrishnan. 2012. mclerk: enabling mobile crowdsourcing in developing regions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, pages 1843–1852.

Haibo He and Edwardo A Garcia. 2009. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering* 21(9):1263–1284.

Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. 2017. On detecting adversarial perturbations. *International Conference on Learning Representations* .

Dan Hendrycks and Kevin Gimpel. 2016. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *International Conference on Learning Representations* .

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with mace. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Atlanta, Georgia, pages 1120–1130.

Lu Jiang, Deyu Meng, Shoou-I Yu, Zhenzhong Lan, Shiguang Shan, and Alexander Hauptmann. 2014. Self-paced learning with diversity. In *Advances in Neural Information Processing Systems 27*, pages 2078–2086.

Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander G. Hauptmann. 2015. Self-paced curriculum learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI'15, pages 2694–2700.

Armand Joulin, Edouard Grave, and Piotr Bojanowski Tomas Mikolov. 2017. Bag of tricks for efficient text classification. *Proceedings of the 15th Eropean Chapter of the Association for Computational Linguistics* page 427.

Alex Krizhevsky. 2009. Learning multiple layers of features from tiny images. *Technical Report, University of Toronto* .

M Pawan Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*. pages 1189–1197.

Walter S Lasecki, Christopher D Miller, and Jeffrey P Bigham. 2013. Warping time for more effective real-time crowdsourcing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, pages 2033–2036.

Sebastian Leitner. 1974. *So lernt man lernen: der Weg zum Erfolg (How to learn to learn)*. Herder.

Hrafn Loftsson. 2009. Correcting a pos-tagged corpus using three complementary methods. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*. pages 523–531.

Jaap MJ Murre and Joeri Dros. 2015. Replication and analysis of ebbinghaus' forgetting curve. *PloS one* 10(7):e0120644.

Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. 2013. Learning with noisy labels. In *Advances in neural information processing systems*. pages 1196–1204.

Timothy P Novikoff, Jon M Kleinberg, and Steven H Strogatz. 2012. Education of a model student. *Proceedings of the National Academy of Sciences* 109(6):1868–1873.

Siddharth Reddy, Igor Labutov, Siddhartha Banerjee, and Thorsten Joachims. 2016. Unbounded human learning: Optimal scheduling for spaced repetition. In *Proceedings of SIGKDD*. pages 1815–1824.

Ines Rehbein and Josef Ruppenhofer. 2017. Detecting annotation noise in automatically labelled data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1160–1170.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2010. From baby steps to leapfrog: How less is more in unsupervised dependency parsing. In *North American Chapter of the Association for Computational Linguistics*. pages 751–759.

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

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2016. Towards ai-complete question answering: A set of prerequisite toy tasks. *International Conference on Learning Representations* .

John T Wixted. 1990. Analyzing the empirical course of forgetting. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 16(5):927.

Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. 2015. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 2691–2699.

Wojciech Zaremba and Ilya Sutskever. 2014. Learning to execute. *arXiv preprint arXiv:1410.4615* .

Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. 2016. Improving the robustness of deep neural networks via stability training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 4480–4488.

Xingquan Zhu and Xindong Wu. 2004. Class noise vs. attribute noise: A quantitative study. *Artificial Intelligence Review* 22(3):177–210.

Xingquan Zhu, Xindong Wu, and Qijun Chen. 2003. Eliminating class noise in large datasets. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*. AAAI Press, ICML'03, pages 920–927.

The Timing of Lexical Memory Retrievals in Language Production

Jeremy R. Cole and David Reitter
College of Information Sciences and Technology
The Pennsylvania State University
University Park, PA
jrcole,reitter@psu.edu

Abstract

This paper explores the time course of lexical memory retrieval by modeling fluent language production. The duration of retrievals is predicted using the ACT-R cognitive architecture. In a large-scale observational study of a spoken corpus, we find that language production at a time point preceding a word is sped up or slowed down depending on activation of that word. This computational analysis has consequences for the theoretical model of language production. The results point to interference between lexical and phonological stages as well as a quantifiable buffer for lexical information that opens up the possibility of non-sequential retrievals.

1 Introduction

Speech varies greatly in fluency, and some of its speed variation can be traced to the utterance spoken (Jespersen, 1992). Low-frequency words, for instance, are known to slow down speech (e.g., Bell et al., 2009). Variables correlated with fluency give valuable cues to the architecture of the language processing system. However, a model to explain these data has yet to emerge.

In this paper, we propose a cognitive model of fluency, in which lexical memory retrievals may explain some of the variability in speech rates. In particular, frequency, context and recent uses together have the potential to quantify retrieval delays through *activation* (Anderson, 1991). Activation, in its most common usage, refers to the way nodes in semantic networks become easier to retrieve after adjacent nodes have been activated, typically through a presentation (Collins and Loftus, 1975). In particular, activation makes a direct claim that more highly activated words require less time to retrieve, and vice versa (Anderson, 1983).

The language production process as a whole likely requires some amount of sequential processing. For instance, the standard model proposes that an idea is generated, lexicalized, grammatically and morphologically encoded, and only then phonologically encoded (Bock and Levelt, 2002). Still, most models of language production presuppose some amount of planning of output (e.g., Pickering and Garrod, 2013), so we could instead divide language production into planning this output and the actual process of outputting. The overlap and relationship of these processes is not fully understood, but given that most output is likely planned, the scale at which the planning takes place and the amount of time between planned output and the actual process of outputting remains unclear. However, if interactions between processes are observed, then we can likewise see when they overlap in time.

To summarize, we are suggesting that some of the variance in speech rate is not due to the linguistic properties of the words currently or about to be outputted, but the words still in the planning phase. We propose a model that uses a buffer of several words between initial retrieval and output, during which grammatical and morphological encoding take place. We examine this by calculating retrieval activation for a word and evaluating the influence of that activation on the empirical speech timing several words beforehand, using the Switchboard corpus. The effect of activation is distributed over preceding words in a way that is characteristic of a shared-resource, buffer-based account of language production.

2 Related Work

2.1 Stages of Language Production

Grammatical encoding can be divided into *functional* and *positional* processing steps (Bock and Levelt, 2002). The functional step selects lexical items and assigns functions, while

Proceedings of NAACL-HLT 2018, pages 2017–2027
New Orleans, Louisiana, June 1 - 6, 2018.

the positional step then combines the items to produces constituents. In our account, we expect that these mutually dependent steps work in parallel.

An important early part of functional processing retrieves lexical information, which we will examine in this paper. We evaluate the consequences of lexical access, which is assumed to be affected by the cost associated with any retrieval from declarative memory. Much discussion in this area has concerned the question whether lexical access happens in a single stage (Dell et al., 1997) or in multiple stages and overlaps with grammatical encoding (Caramazza, 1997; Roelofs et al., 1998; Caramazza, 2006). Here, we follow ACT-R's serial and partially symbolic nature, which in turn leads to some theoretical commitments to non-parallel processing: language production is staged and discrete. Nonetheless, each stage can be composed of several steps, and steps from syntactic and phonological processing likely interleave. This is compatible with empirical findings and the overall theoretical debate (Ferreira and Slevc, 2007). The precise timeline of processing is unclear, but as we will argue in this paper, large-scale speech data can give us usable clues to that effect.

2.2 Incrementality in Language Production

The second issue we address concerns the timing of memory retrievals, which is also related to the idea of incremental processing. It is a commonly implied assumption that language processing proceeds incrementally. In grammatical encoding, this property concerns when and in which order syntactic choices are made. For instance, all of them could be made before phonological processing starts (non-incremental case), or they could be made in order as necessary. Existing high-level models of language production proceed incrementally at various steps in a chain of content selection, aggregation and sentence realization (e.g., Bock and Levelt, 2002; Guhe, 2007).

Ferreira (1996) makes an argument for incrementality, based on the observation that competitive syntactic alternatives facilitate production rather than making it more difficult. An incremental account of sentence realization would predict such an effect, as syntactic "flexibility" introduced by the alternatives makes it easier to find a workable syntactic decision. By contrast, without incremental commitment to each structure, competing material slows down the process, because it would lead to combinatory explosion. However, later results establish nuance. Ferreira and Swets (2002) show that incremental production is possible, but it is "under strategic control"; it depends on semantic information, and it could be modulated by external factors, such as stress.

If processing were fully incremental, then it would follow that lexical memory retrievals are also fully incremental. The order words are retrieved in would be the same as the order words are eventually outputted in. However, if other features modulate this, then it would imply that incremental processing is instead variable, as suggested by earlier accounts.

2.3 Speech Rates

Several studies have illustrated the effects of frequency, recency, and context (Bell et al., 2009; Arnon and Priva, 2014) on speech rates. These studies motivated our modelling choices, as recency, frequency, and context are also the key components of the ACT-R theory of memory.

Recent research has found a correlation between rate of speech and the information content of that speech. (e.g., Arnon and Cohen Priva, 2013). Thus far, this correlation lacks a precise theory with a cognitive explanation. By producing a cognitive model of these speech rates, we provide evidence for such a theory.

2.4 Lexical Retrieval

This paper examines the time course of lexical retrieval for the case of fluent, naturalistic speech. Different facets of language can interfere with lexical retrieval in different contexts, which provides evidence toward an architecture: Schriefers et al. (1990) found that semantic, but not phonological material can cause interference, suggesting that the two are represented separately. Ratcliff and McKoon's (1989) study focuses on sentence retrieval and found that semantic information is also retrieved in stages. Here, we seek to model the retrieval process in the context of fluent speech.

There are a number of memory models in the literature that provide accounts of the timing of lexical access. For instance, classic models such as Dell's (1986)'s model of spreading activation during language production and Levelt et al.'s (1999) WEAVER++ model both provide quantitative values for retrieval times based on the form of a word.

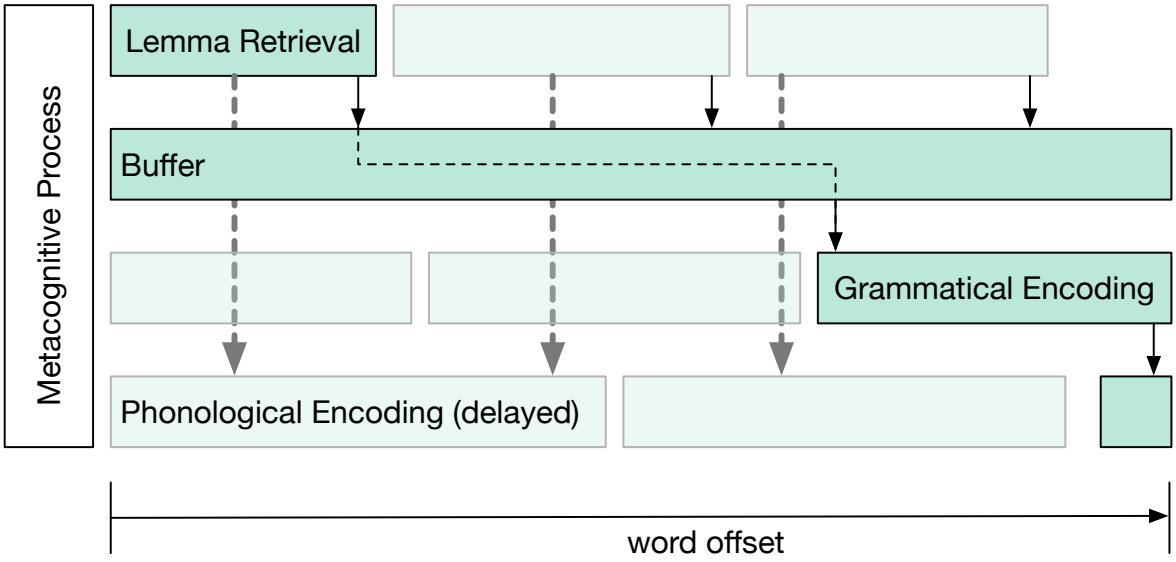


Figure 1: Our psychological model assumes that language production involves several parallel processes, and that retrieval of lemmas can interact with concurrent retrieval and/or encoding of phonological representations (dashed arrows) due to concurrent resource usage. Lemmas are retrieved several words before they are spoken. Their exact point of retrieval could depend on other factors. Likewise, while we represent phonological retrieval/encoding as a separate process for clarity, we make no claim to what extent these processes overlap.

Models such as Rapp and Goldrick (2000) focus on modeling speech errors based on word activation and context. Our model differs from these in that it attempts to model retrievals from fluent speech rates, rather than single word lexical retrieval based on picture naming tasks. Finally, while speech errors are likely related to failed lexical memory retrievals, we focus on speech that was eventually successfully retrieved and produced.

More relevantly, Dell and O'Seaghdha (1992) examine the time course of lexical access in language production. In particular, they use series of three words and EEG data to estimate lexical retrieval time. However, the lab setting it took place in precluded it as a study of naturalistic speech. Further, their model of the effects of word-properties relied on primarily qualitative attributes, such as semantic or phonetic relatedness. In particular, they find additional evidence for lemma and phonological retrieval taking place in separate stages, based on inhibition and facilitation effects. The goal of the present study is to expand the examined time frame in the hopes of replicating their argument on naturalistic speech while viewing effects found throughout, rather than just a three word window.

3 ACT-R Model

To motivate the corpus-based empirical analysis, we first describe our high-level model of the language production process. Our method primarily relies on simulating the state of lexical declarative memory during language production. After we simulate the memory retrievals for each word, we can compare this information to the actual empirical timing data in the corpus. In particular, we rely on Anderson's (1983) original account of memory. This framework was selected rather than newer or more task-specific frameworks as it is the same underlying memory model of ACT-R, which has been used to explain a wide variety of language phenomena (e.g., Vasishth and Lewis, 2004; Reitter et al., 2011), but also has been used to explain everything from decision-making (e.g., Marewski and Mehlhorn, 2011) to visual attention in graphical user interfaces (e.g., Byrne et al., 1999). Thus, by using this model, our work naturally builds upon a large body of work, using the same mechanisms to explain a variety of tasks.

Figure 1 illustrates how lemma retrieval of a target word affects phonological encoding of speaking of an earlier word. Retrieval timing is computationally estimated using the cognitive architecture ACT-R, and we assume that this retrieval

time proportionally affects phonological encoding. This can take place strategically, via a metacognitive process that coordinates these different modules, or via interference because both processes share declarative memory resources.

Our model of lexical memory is principally based on Anderson (1983)'s discussion of recency, frequency, and context effects. Activation (A) within the context of the ACT-R system is generally described by the sum of *base-level learning* (bll) and *spreading activation* (sa), which we adopt for our model as well (Anderson et al., 2004). Activation, can be defined as a linear combination of spreading activation base-level learning:

$$A(x) = sa(x) + bll(x) \tag{1}$$

For our purposes, we consider x to refer to an individual word. Base-level learning refers to the frequency and recency effects. In the base-level learning equation, it can refer to both because of the *decay* parameter, d, which causes more recent presentations to be more important, with older presentations (signified by their time of presentation, t) becoming exponentially less relevant. These older presentations, when considered together, add to the equation through their sheer quantity, providing the frequency effect, defined as:

$$bll(x) = \log\left(\sum_{i \in P_x} t_i^{-d}\right) \tag{2}$$

In this equation, P_x refers to the list of x's presentations, so t_i is the time from that presentation to the present. Naturally, for something with as many presentations as any given word, it is infeasible to computationally manage that sum. However, the full equation can be approximated using only the total number of presentations and the k most recent presentations and $n_x = |P_x|$ (Petrov, 2006).

$$bll(x) \approx \log\left[\sum_{i}^{k} t_i^{-d} + \frac{(n_x - k)\left(t_{n_x}^{1-d} - t_k^{1-d}\right)}{(1-d)\left(t_{n_x} - t_k\right)}\right] \tag{3}$$

While Petrov (2006) shows that the equation is close even for $k = 1$, we used $k = 5$ to more closely approximate the original equation. We then use the ACT-R default for the decay parameter, 0.5. Note that it has been suggested (e.g., Lewis and Vasishth, 2005; Cole et al., 2017) that this decay parameter could be different for language processing. In this work, we are only concerned with relative, rather than absolute values for a word's activation in memory.

In order to compute the total number of presentations, we relied on a fairly simple estimate. We multiply the number of seconds a person has been alive with the average speaking rate and that word's frequency to obtain an estimate of the amount of times a person has encountered that word; it is difficult to measure the difference between being exposed to the lexical form of the word compared to the phonological form, and it is even harder to measure any subsymbolic exposure due to thought. Still, using this formula, a unigram score computed by SRILM (Stolcke, 2002) applied to the British National Corpus, the average speaking rate of Switchboard participants (197 words/minute) as computed by Yuan et al. (2006), and the average age of Switchboard participants (37) (Godfrey et al., 1992), we can compute a baseline number of presentations for every word in Switchboard.

Next, computing spreading activation on a corpus as described in Anderson (1983) would likewise be computationally intractable. However, Pirolli et al. (2006) showed that for large sample sizes of language, Pointwise-Mutual Information is nearly identical. Therefore, we use Semilar's PMI database computed on the Wikipedia corpus (Rus et al., 2013; Church and Hanks, 1990).

In the ACT-R system, generally only items currently in working memory affect memory retrievals (Anderson et al., 2004). Likewise, we maintain the n previous words in a buffer to compute their spreading activation to the next word. We used $n = 5$ as an estimate for working memory size in language, as found in a reading task (Daneman and Carpenter, 1980). For our model, we compute the spreading activation between retrieved word, x, and each word in working memory, y, as:

$$sa(x) \approx \sum_{y}^{n} pmi(x, y) = \sum_{y}^{n} \log \frac{p(x, y)}{p(x)p(y)} \tag{4}$$

Once we have a value for activation, it's fairly simple to compute an estimate for retrieval time (RT) using the same equations from Anderson

(1983).

$$RT = I + 1/A - \frac{Ke^{-KA}}{(1 - e^{-KA})} \quad (5)$$

In this equation, I is an intercept, easily fitted with a linear model. As a parameter, K represents the cutoff time (in seconds) before there is a retrieval failure. This equation actually only represents the time required in the case of successful retrievals, which is nonetheless bounded by K, which in that sense could be thought of as the maximum possible time for a successful retrieval. While retrieval failures are part of normal ACT-R processing, they are not relevant to our model. Since our model is formed of already spoken words, they cannot represent retrieval failures. Thus, while the equation only represents successful retrievals, it is appropriate for our model. We chose the architectural default of 1.0 for K.

4 Methods

4.1 Corpus Analysis

The empirical speech data was taken from the Switchboard corpus (Godfrey et al., 1992) which is part of the Penn Treebank corpus (Marcus et al., 1993). This dataset consists of telephone conversations between strangers on a random topic, annotated to include the start and finish time for every word that has been spoken. Using our model of lexical memory as described in the previous section, we trace through the model and compute the activation of each word at its onset time.

Once the activation was computed for each word at the point when it was spoken, our goal was to observe its effect on overall speaking rates. In order to estimate when x was retrieved, we examined the speech some number of *words* back from word x. If words are spoken systematically more slowly or quickly based on word x's activation and their positional relationship to word x, then we can assume where words are spoken more slowly, retrievals are taking place. Where words are spoken more quickly, retrievals have finished. Importantly, since this is being computed at every sentence position, this should not capture positional effects. See Figure 1 for a visual depiction of our model of interference during lexical retrieval, which allows us to infer retrieval based on such interference.

While a naive model may expect lexical retrieval to occur immediately before grammatical or phonological encoding, this is not necessarily the case. Indeed, the amount of time before encoding may not be constant and may vary from word to word.

Our analysis of the corpus requires computing each word's *delay*, which is defined as the amount of time between the onsets of two sequential words, including any disfluencies that occur. As words themselves naturally can require different amounts of time to speak, we instead use the *adjusted* delay which is computed by taking the average of all of the durations of that word (as found in Switchboard) and subtracting it from the given duration. Thus, the adjusted delay could be a positive or a negative number, representing slowdowns and speedups, respectively. Throughout this paper, we use the term *delay* to actually refer to this adjusted delay. The delay referred to in Figure 1 is thus the adjusted delay: the difference between the expected delay based on the word form and the actual observed delay. To be clear, that means that if a delay term is not zero, there was a variation from the normal speed of processing, to either be quicker (negative delay) or slower (positive delay).

These speedups and slowdowns, and their relationship to retrieval time, allow us to make an argument about the interaction between lexical and phonological processing. From a statistical point of view, as we are comparing retrieval time and slowdowns in the same units, our linear model could be thought of as the percentage of retrieval time that is behaviorally reflected in language production.

4.2 Experiment

Data were analyzed with two related models. Initially, we tested an *interaction* model in order to test our hypothesis of the interaction between delay and offset (see Table 1). From this information, we use exploratory data analysis in the form of a *discrete* model, in order to explore the critical regions of the graph (see Table 2). From this exploratory data analysis, we present the pooled version of the discrete model for easier interpretation of our found effects (see Table 3). For both models, the activation of a target word and its expected retrieval time burden was computed, as were the delays for the n words preceding the target word. Importantly, note that in both models, when we refer to the expected retrieval time or activation, we

are referring to the target word, not any of the preceding words. Both models are concerned with the word *offset* (*i*), which refers to the number of interceding words between the given delay and the target word, such that $\mathrm{i} = 0$ refers to the word immediately before the target word.

In the interaction model, we are interested in the interaction term between word offset and delay: its goal is to show how the correlation changes with offset. In this model, every observation only uses a single offset, chosen randomly, for each target word. All of the other observations for that word are discarded. This is to ensure the observations are independent. The correlation coefficients of interest are the correlation of delay as a whole, and its interaction effect with offset. In general, the coefficient of offset by itself is likely capturing some distributional information about the data, rather than anything interesting with how it relates to memory retrievals. As a linear model:

$$RT \sim \text{delay} * \text{offset}$$

Meanwhile, the discrete model's observations consist of a word's expected retrieval time and the delays from previous words. Then, we make a linear model using each of the delays as a predictor. Note that in this notation, delay_i refers to the delay of offset word i. To reiterate, i represents how many interceding words there are between that offset word and the target word. As a linear model, this would be:

$$RT \sim \text{delay}_0 + \text{delay}_1 + ... + \text{delay}_n$$

The goal of the interaction model is to show the robustness of the slope associated with index, while the goal of the discrete model is to allow for a non-linear relationship between offset and the effect of delay on activation, examining up to 25 previous words. Exploring this non-linear relationship allowed us to infer the critical regions of this effect. Importantly, the discrete model's goal was to explore the significant relationship found in the interaction model more deeply, rather than to itself justify the effect.

Under the model shown in Figure 1, we expect that longer retrieval times of the target word are associated with slowdowns of speech production at some time before the target word is spoken. Earlier than that point, the target word should have no influence on speech production.

	Estimate	Std. Error	t-value	p-value
(Intercept)	.1796	.0002	728.715	$<$ .00001 ***
offset	.0015	.0010	1.1512	.011 **
delay	.0668	.0048	13.839	$<$ .00001 ***
delay*offset	-.0066	.0005	-12.411	$<$ .00001 ***

F-stat	DF	p-value	Adj R^2	Multi R^2
102.2	802055	$<$ .00001	0.0005	0.0005

Table 1: Linear regression predicting expected retrieval time of a target word as a function of the delay in speaking of a previous word at that offset.

	Estimate	Std. Error
I	.1844	.0002
d0	-.0033	.0001
d1	-.0011	.0001
d2	-.0006	.0001
d3	-.0002	.0001
d4	-.0001	.0001
d5	-.0001	.0001
d6	.0002	.0001
d7	.0001	.0001
d8	.0002	.0001
d9	-.0001	.0001
d10	.0002	.0001
d11	.0000	.0001
d12	.0002	.0001
d13	.0000	.0001
d14..d24	0.000.	0.000

Table 2: The linear effects model relating each discrete delay term with expected retrieval time. A higher number on the delay term signifies the number of words between the delayed word and the target word. This exploratory data analysis was done to inform the pooled model. Also see Figure 2.

5 Results and Discussion

If one focuses on the interaction model, our experiments yield a relatively counterintuitive result: namely, delay is correlated in the direction opposite to what is expected. One would imagine that delay and retrieval time should be positively correlated: if people are speaking words more slowly (positive delay), then likewise, their retrieval time should be higher. However, we discovered a robust effect in the opposite direction: higher delays imply shorter expected retrieval times, and shorter delays imply longer expected retrieval times. In other words, when people are expected to need the longest to retrieve words, they actually speak more quickly, and vice versa.

	Estimate	Std. Error	t-value	p-value	
(Intercept)	.1843	.0002	728.715	$< .00001$	***
early	-.0052	.0010	-46.36	$< .00001$	***
early(ns)	0.000	0.000	-0.8200	.412	
late	.0009	.0002	5.448	$< .00001$	***
late(ns)	.0002	.0002	1.263	.2070	

F-stat	DF	p-value	Adj R^2	Multi R^2
567.9	777924	$< .00001$	0.003	.003

Table 3: Pooled version of discrete linear model, based on critical regions from the graph. Regions are broken at 3, 5, and 14 respectively.

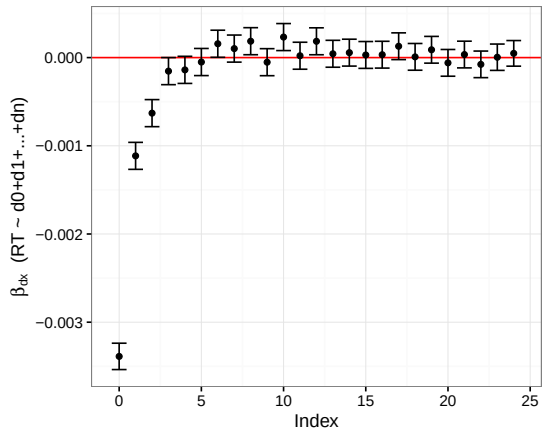


Figure 2: The discrete model's linear predictors (see also Table 2). Error bars represent normal 95% confidence intervals. This graph and Figure 3 have similar critical regions, which informed the pooled model presented in Table 3: 0-2 appear significant and negative, 3-4 are not significant (slightly negative), 5-14 are significant and positive, 15+ is not significant.

Examining the effect for larger offsets, however, we observe that the effect reverses before disappearing. Thus, we see an effect in the expected direction for the delays of word offsets 4 through 14. This is commensurate with word planning that takes place several words in advance rather than immediately before the word; likewise, the effect also disappears in the interaction model based on the interaction effect.

See also Figure 2 and Figure 3, which are visualizations of the discrete and interaction model, respectively. These graphs show how the relationship between activation of a word and speech delay develops over the offsets, i, before the word. While Figure 2 has its effects pulled directly from Table 2, Figure 3 is produced by raw data, defined by:

$$y(j,i) = \frac{A_j - \beta_0}{\text{delay}_i} \tag{6}$$

These graphs were designed to demonstrate how the effect switches from positive to negative as we move back from immediately before the word to earlier in the utterance. With the interaction model, we wanted to show statistical evidence for the pattern of effects; the discrete model quantifies the gradual fade to zero. We interpret the models as follows.

1. There is a strong negative correlation of the word delays with expected retrieval time for the words immediately before the target word. Since retrieval time is a function of activation, this would imply that the observable phonological effect happens later for more activated words, which are likely retrieved shortly before their use.
2. There is a weaker but significant positive correlation of the word delays with expected retrieval time for words about 5-14 words preceding the target word. These delays likely occur for words with less activation, whose retrievals are likely initiated early to ensure that there is enough time.
3. For words very far away from the target word, there is no reliable effect, implying that this is not just an effect of a cyclical information distribution.

6 General Discussion

These results confirm some classical findings on lexical retrieval, while adding a subtle but reliable new effect. Further, these findings have some implications for incrementality and uniform information density.

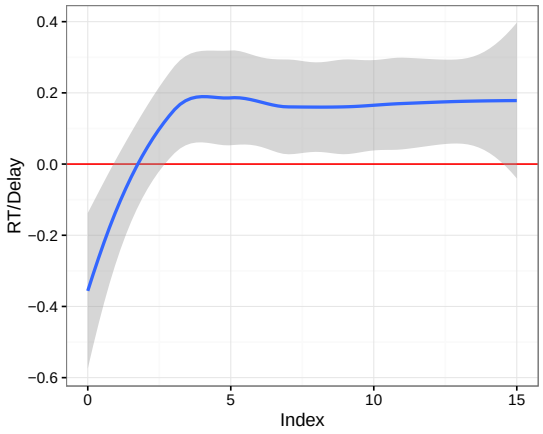


Figure 3: Smoothed correlation between delay and expected retrieval time across offset, created using a sample of raw data (representative of the interaction model). Effect disappears after offset 15, but full graph is not shown to avoid smoothing small but significant effects with non-significant effects.

In our discussion, we will frequently refer to the *activation* of a word. Recall that activation in the ACT-R sense is the inverse of the expected retrieval time: higher activation implies a shorter expected retrieval time. While retrieval time makes more sense in a time-predictive linear model, it is easier to interpret our results based on its relationship to activation.

6.1 Lexical Retrieval

It is difficult to separate the lexical retrieval effects we found into the two categories of retrievals described by Levelt (1992): a lemma retrieval and a later phonological retrieval. However, this is not to claim that they cannot be, but simply that our methodology did not easily allow us to. A commonly implied assumption is that lemma retrievals shouldn't interfere with phonological processes (e.g., Schriefers et al., 1990), though it is difficult to know if a speech slowdown is due to a phonological or semantic interference due to our experimental setup. However, since in our experiment, effects are still observed at large distances from the target words, either phonological forms can be retrieved in a non-incremental way (possibly even before lemmas for other words are retrieved), or the retrieval of the lemma does interfere with phonological encoding in some way; for instance, by activating related phonological forms. Still, we ultimately find the same pattern of effects as Dell and O'Seaghdha (1992): facilitatory effects close to the target word, with inhibitory effects further away. The primary difference is the time frame, which is possibly due to their experimental setup.

6.2 Process Model

We found a surprising effect: words with higher activation are not spoken more quickly, but more slowly. This also applies to the words that immediately precede them. However, if we look further back, we see a robust effect in the expected direction: if the approaching word has a high activation, they are said more quickly, but if the approaching word has a low activation, they are said more slowly. We argue that this slowdown is the result of shared resources between phonological and grammatical encoding, and as activation directly predicts retrieval time, we posit that word retrievals are part of what causes slowdowns. The corresponding speedups could be because the work of planning the sentence up to that point is then done. The most important prediction of this is that it means low activation words are retrieved earlier, which would imply that there is some cognitive strategy facilitating the necessity of initiating early retrievals for low activation words.

6.3 Incrementality

These results provide information about the timing of memory retrievals, given that such retrievals are related to activation. As activation is inherently related with how long a memory retrieval should take, it makes sense there are some cognitive strategies for coping with this disparity in order to produce seemingly fluent dialogue. That strategy involves buffering: retrieving and storing the words that will need longer to retrieve, based on the structure of the sentence.

Further, this type of buffering strategy could be part of the strategy that Ferreira and Swets (2002) refer to, when they propose the incrementality of language production is under "strategic control." While a purely incremental strategy might have interlocutors retrieve in a purely incremental fashion, there are some hiccups: certain words take longer to retrieve than others. By this logic, if grammatical encoding proceeds in a purely incremental fashion, then lexical retrieval does not, and vice versa. Thus, it is reasonable to believe that the grading of incrementality found in natural human discourse is not only variable from situation to situation, but it may be variable amongst competing processes for any given situation.

6.4 Uniform Information Density

Let's consider an additional explanation. The *Constant Entropy Rate* Hypothesis (Genzel and Charniak, 2002) posits that lexical material is distributed across a sentence (and other units) such that its information is held approximately constant. Could a difficult-to-retrieve, slow word at position j be likely to be combined with easier-to-retrieve, high-frequency words at positions $j-4...j-1$, causing the significantly increased speech rate we found there?

The model of buffered retrievals, along with the empirical evidence, may provide a cognitive mechanism that results in an approximately constant entropy rate. Thus, Uniform Information Density (UID, e.g., Jaeger, 2010) could be considered a consequence of the cognitive procedures involved in retrieving syntactic-lexical items from

declarative memory while grammatically encoding those materials retrieved earlier.

7 Future Work

Our work opens up several possible avenues for future research. While it is unclear if syntax rules are retrieved from some form of implicit memory (e.g., Reitter et al., 2011), lexical items clearly are. Syntactic processing could potentially adapt to working memory, rather than itself guide lexical retrievals (e.g., Cole and Reitter, 2017). By this argument, memory retrieval is a largely automatic, rather than attention-driven process, and syntax makes use of what is available to produce fluent dialogue. In this type of model, the constant size of the retrieval buffer would provide a clear corollary to Uniform Information Density.

Furthermore, this paper does not clearly differentiate between lemma and phonological retrieval. Although we do not expect phonological forms to be retrieved as early as the effects we are seeing, we also do not expect lemma retrieval to have effects on phonological encoding. A computationally implemented process model could explore these effects in more detail.

Lastly, this study provides another mechanism by which non-sequential dependencies in language production are observed. It seems possible that non-incremental language processing can be explained as a process that involves general memory mechanisms including cue-based memory retrieval. What is in question is whether we really process local syntax using structured, memory-hungry models (i.e., with syntax trees); we note that in natural language processing, skip-grams can capture local, non-incremental relationships among words. Thus, the relationship between working memory, syntax trees, and skip-grams appears to be of continued interest.

8 Conclusions

In this paper, we explore the process of lexical memory retrieval in the context of language production. In contrast to previous work, we look at a corpus of natural speech and do not rely on single word retrievals in an experimental setting. This allows us to observe how certain processes involved in fluent language production overlap. In particular, the data support a model according to which lexical retrievals can happen quite early. By using the formalism defined by the empirically-validated ACT-R framework, we show when memory retrievals are taking place through the effect on speaking rates, seeing facilitation early and inhibition later. We conclude that low-activation words can be retrieved as early as 14 words before they are spoken. As low activation words are higher information and require longer to retrieve, this has theoretical implications for some empirical findings of language processing.

Acknowledgements

This project was supported by National Science Foundation projects BCS-1457992 and IIS-1459300. We would like to thank Alex Ororbia, Matthew Kelly, Yang Xu, and Ying Xu for their comments on an earlier version of the paper.

References

John Anderson. 1983. A Spreading Activation Theory of Memory. *Journal of Verbal Learning and Verbal Behavior* 22:261–295.

John R. Anderson. 1991. Cognitive architectures in a rational analysis. In Kurt VanLehn, editor, *Architectures for Intelligence*, Lawrence Erlbaum Associates.

John R. Anderson, Daniel Bothell, Michael D. Byrne, Scott Douglass, Christian Lebiere, and Yulin Quin. 2004. An integrated theory of the mind. *Psychological Review* 111:1036–1060.

Inbal Arnon and Uriel Cohen Priva. 2013. More than words: The effect of multi-word frequency and constituency on phonetic duration. *Language and Speech* 56(3):349–371.

Inbal Arnon and Uriel Cohen Priva. 2014. Time and again: The changing effect of word and multiword frequency on phonetic duration for highly frequent sequences. *The Mental Lexicon* 9(3):377–400.

Alan Bell, Jason M Brenier, Michelle Gregory, Cynthia Girand, and Dan Jurafsky. 2009. Predictability effects on durations of content and function words in conversational english. *Journal of Memory and Language* 60(1):92–111.

J Kathryn Bock and Willem J M Levelt. 2002. *Language production: Grammatical encoding*, Routledge, volume 5, pages 405–452.

Michael D Byrne, John R Anderson, Scott Douglass, and Michael Matessa. 1999. Eye tracking the visual search of click-down menus. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM, pages 402–409.

Alfonso Caramazza. 1997. How many levels of processing are there in lexical access? *Cognitive Neuropsychology* 14(1):177–208.

Alfonso Caramazza. 2006. Lexical access in bilingual speakers: What's the (hard) problem? *Bilingualism: Language and Cognition* 9:153–166.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics* 16(1):22–29.

Jeremy R. Cole, Moojan Ghafurian, and David Reitter. 2017. Linking memory activation and word adoption in social language use via rational analysis. In *Proceedings of the 15th International Conference on Cognitive Modeling*. London, UK.

Jeremy R Cole and David Reitter. 2017. Examining Working Memory during Sentence Construction with an ACT-R Model of Grammatical Encoding. In *Proceedings of the 15th International Conference on Cognitive Modeling*. London, UK.

Allan M Collins and Elizabeth F Loftus. 1975. A spreading-activation theory of semantic processing. *Psychological Review* 82(6):407.

Meredyth Daneman and Patricia A Carpenter. 1980. Individual differences in working memory and reading. *Journal of verbal learning and verbal behavior* 19(4):450–466.

Gary S. Dell. 1986. A spreading-activation theory of retrieval in sentence production. *Psychological Review* 93(3):283.

Gary S. Dell and Padraig G. O'Seaghdha. 1992. Stages of lexical access in language production. *Cognition* 42(1):287–314.

Gary S. Dell, Myrna F. Schwartz, Nadine Martin, Eleanor M. Saffran, and Deborah A. Gagnon. 1997. Lexical access in aphasic and nonaphasic speakers. *Psychological Review* 104(4):801–838.

Fernanda Ferreira and Benjamin Swets. 2002. How incremental is language production? Evidence from the production of utterances requiring the computation of arithmetic sums. *Journal of Memory and Language* 46(1):57–84.

Victor S. Ferreira. 1996. Is it better to give than to donate? Syntactic flexibility in language production. *Journal of Memory and Language* 35:724–755.

Victor S. Ferreira and L. Robert Slevc. 2007. Grammatical encoding. In M. Gareth Gaskell, editor, *The Oxford Handbook of Psycholinguistics*, Oxford University Press, page 453.

Dmitriy Genzel and Eugene Charniak. 2002. Entropy rate constancy in text. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, pages 199–206. https://doi.org/10.3115/1073083.1073117.

John J. Godfrey, Edward C. Holliman, and Jane McDaniel. 1992. SWITCHBOARD: telephone speech corpus for research and development. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-92)*. IEEE, volume 1, pages 517–520.

Markus Guhe. 2007. *Incremental Conceptualization for Language Production*. Lawrence Erlbaum Associates.

T Florian Jaeger. 2010. Redundancy and reduction: Speakers manage syntactic information density. *Cognitive Psychology* 61(1):23–62.

Otto Jespersen. 1992. *The philosophy of grammar*. University of Chicago Press.

Willem JM Levelt. 1992. Accessing words in speech production: Stages, processes and representations. *Cognition* 42(1):1–22.

Willem JM Levelt, Ardi Roelofs, and Antje S Meyer. 1999. A theory of lexical access in speech production. *Behavioral and brain sciences* 22(1):1–38.

Richard L. Lewis and Shravan Vasishth. 2005. An activation-based model of sentence processing as skilled memory retrieval. *Cognitive science* 29(3):375–419.

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

Julian N Marewski and Katja Mehlhorn. 2011. Using the ACT-R architecture to specify 39 quantitative process models of decision making. *Judgment and Decision Making* 6(6):439.

Alexander A. Petrov. 2006. Computationally efficient approximation of the base-level learning equation in ACT-R. In *Proceedings of the seventh international conference on cognitive modeling*. pages 391–392.

Martin J. Pickering and Simon Garrod. 2013. An integrated theory of language production and comprehension. *Behavioral and Brain Sciences* 36(04):329–347.

Peter Pirolli, Wai-tat Fu, Ed Chi, and Ayman Farahat. 2006. Information scent and web navigation: Theory, models and automated usability evaluation. *The Next Wave: NSA's Review of Emerging Technologies* 15(2):5–12.

Brenda Rapp and Matthew Goldrick. 2000. Discreteness and interactivity in spoken word production. *Psychological review* 107(3):460.

Roger Ratcliff and Gail McKoon. 1989. Similarity information versus relational information: Differences in the time course of retrieval. *Cognitive Psychology* 21(2):139–155.

David Reitter, Frank Keller, and Johanna D Moore. 2011. A computational cognitive model of syntactic priming. *Cognitive Science* 35(4):587–637.

Ardi Roelofs, Antje S Meyer, and Willem J M Levelt. 1998. A case for the lemma/lexeme distinction in models of speaking: Comment on caramazza and miozzo (1997). *Cognition* 69(2):219–230.

Vasile Rus, Mihai C Lintean, Rajendra Banjade, Nobal B Niraula, and Dan Stefanescu. 2013. Semilar: The semantic similarity toolkit. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Sofia, Bulgaria, pages 163–168.

Herbert Schriefers, Antje S. Meyer, and Willem J.M. Levelt. 1990. Exploring the time course of lexical access in language production: Picture-word interference studies. *Journal of Memory and Language* 29(1):86–102.

Andreas Stolcke. 2002. SRILM-an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*. Denver, volume 2, pages 901–904.

Shravan Vasishth and Richard L. Lewis. 2004. Modeling sentence processing in ACT-R. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*. Association for Computational Linguistics, pages 82–87.

Jiahong Yuan, Mark Liberman, and Christopher Cieri. 2006. Towards an integrated understanding of speaking rate in conversation. In *Proceedings of the 9th International Conference on Spoken Language Processing*. Pittsburgh, Pennsylvania.

Unsupervised Induction of Linguistic Categories with Records of Reading, Speaking, and Writing

Maria Barrett[1] **Ana V. González-Garduño**[2]
Lea Frermann[3] **Anders Søgaard**[2]
[1]Department of Nordic Studies and Linguistics, University of Copenhagen, Denmark
[2]Department of Computer Science, University of Copenhagen, Denmark
[3]Amazon Development Center, Berlin, Germany
barrett@hum.ku.dk {ana, soegaard}@di.ku.dk
lfrerman@amazon.com

Abstract

When learning POS taggers and syntactic chunkers for low-resource languages, different resources may be available, and often all we have is a small tag dictionary, motivating type-constrained unsupervised induction. Even small dictionaries can improve the performance of unsupervised induction algorithms. This paper shows that performance can be further improved by including data that is readily available or can be easily obtained for most languages, i.e., eye-tracking, speech, or keystroke logs (or any combination thereof). We project information from all these data sources into shared spaces, in which the union of words is represented. For English unsupervised POS induction, the additional information, which is not required at test time, leads to an average error reduction on Ontonotes domains of 1.5% over systems augmented with state-of-the-art word embeddings. On Penn Treebank the best model achieves 5.4% error reduction over a word embeddings baseline. We also achieve significant improvements for syntactic chunk induction. Our analysis shows that improvements are even bigger when the available tag dictionaries are smaller.

1 Introduction

It is a core assumption in linguistics that humans have knowledge of grammar and that they use this knowledge to generate and process language. Reading, writing, and talking leave traces of this knowledge and in psycholinguistics this data is used to analyze our grammatical competencies. Psycholinguists are typically interested in falsifying a specific hypothesis about our grammatical competencies and therefore collect data with this hypothesis in mind. In NLP, we typically require big, representative corpora. NLP usually has induced the models from expensive corpus annotations by professional linguists, but recently, a few researchers have shown that data traces from human processing can be used directly to improve NLP models (Klerke et al., 2016; Barrett et al., 2016; Plank, 2016).

In this paper, we investigate whether unsupervised POS induction and unsupervised syntactic chunking can be improved using human text processing traces. We also explore what traces are beneficial, and how they are best combined. Our work supplements psycholinguistic research by evaluating human data on larger scale than usual, but more robust unsupervised POS induction also contributes to NLP for low-resource languages for which professional annotators are hard to find, and where instead, data from native speakers can be used to augment unsupervised learning.

We explore three different modalities of data reflecting human processing plus standard, pretrained distributional word embeddings for comparison, but also because some modalities might fare better when combined with distributional vectors. Data reflecting human processing come from reading (two different eye-tracking corpora), speaking (prosody), and typing (keystroke logging). We test three different methods of combining the different word representations: a) canonical correlation analysis (CCA) (Faruqui and Dyer, 2014b) and b) singular value decomposision and inverted softmax feature projection (SVD+IS) (Smith et al., 2017) and c) simple concatenation of feature vectors.

Contributions We present experiments in unsupervised POS and syntactic chunk induction using multi-modal word representations, obtained from records of reading, speaking, and writing. Individually, all modalities are known to contain syntactic processing signals, but to the best of our

Lea Frermann carried out this work while at the University of Edinburgh.

Proceedings of NAACL-HLT 2018, pages 2028–2038
New Orleans, Louisiana, June 1 - 6, 2018.

knowledge, we are the first to combine them in one model. Our work extends on previous work in several respects: (a) We compare using data traces from gaze, speech, and keystrokes. (b) We consider three ways of combining such information that do not require access to data from all modalities for all words. (c) While some previous work assumed access to gaze data at test time, our models do not assume access to any modalities at test time. (d) We evaluate how much the additional information helps, depending on the size of the available tag dictionary. (e) While related work on keystrokes and prosody focused on a single feature, all our word representations are multi-dimensional and continuous.

2 Related work

Eye-tracking data reflect the eye movements during reading and provide millisecond-accurate records of the readers fixations. It is well established that the duration of the fixations reflect the processing load of the reader (Rayner, 1998). Words from closed word classes are usually fixated less often and for shorter time than words from open word classes (Rayner and Duffy, 1988). Psycholinguistics, however, is generally not interested in covering all linguistic categories, and psycholinguists typically do not study corpora, but focus instead on small suites of controlled examples in order to explore human cognition. This is in contrast with NLP. Some studies have, however, tried to bridge between psycholinguistics and NLP. Demberg and Keller (2008) found that eye movements reflected syntactic complexity . Barrett and Søgaard (2015a) and Barrett and Søgaard (2015b) have tried to–respectively–predict a full set of syntactic classes and syntactic functions across domains in supervised setups. Barrett et al. (2016), which is the work most similar to ours, used eye-tracking features from the Dundee Corpus (Kennedy et al., 2003), which has been augmented with POS tags by Barrett et al. (2015). They tried for POS induction both on token-level and type-level features. They found that eye-tracking features significantly improved tagging accuracy and that type-level eye-tracking features helped more than token-level. We use the same architecture as Barrett et al. (2016).

Keystroke logs also reflect the processing durations, but of writing. Pauses, burst and revisions in keystroke logs are used to investigate the cognitive process of writing (Matsuhashi, 1981; Baaijen et al., 2012). Immonen and Mäkisalo (2010) found that for English-Finnish translation and monolingual Finnish text production, predicate phrases are often preceded by short pauses, whereas adpositional phrases are more likely to be preceded by long pauses. Pauses preceding noun phrases grow with the length of the phrase. They suggest that the difference is explained by the processing of the predicate begins before the production of the clause starts, whereas noun phrases and adpositional phrases are processed during writing. Pre-word pauses from keystroke logs have been explored with respect to multi-word expressions (Goodkind and Rosenberg, 2015) and have also been used to aid shallow parsing (Plank, 2016) in a multi-task bi-LSTM setup.

Prosodic features provide knowledge about how words are pronounced (tone, duration, voice etc.). Acoustic cues have already been used to improve unsupervised chunking (Pate and Goldwater, 2011) and parsing (Pate and Goldwater, 2013). Pate and Goldwater (2011) cluster the acoustic signal and use cluster label as a discrete feature whereas Pate and Goldwater (2013) use a quantized word duration feature.

Plank (2016) and Goodkind and Rosenberg (2015) also used a single keystroke feature (keystroke pre-word pause) and the former study also discretized the feature. Our work, in contrast, uses acoustic and keystroke features as multi-dimensional, continuous word representations.

3 Modalities

In our experiments, we begin with five sets of word representations: prosody, keystroke, gaze as recorded in the GECO corpus, gaze as recorded in the Dundee corpus, as well as standard, text-based word embeddings from eigenwords. See below for details and references. All modalities except the pre-trained word embeddings reflect human processing of language. For all modalities, we use type-level-averaged features of lower-cased word types.

The choice of using type-averaged features is motivated by Barrett et al. (2016), who tried both token-level and type-averaged eye-tracking features for POS induction and found that type-level gaze features worked better than token-level. Type-averaged features also have the advantage of not relying on access to the auxillary data at test

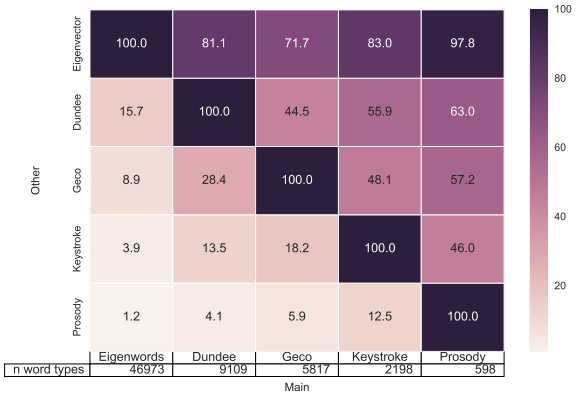


Figure 1: The percentage of overlapping word types for pairs of modalities. Overlapping words are used for projecting word representations into a shared space. Read column-wise. E.g. when combining eigenwords and prosody, only 1.2% of the 46973 eigenvector word types are overlapping (bottom left), and 97.8% of the 598 prosody word types are overlapping (top right).

time. Type-level averages are simply looked up in an embedding file for all previously seen words. On the other hand, type-level features obviously do not represent ambiguities, e.g., *beat* as a verb and a noun separately. All our features, except log-transformed word frequencies were normalized.

We run unsupervised induction experiments for all ($2^5 - 1 = 31$) combinations of our five data sources on the development sets to determine which data types contribute to the task. We consider three different ways of combining modalities, two of which learn a projection into a shared space using word overlap as supervision, and one simply concatenates the embedding spaces. The combination methods are further described in §4.

We list the number of word types per modality and percentage of pair-wise overlapping words in Figure 1. We only use existing data from native speaking participants, for reproducibility and in order not to get learner effects ie. biases introduced by non-native speakers. §3.2-3.5 describe each modality in detail, and how we compute the word representations. §3.1 describes a set of basic features used in all of our experiments.

3.1 Basic features

Like Li et al. (2012), we append a small set of basic features to all our feature sets: features relating to orthography such as capitalization, digits and suffix. Furthermore we append log word frequency and word length. Word frequencies per million are

Modality	n found pairs	Weigh. av. cor.
Prosody	31	0.369
Keystroke	1082	0.060
GECO	2449	-0.030
Dundee	4066	-0.035
Eigenwords	9828	0.197

Table 1: Results on word association norms from wordvectors.org Correlation weighted by number of found pairs per word embedding type.

obtained from British National corpus (BNC) frequency lists (Kilgarriff, 1995). Word length and word frequency explain around 70% of the variance in the eye movement (Carpenter and Just, 1983) and are therefore also important for estimating the impact of gaze features beyond such information. Plank (2016) used keystroke features for shallow chunking and did not find any benefit of normalizing word length by pre-word pause before typing each word, but Goodkind and Rosenberg (2015) did find a strong logarithmic relationship between word length and pre-word pause as well as between word frequency and pre-word pause.

3.2 Dundee and GECO eye-tracking corpora

We use two different eye-tracking corpora. The GECO corpus (Cop et al., 2017) and the Dundee Corpus (Kennedy et al., 2003) are the two largest eye movement corpora with respect to word count. We use the native English part of the GECO corpus and the English part of the Dundee Corpus. The GECO corpus is publicly available[1] and the Dundee Corpus is available for research purposes.

Participants and data The Dundee Corpus is described in Kennedy and Pynte (2005). The Dundee Corpus consists of the eye movements of 10 readers as they read the same 20 newspaper articles. For GECO, all 14 participants in the native English part read a full Agatha Christie novel. Both corpora contain > 50.000 words per reader. All participants for both corpora are adult, native speakers of English and skilled readers.

Self-paced reading Both eye-tracking corpora reflect natural reading by making the reading self-paced and using naturally-occurring, contextualized text.

[1] http://expsy.ugent.be/downloads/geco/

Features Eye movements–like most features reflecting human processing–are very susceptible to experiment-specific effects e.g. instructions and order effects such as fatigue. Furthermore, the GECO corpus has a slightly different eye movement feature set than what we have for the Dundee corpus. Therefore we treat the two eye movement corpora as two individual modalities in order to assess their individual contributions. GECO has 34 features reflecting word-based processing. Dundee has 30 word-based features that were extracted from the raw data and previously used for POS induction by Barrett et al. (2016). For GECO, we use the features that are already extracted by the authors of the corpus. Both corpora include five word-based features e.g., first fixation duration (which is a measure said to reflect early syntactic and semantic integration), total fixation time and fixation probability. The Dundee Corpus has more features concerning the context words whereas GECO has pupil size and many features distinguishing the different passes over a word.

3.3 Prosody

The prosody features are described in detail in Frermann and Frank (2017) and are freely available.[2] They are derived from the Brent (Brent and Siskind, 2001) and Providence (Demuth et al., 2006) portions of the CHILDES corpus (MacWhinney, 2000), comprising longitudinal datasets of raw speech directed to 22 children, and its transcription. Word-level speech-text alignments were obtained automatically using forced alignment. For each token-level audio snippet, a set of 88 prosody features was extracted based on a previously established feature set (Eyben et al., 2016), including standard features derived from F0–F3 formants, spectral shape and rhythm features, intensity and MFCC features among others. Type-level prosody features were obtained as averaged token-level features for each word type.

3.4 Keystroke features

We extracted keystroke features from the publicly available data from Killourhy and Maxion (2012). This data contains key hold times and pauses of all key presses of 20 subjects as they completed transcription and free composition tasks. We only used data from the free composition part. A pause is defined by the authors as the duration from keydown to keydown. The free composition data consists of a total of 14890 typed words and 2198 word types.

For each word, we extracted the following features: (i) average key hold duration of all characters associated with producing the word, (ii) pre-word pause, (iii) hold duration of space key before word, (iv) pause length of space key press pause before word, and (v) ratio of keypresses used in the word production to length of the final word. For each word, we also included these five features for up to 3 words before. In total, we have $5 * 4 = 20$ keystroke features. We use lower-cased word type averages, as with the other modalities.

3.5 Eigenwords

Eigenwords are standard, pre-trained word embeddings, induced using spectral-learning techniques (Dhillon et al., 2015). We used the 30-dimensional, pre-trained eigenvectors.[3]

3.6 Preliminary evaluation

Our application of these word representations and their combinations is unsupervised POS and syntactic chunk induction, but before presenting our projection methods in §4 and our experiments in §5, we present a preliminary evaluation of the different modalities using word association norms.

Table 1 shows the weighted correlation between cosine distances in the representations and the human ratings in the word association norm datasets available at `wordvectors.org` (Faruqui and Dyer, 2014a). Eigenwords, not surprisingly, correlates better than the representation based on processing data – with the exception of prosody. The correlation with prosody is non-significant, however, because of the small sample size.

4 Combining datasets

We now have word representations from different, complementary modalities, with very different coverages, but all including a small overlap. We assume that the different modalities contain complementary human text processing traces because they reflect different cognitive processes, which motivates us to combine these different sources of information. Our assumption is confirmed in the evaluation. The fact that we have very low coverage for some modalities, and the

[2] https://github.com/ColiLea/prosodyAOA

[3] http://www.cis.upenn.edu/~ungar/eigenwords/

fact that we have an overlap between all our vocabularies, specifically motivates an approach, in which we use the intersection of word types to learn a projection from two or more of these modalities into a shared space. Obviously, we can also simply concatenate our representations, but because of the low coverage of some modalities and because co-projecting modalities has some regularization effect, we hypothesize that it is better to learn a projection into a shared space. This hypothesis is verified by the results in §6.

4.1 Concatenating modalities

The simplest way of combining the modalities is concatenating the corresponding vectors for each word. The different modalities have different dimensionalities, so we would need to perform dimensionality reduction to sum or average vectors, and the non-overlapping words don't allow for e.g. taking the outer product, so we simply concatenate the vectors instead. We use 0 for missing values.

4.2 CCA

§4.2 and §4.3 describe two different projection methods for projecting the representations in the different modalities into a shared space. We use the intersection of the lower-cased vocabulary for the alignment, i.e., as a supervision signal. For example, if the words *man*, *dog* and *speak* exist in both eigenword and keystroke data, from these 2 x 3 vectors, CCA estimate the transformation for the vectors for *house*, *cat* and *boy*, which (in this example) only exists in the keystroke data.

Canonical Correlation Analysis (CCA), as originally proposed by Hotelling (1936), is a method of finding the optimum linear combination between two sets of variables, so the set of variables are transformed onto a projected space while the correlation is maximized. We use the implementation of Faruqui and Dyer (2014b) made for creating bilingual embeddings. We use modalities instead of languages. The size of the projected space is smaller than or equal to the original dimension.

We incrementally combine modalities and project them to new, shared spaces using the intersection of the lower-cased vocabulary. We add them by the order of word type count starting with the modality with most word types. For the first projection only, we reduce the size of the projected space. We set the ratio of the first projected space (only two modalities) to 0.6 based on POS induction results on development data using the setup described in §5.

4.3 SVD and Inverted Softmax

As an alternative to CCA, but closely related, we also use a projection method proposed and implemented by Smith et al. (2017), which uses singular value decomposition and inverted softmax (SVD+IS). This method uses a reference space, rather than projecting all modalities into a new space. Smith et al. (2017) apply SVD+IS to obtain an orthogonal transformation matrix that maps the source language into the target language. In addition, in order to estimate their confidence on the predicted target, they use an inverted softmax function for determining the probability that a target word translates back into a source word.

Like for CCA, we incrementally project datasets onto each other starting with the most word-type rich modality. We use the highest dimensionality of any of our representations (88 dimensions).

5 Experiments

This section presents our POS and syntactic chunk induction experiments. We present the datasets we used in our experiments, the sequence tagging architecture, based on second-order hidden Markov models, as well as the dictionary we used to constrain inference at training and test time.

5.1 Data

For unsupervised POS induction, we use Ontonotes 5.0 (Weischedel et al., 2013) for training, development and test. We set all hyper-parameters on the newswire (NW) domain, optimizing performance on the development set. Size of the development set is 154,146 tokens. We run individual experiments on each of the seven domains, with these hyper-parameters, reporting performance on the relevant test set. The domains are broadcast conversation (BC), broadcast news (BN), magazines (MZ), newswire (NW), the Bible (PT), telephone conversations (TC), and weblogs (WB). We also train and test unsupervised POS induction on the CoNLL 2007 (Nivre et al., 2007) splits of the Penn Treebank (Marcus et al., 1993) using the hyper-parameter settings from Ontonotes. We mapped all POS labels to Google's coarse-grained, universal POS tagset (Petrov et al., 2012). For model selection, we select based both on best results on Ontonotes

Rules		
DET	→	NP
VERB	→	VP
NOUN|PRONOUN|NUM	→	NP
.	→	O
ADJ	→	NP|ADJP
ADV	→	NP|VP|ADVP|AD
PRT	→	NP|PRT
CONJ	→	O|NP
ADP	→	PP|VP|SBAR

Table 2: Heuristics for expanding our POS dictionary to chunks

NW development as well as Penn Treebank development sets.

For syntactic chunk induction, we use the bracketing data from Penn Treebank with the standard splits for syntactic chunking. We tune hyperparameters for chunking on the development set and select best models based on the development result.

5.2 Model

We used a modification of the implementation of a type-constrained, second-order hidden Markov model with maximum entropy emissions from Li et al. (2012) (SHMM-ME). It is a second-order version of the first order maximum entropy HMM presented in (Berg-Kirkpatrick et al., 2010) with the important addition that it is constrained by a crowd-sourced tag dictionary (Wiktionary). This means that for all words in the Wiktionary, the model is only allowed to predict one of the tags listed for it in Wiktionary

The same model was used in Barrett et al. (2016) to improve unsupervised POS inducing using gaze data from the Dundee Corpus, and in Bingel et al. (2016) to augment an unsupervised POS tagger with features from fMRI recordings.

The number of EM iterations used for inducing our taggers was tuned using eigenvector embeddings on the development data, considering values 1..50. PoS performance peaked at iterations 30 and 31. We use 30 in all our POS experiments. For syntactic chunking, we use 48 iterations, which led to the best performance on the PTB development data using only eigenword embeddings.

5.3 Wiktionary

The Wiktionary constrains the predicted tags in our model. The better the Wiktionary, the better the predictions.

For POS-tagging we used the same Wiktionary dump[4] that Li et al. (2012) used in their original experiments. The Wiktionary dump associated word types with Google's universal parts-of-speech labels.

For chunking, Wiktionary does not provide direct information about the possible labels of words. We instead apply simple heuristics to relate POS information to syntactic chunking labels. Since we already know the relation between words and POS labels from Wiktionary, we can compute the transitive closure in order to obtain a dictionary relating words with syntactic chunking labels. We present the heuristics in Table 2.

Note that the rules are rather simple. We do not claim this is the best possible mapping. We are relying on these simple heuristics only to show that it is possible to learn syntactic chunkers in an unsupervised fashion by relying on a combination of features from different modalities and a standard, crowd-sourced dictionary.

Feature set	TA
No embeddings	60.32
Eigenwords	59.26
Best combined models	
CCA Dun_GECO_Pros	**63.33***†
SVD+IS GECO_Key_Pros	62.91*
Concat Eig_GECO_Key	61.16

Table 3: Chunk tagging accuracy. Best models from CCA, SVD+IS and concatenation. Model section on development set. * $p < .001$ Mcnemar mid-p test when comparing to no embeddings. † $p < .001$ Mcnemar mid-p test when comparing to Eigenwords.)

6 Results

All our POS tagging accuracies can be seen in Table 4. Our first observation is that human processing data helps unsupervised POS induction. In fact, the models augmented with processing data are *consistently* better than the baseline without vector representations, as well as better than only using distributional word embeddings.

Generally, CCA seems to find the best projection into a common space for system combinations. For Penn Treebank, the CCA-aligned model is the best and this result is significant ($p <$

[4]https://code.google.com/archive/p/wikily-supervised-pos-tagger/

Feature set	Ontonotes BC	BN	MZ	NW	PT	TC	WB	avg	PTB
No embeddings	83.1	84.41	85.32	84.94	85.14	77.8	85.93	83.81	82.83
Eigenwords	83.16	84.68*	85.48	85.07	85.31	78.07	85.88	83.95	83.38*
Best Ontonotes NW models									
CCA Eig_Dun	**83.45***†	**84.99***	85.79*	**85.38***†	85.2	77.99	**86.38***†	84.17	**84.28***†
SVD+IS Dun_GECO_Key	83.24	84.76	**86.22***†	85.33*†	85.44	77.84	85.95	84.11	84.25*†
Concat Eig_Dun_GECO	83.39*†	84.78*	85.8*†	85.36*†	**85.45**	**78.38***	86.21†	**84.19**	83.91*†
Best PTB models									
CCA Eig_Dun	**83.45***†	**84.99***	85.79*	**85.38***†	85.2	77.99	**86.38***†	84.17	**84.28***†
SVD+IS Dun_Key	83.24	84.59	86.12*†	85.28*†	85.39	77.90	85.86	84.05	84.24*†
Concat Eig_Pros	83.22	84.54	85.67	85.01	84.98	77.98	85.97	83.91	84.22*†

Table 4: POS tagging accuracies for baselines and the model combinations that performed best on newswire development data (NW). Best performance per domain is boldfaced. *) $p < .001$ McNemar mid-p test when compared to the no embeddings condition for the corresponding test set. †) $p < .001$ McNemar mid-p test when compared to eigenwords for the corresponding test set.

.001) when comparing both to no embeddings and eigenwords. For Ontonotes 5.0, CCA is better than the other projection methods in 4/7 domains, but when averaging, concatenation gets the higher result.

The standard embeddings are often part of the best combinations, but the human processing data contributes with important information; in 4/7 domains as well as on PTB data, we see a significantly better performance ($p < .001$) with a combination of modalities when comparing to eigenwords.

Aligning Dundee with eigenwords is the best POS model both according to the Ontonotes 5.0 NW development set and the Penn Treebank development set. Dundee is the most frequent modality in the six best POS induction models with five appearances. Eigenwords is second most frequent with four appearances.

The syntactic chunking accuracies are in Table 3. Also here CCA is the better combination method. For chunking, all combined models are better than no embeddings and eigenwords. The improvement is significant compared to no embeddings for concatenation $p < .001$. For CCA, the result is significantly better than no embeddings and eigenwords.

For chunking, GECO data appears in all best models and is thus the most frequent modalities. Keystroke and prosody appears in two best models each.

	Keystroke	Dundee	GECO
Dundee	16.84		
GECO	11.39	1.02	
CCA all	13.98	3.72	3.09

Table 5: Graph similarities in $[0, \infty)$, $0 =$ identical.

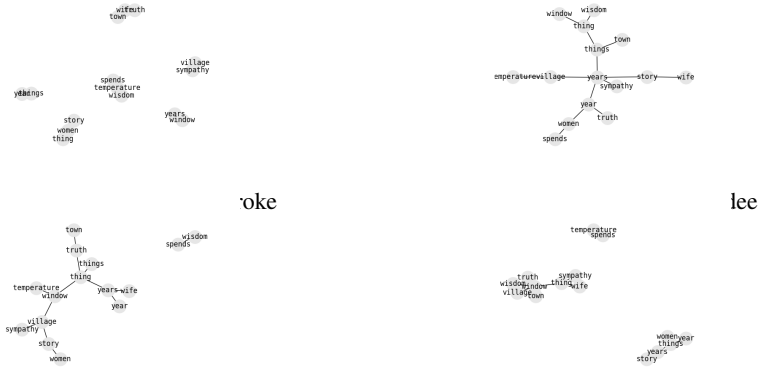
7 Analysis

7.1 What is in the vectors?

Nearest neighbor graphs We include a detailed analysis of subgraphs of the nearest neighbor graphs in the embedding spaces of keystrokes, Dundee, GECO, and CCA projection of all modalities. Specifically, we consider the nearest neighbor graphs among the 15 most frequent unambigous nouns, according to Wiktionary.[5] See Figure 2 for plots of the nearest neighbor graphs. The prosody features containing less than 600 word types only contained 2 of the 15 nouns and is therefore not included in this analysis.

Projecting word representations into a shared space using linear methods assumes approximate isomorphism between the embedding spaces - or at least their nearest neighbor graphs. We use the VF2 algorithm (Cordella et al., 2001) to verify that the subgraphs are *not* isomorphic, but this can also be seen directly from Figure 2. Neither keystroke and gaze embeddings, nor the two different gaze-induced embeddings are isomorphic.

[5] Wiktionary is a crowd-sourced, imperfect dictionary, and one of the "unambiguous nouns" is *spends*, which, we assume, you are more likely to encounter as a verb.



(c) Geco (d) CCA projection, all modalities

Figure 2: Nearest neighbor graphs for 15 frequent nouns.

Since none of the modalities induce isomorphic nearest neighbor graphs, this does not tell us much about similarities between modalities. To quantify the similarity of non-isomorphic graphs, we use *eigenvector similarity* Shigehalli and Shettar (2011), which we calculate by computing the Laplacian eigenvalues for the nearest neighbors, and for each graph, find the smallest k such that the sum of the k largest eigenvalues is $<90\%$ of the eigenvalues. We then take the smallest k of the two, and use the sum of the squared differences between the largest k eigenvalues as our similarity metric.

Using this metric to quantify graph similarity, we see in Table 5 that, not surprisingly, the gaze graphs are the most similar. The projected space is more similar to the gaze spaces, but balances gaze and keystroke information. The GECO embeddings agree more with the keystrokes than the Dundee embeddings does.

t-SNE plots We take words that–according to the Wiktionary–can only have one tag and sort them by BNC frequency (Kilgarriff, 1995) in descending order. For these words and their POS tags we get the feature vector of the POS model yielding the highest result on both Ontonotes and PTB: CCA-projected eigenwords and Dundee features. For the first 200 occurrences of the frequency-sorted list, we reduce dimensionality using t-Distributed Stochastic Neighbor Embedding (t-SNE) (Maaten and Hinton, 2008) and plot the result. Figure 3 shows that 200 most frequent content words cluster with respect to their POS tag, somewhat distinguishing verbs from nouns and adjectives from adverbs in CCA space.

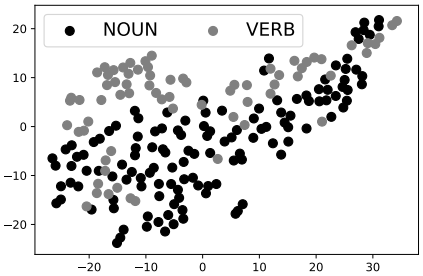
7.2 How big a Wiktionary do we need?

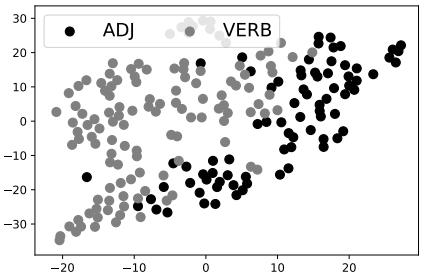
Our Wiktionary for English contains POS information for 72,817 word types. Word types have 6.2 possible POS categories on average meaning we have over 450.000 entries in our POS dictionary. For Penn Treebank, 70.0% of wordtypes of the test set are covered by the dictionary. For the chunking data, 70.4% of wordtypes of the test set are covered by the dictionary. The English Wiktionary is thus much bigger than wiktionaries for low-resource language (Garrette and Baldridge, 2013). How big a dictionary is needed to achieve good performance, and can we get away with a smaller dictionary if we have processing data? This section explores the performance of the model as a function of the Wiktionary size.

We sorted the Wiktionary by word frequency obtained from BNC (Kilgarriff, 1995) and increased the Wiktionary size for the best POS system starting with 0 (no dictionary). For each Wiktionary size, we compare with the baseline without access to processing data and eigenwords. The learning curve can be seen in Figure 4a and Figure 4b. We observe that having entries for the most frequent words is a lot better than having no dictionary, and that the difference between our best system and the baseline exists across all dictionary sizes. With 10,000 entries, all systems seems to reach a plateau.

8 Discussion

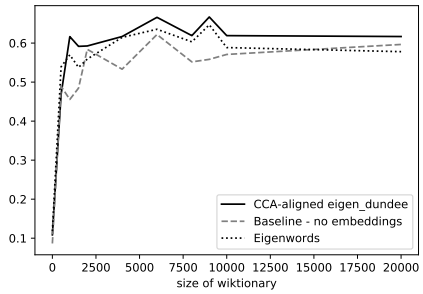
Genres and domains When collecting our human language processing data, we did not control for genre. Our data sets span child-directed speech, free text composition, and skilled adults reading fiction and newspaper articles. The

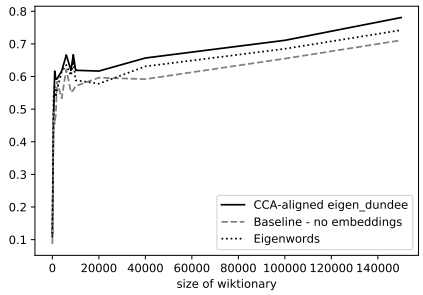




(b) ADJ and VERB

Figure 3: t-SNE plots of CCA-projected eigen_dundee features for pairs of tags.





(b) 0-150,000 entries

Figure 4: Learning curve assuming Wiktionary entries for k most frequent words, comparing our best PoS induction system against our baseline. On Ontonotes WB development data, 30 training iterations.

Dundee corpus (newspaper articles) matches the genre of at least some of the Ontonotes test set. Immonen and Mäkisalo (2010) found that for keystroke, genre does seem to have an effect on average pause length, be it sentence initial, word initial, clause initial or phrase initial. Texts organized linearly–e.g. reports and narratives–require less pausing than texts with a global approach, like expository, persuading and generalizing text. Our results show that human processing features transfer across genres, but within-genre data would probably be beneficial for results.

Richer representations The type-level features we use, do not take context into account, and the datasets we use, are too small to enrich our representations. Human processing data is more and more readily available, however. Eye trackers are probably built into the next generation of consumer hardware, and speech records and keystroke logs are recordable with existing technology.

9 Conclusion

We have shown how to improve unsupervised POS induction and syntactic chunking significantly using data reflecting human language processing. Our model, which is a second-order hidden Markov model, is the first to combine multidimensional, continuous features of eye movements, prosody and keystroke logs. We have shown that these features can be combined using projection techniques, even when they only partially overlap in word coverage. None of our models require access to these features at test time. We experimented with all combinations of modalities, and our results indicate that eye tracking is useful for both chunking and POS induction. Finally, we have shown that the potential impact of human processing data also applies in a low-resource setting, i.e., when available tag dictionaries are small.

Acknowledgements

Thanks to Desmond Elliott for valuable ideas.

This research was partially funded by the ERC Starting Grant LOWLANDS No. 313695, as well as by Trygfonden.

References

Veerle M Baaijen, David Galbraith, and Kees de Glopper. 2012. Keystroke analysis: Reflections on pro-

cedures and measures. *Written Communication* 29(3):246–277.

Maria Barrett, Željko Agić, and Anders Søgaard. 2015. The Dundee treebank. In *The 14th International Workshop on Treebanks and Linguistic Theories (TLT 14)*. pages 242–248.

Maria Barrett, Joachim Bingel, Frank Keller, and Anders Søgaard. 2016. Weakly supervised part-of-speech tagging using eye-tracking data. In *ACL*. pages 579–584.

Maria Barrett and Anders Søgaard. 2015a. Reading behavior predicts syntactic categories. *CoNLL 2015* pages 345–349.

Maria Barrett and Anders Søgaard. 2015b. Using reading behavior to predict grammatical functions. In *Workshop on Cognitive Aspects of Computational Language Learning (CogACLL)*. pages 1–5.

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Cote, John DeNero, , and Dan Klein. 2010. Painless unsupervised learning with features. In *NAACL*. pages 582–590.

Joachim Bingel, Maria Barrett, and Anders Søgaard. 2016. Extracting token-level signals of syntactic processing from fmri-with an application to pos induction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 747–755.

M. R. Brent and J. M Siskind. 2001. The role of exposure to isolated words in early vocabulary development. *Cognition* 81:31–44.

Patricia A Carpenter and Marcel Adam Just. 1983. What your eyes do while your mind is reading. *Eye movements in reading: Perceptual and language processes* pages 275–307.

Uschi Cop, Nicolas Dirix, Denis Drieghe, and Wouter Duyck. 2017. Presenting geco: An eyetracking corpus of monolingual and bilingual sentence reading. *Behavior research methods* 49(2):602–615.

L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. 2001. An Improved Algorithm for Matching Large Graphs. *Proc. of the 3rd IAPR TC-15 Workshop on Graphbased Representations in Pattern Recognition* 17:1–35.

Vera Demberg and Frank Keller. 2008. Data from eye-tracking corpora as evidence for theories of syntactic processing complexity. *Cognition* 109(2):193–210.

Katherine Demuth, Jennifer Culbertson, and Jennifer Alter. 2006. Word-minimality, epenthesis and coda licensing in the early acquisition of english. *Language and Speech* 49(2):137–173.

Paramveer Dhillon, Dean Foster, and Lyle Ungar. 2015. Eigenwords: Spectral word embeddings. *Journal of Machine Learning Research* 16:3035–3078.

Florian Eyben, Klaus Scherer, Bjrn Schuller, Johan Sundberg, Elisabeth Andr, Carlos Busso, Laurence Devillers, Julien Epps, Petri Laukka, Shrikanth Narayanan, and Khiet Phuong Truong. 2016. The Geneva minimalistic acoustic parameter set (GeMAPS) for voice research and affective computing 7(2):190–202.

Manaal Faruqui and Chris Dyer. 2014a. Community evaluation and exchange of word vectors at wordvectors.org. In *ACL: System Demonstrations*. pages 19–24.

Manaal Faruqui and Chris Dyer. 2014b. Improving vector space word representations using multilingual correlation. In *EACL*. pages 462–471.

Lea Frermann and Michael C. Frank. 2017. Prosodic features from large corpora of child-directed speech as predictors of the age of acquisition of words. *CoRR* .

Dan Garrette and Jason Baldridge. 2013. Learning a part-of-speech tagger from two hours of annotation. In *NAACL*.

Adam Goodkind and Andrew Rosenberg. 2015. Muddying the multiword expression waters: How cognitive demand affects multiword expression production. In *MWE@ NAACL-HLT*. pages 87–95.

Sini Immonen and Jukka Mäkisalo. 2010. Pauses reflecting the processing of syntactic units in monolingual text production and translation. *HERMES-Journal of Language and Communication in Business* 23(44):45–61.

Alan Kennedy, Robin Hill, and Joël Pynte. 2003. The Dundee Corpus. In *Poster presented at ECEM12: 12th European Conference on Eye Movements*.

Alan Kennedy and Joël Pynte. 2005. Parafoveal-on-foveal effects in normal reading. *Vision research* 45(2):153–168.

Adam Kilgarriff. 1995. BNC database and word frequency lists. *Retrieved Dec. 2017* .

Kevin S Killourhy and Roy A Maxion. 2012. Free vs. transcribed text for keystroke-dynamics evaluations. In *Proceedings of the 2012 Workshop on Learning from Authoritative Security Experiment Results*. ACM, pages 1–8.

Sigrid Klerke, Yoav Goldberg, and Anders Søgaard. 2016. Improving sentence compression by learning to predict gaze. *NAACL* pages 1528–1533.

Shen Li, João Graça, and Ben Taskar. 2012. Wikily supervised part-of-speech tagging. In *EMNLP*. pages 1389–1398.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research* 9(Nov):2579–2605.

Brian MacWhinney. 2000. *The CHILDES project: Tools for analyzing talk.*. Lawrence Erlbaum Associates, Hillsdale, NJ, USA, third edition edition.

Mitchell Marcus, Mary Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics* 19(2):313–330.

Ann Matsuhashi. 1981. Pausing and planning: The tempo of written discourse production. *Research in the Teaching of English* pages 113–134.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser. *Natural Language Engineering* 13(2):95–135.

John K Pate and Sharon Goldwater. 2011. Unsupervised syntactic chunking with acoustic cues: computational models for prosodic bootstrapping. In *Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics*. Association for Computational Linguistics, pages 20–29.

John K Pate and Sharon Goldwater. 2013. Unsupervised dependency parsing with acoustic cues. *Transactions of the Association for Computational Linguistics* 1:63–74.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *LREC*. pages 2089–2094.

Barbara Plank. 2016. Keystroke dynamics as signal for shallow syntactic parsing. In *COLING*. pages 609–618.

Keith Rayner. 1998. Eye movements in reading and information processing: 20 years of research. *Psychological bulletin* 124(3):372–422.

Keith Rayner and Susan A Duffy. 1988. On-line comprehension processes and eye movements in reading. *Reading research: Advances in theory and practice* 6:13–66.

Vijayalaxmi Shigehalli and Vidya Shettar. 2011. Spectral Technique using Normalized Adjacency Matrices for Graph Matching. *International Journal of Computational Science and Mathematics* 3:371–378.

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium* .

Challenging Reading Comprehension on Daily Conversation: Passage Completion on Multiparty Dialog

Kaixin Ma, Tomasz Jurczyk, Jinho D. Choi
Math and Computer Science
Emory University
Atlanta, GA 30322, USA
{kaixin.ma, tomasz.jurczyk, jinho.choi}@emory.edu

Abstract

This paper presents a new corpus and a robust deep learning architecture for a task in reading comprehension, passage completion, on multiparty dialog. Given a dialog in text and a passage containing factual descriptions about the dialog where mentions of the characters are replaced by blanks, the task is to fill the blanks with the most appropriate character names that reflect the contexts in the dialog. Since there is no dataset that challenges the task of passage completion in this genre, we create a corpus by selecting transcripts from a TV show that comprise 1,681 dialogs, generating passages for each dialog through crowdsourcing, and annotating mentions of characters in both the dialog and the passages. Given this dataset, we build a deep neural model that integrates rich feature extraction from convolutional neural networks into sequence modeling in recurrent neural networks, optimized by utterance and dialog level attentions. Our model outperforms the previous state-of-the-art model on this task in a different genre using bidirectional LSTM, showing a 13.0+% improvement for longer dialogs. Our analysis shows the effectiveness of the attention mechanisms and suggests a direction to machine comprehension on multiparty dialog.

1 Introduction

Reading comprehension that challenges machine's ability to understand a document through question answering has gained lots of interests. Most of the previous works for reading comprehension have focused on either children's stories (Richardson et al., 2013; Hill et al., 2016) or newswire (Hermann et al., 2015; Onishi et al., 2016). Few approaches have attempted comprehension on small talks, although they are evaluated on toy examples not suitable to project real-life performance (Weston et al., 2015). It is apparent that the main stream of reading comprehension has not been on the genre of multiparty dialog although it is the most common and natural means of human communication. The volume of data accumulating from group chat or messaging continues to outpace data accumulation from other writing sources. The combination of available and rapidly developing analytic options, a marked need for dialogue processing, and the disproportionate generation of data from conversations through text platforms inspires us to create a corpus consisting of multiparty dialogs and develop learning models that make robust inference on their contexts.

Passage completion is a popular method of evaluating reading comprehension that is adapted by several standardized tests (e.g., SAT, TOEFL, GRE). Given a document and a passage containing factual descriptions about the contexts in the document, the task replaces keywords in the passage with blanks and asks the reader to fill in the blanks. This task is particularly challenging when the document is in a form of dialog because it needs to match contexts between colloquial (dialog) and formal (passage) writings. Moreover, a context that can be described in a short passage, say a sentence, tends to be expressed across multiple utterances in dialog, which requires discourse-level processing to make the full interpretation of the context.

This paper introduces a new corpus for passage completion on multiparty dialog (Section 3), and a deep learning architecture that produces robust results for understanding dialog contexts (Section 4). Our experiments show that models trained by this architecture significantly outperform the previous state-of-the-art model using bidirectional LSTM, especially on longer dialogs (Section 5). Our analysis highlights the comprehension of our models for matching utterances in dialogs to words in passages (Section 6). To the best of our knowledge, this is the first time that the sentence completion task is thoroughly examined with a challenging dataset on multiparty dialog using deep learning models.

Proceedings of NAACL-HLT 2018, pages 2039–2048
New Orleans, Louisiana, June 1 - 6, 2018.

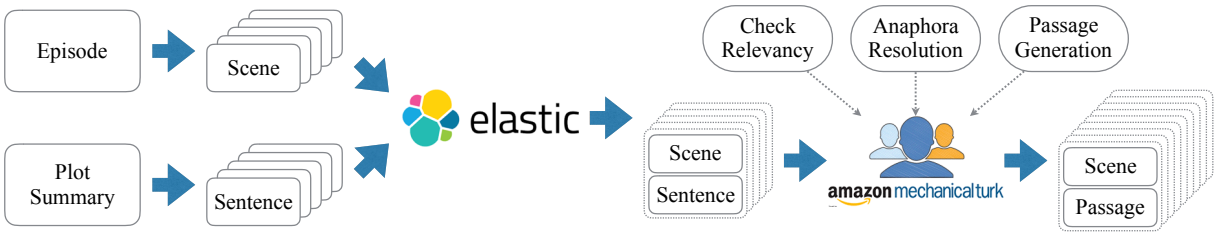


Figure 1: The overview of passage generation. Each episode is split into scenes, and each summary is segmented to sentences. Elasticsearch passes the scene-sentence pairs to crowd workers who are asked to check the relevancy, replace all pronouns with the corresponding names, and generate new passages for the scenes (Section 3.1).

2 Related Work

Hermann et al. (2015) introduced the CNN/Daily Mail dataset where documents and passages were news articles and their summaries respectively, and evaluated neural models with three types of readers. Chen et al. (2016) proposed the entity centric model and the bidirectional LSTM model using attention, and conducted a thorough analysis on this dataset. Trischler et al. (2016) presented the EpiReader that combined a reasoner with an extractor for encoding documents and passages using both CNN and RNN. Dhingra et al. (2017) proposed the gated-attention reader that incorporated attention on multiplicative interactions between documents and passages. At last, Cui et al. (2017) introduced the attention-over-attention reader that placed document-to-passage attention over passage-to-document attention.

Hill et al. (2016) released the Children Book Test dataset where documents were children's book stories and passages were excerpts from those stories. Paperno et al. (2016) introduced the LAMBADA dataset comprising novels from the Book corpus. Onishi et al. (2016) introduced the Who-did-What dataset consisting of articles from the LDC English Gigaword newswire corpus. All corpora described above provide queries, that are passages where certain words are masked by blanks, for the evaluation of passage completion. More datasets are available for another type of a reading comprehension task, that is multiple choice question answering, such as MCTest (Richardson et al., 2013), TriviaQA (Joshi et al., 2017), RACE (Lai et al., 2017), and SQuAD (Rajpurkar et al., 2016).

Unlike the other corpora where documents and passages are written in a similar writing style, they are multiparty dialogs and plot summaries in our corpus, which have very different writing styles. This raises another level of difficulty to match contexts between documents and queries for the task of passage completion.

3 Corpus

The Character Mining project provides transcripts of the TV show *Friends* for ten seasons in the JSON format.[1] Each season contains ≈24 episodes, each episode is split into ≈13 scenes, where each scene comprises a sequence of ≈21 utterances. Chen et al. (2017) annotated the first two seasons of the show for an entity linking task, where personal mentions (e.g., *she*, *mom*, *Rachel*) were identified by their corresponding characters. Jurczyk and Choi (2017) collected plot summaries of all episodes for the first eight seasons to evaluate a document retrieval task that returned a ranked list of relevant documents given any sentence in the plot summaries.

For the creation of our corpus, we collect more plot summaries for the last two seasons of *Friends* from the fan sites suggested by Jurczyk and Choi (2017), generate passages for each dialog using the plot summaries and crowdsourced descriptions (Section 3.1), then annotate mentions of all characters in both the dialogs and the passages for passage completion (Section 3.2).

3.1 Passage Generation

An episode consists of multiple scenes, which may or may not be coherent. In our corpus, each scene is considered a separate dialog. The lengths of the scenes vary from 1 to 256 utterances; we select only scenes whose lengths are between 5 and 25 utterances as suggested by the previous works (Chen and Choi, 2016; Jurczyk and Choi, 2017), which notably improves the readability for crowd workers, resulting higher quality annotation.

The plot summaries collected from the fan sites are associated with episodes, not scenes. To break down the episode-level summaries into scene-level, they are segmented into sentences by the tokenizer in NLP4J.[2] Each sentence in the plot summaries

[1] nlp.mathcs.emory.edu/character-mining
[2] https://github.com/emorynlp/nlp4j

(a) A dialog from *Friends*: Season 8, Episode 12, Scene 2.

ID	Speaker	Utterance
1	-	[Scene: Central Perk, **@ent01** and **@ent02** are there as **@ent03** enters.]
2	**@ent03**	Hey! Oh, I'm so glad you guys are here. I've been dying to tell someone what happened in the Paleontology department today.
3	**@ent01**	(To **@ent02**) Do you think he saw us or can we still sneak out?
4	**@ent03**	Professor **@ent04**, the head of the department, so ...
5	**@ent02**	They made you head of the department!
6	**@ent03**	No, I get to teach one of his advanced classes! Why didn't I get head of the department?
7	**@ent01**	Oh! Hey **@ent02**, listen umm ...
8	**@ent02**	Yeah.
9	**@ent01**	I got a big date coming up, do you know a good restaurant?
10	**@ent02**	Uh, **@ent05**'s Cafe. They got great food and it's really romantic.
11	**@ent01**	Ooh, great! Thanks!
12	**@ent02**	Yeah! Oh, and then afterwards you can take her to the Four Seasons for drinks. Or you go downtown and listen to some jazz. Or dancing - Oh! Take her dancing!
13	**@ent01**	You sure are naming a lot of ways to postpone xxx, I'll tell ya ...
14	**@ent02**	Ooh, I miss dating. Gettin' all dressed up and going to a fancy restaurant. I'm not gonna be able to do that for so long, and it's so much fun! I mean not that sitting at home worrying about giving birth to a sixteen pound baby is not fun.
15	**@ent01**	Hey, y'know what?
16	**@ent02**	Huh?
17	**@ent01**	Why don't I take you out?
18	**@ent02**	What?! **@ent01**, you don't want to go on a date with a pregnant lady.
19	**@ent01**	Yes I do! And we're gonna go out, we're gonna have a good time, and take your mind off of childbirth and c-sections and-and giant baby heads stretching out ...
20	**@ent02**	(interrupting) Okay! I'll go with ya! I'll go! I'll go with ya.
21	**@ent01**	I'll be fun.
22	**@ent02**	All right?

(b) Passages generated for the dialog in (a).

ID	Passage
1	**@ent03** announces that **@ent03** is going to be teaching a graduate class at the university.
2	**@ent02** misses dressing up for romantic dates so **@ent01** promises to take **@ent02** out.
3	**@ent02** misses dating, so **@ent01** promises to show **@ent02** a good time.
4	**@ent01** asks **@ent02** where to go on a date and then **@ent01** decides to take **@ent02** on a date to get **@ent02**'s mind off having a baby.

(c) Queries generated from the passages in (b).

ID	Passage
1.a	$\boldsymbol{x}$ announces that **@ent03** is going to be teaching a graduate class at the university.
1.b	**@ent03** announces that $\boldsymbol{x}$ is going to be teaching a graduate class at the university.
2.a	$\boldsymbol{x}$ misses dressing up for romantic dates so **@ent01** promises to take **@ent02** out.
2.b	**@ent02** misses dressing up for romantic dates so $\boldsymbol{x}$ promises to take **@ent02** out.
2.c	**@ent02** misses dressing up for romantic dates so **@ent01** promises to take $\boldsymbol{x}$ out.
...	

Table 1: An example dialog and its passages and queries from our corpus. All mentions are encoded by their entity IDs. The queries are generated by replacing each unique entity in every passage with the variable x (Section 3.2). @ent01: Joey, @ent02: Rachel, @ent03: Ross, @ent04: Neuman, @ent05: Paul.

is then queried to Elasticsearch that has indexed the selected scenes, and the scene with the highest relevance is retrieved. Finally, the retrieved scene along with the queried sentence are sent to a crowd worker who is asked to determine whether or not they are relevant, and perform anaphora resolution to replace all pronouns in the sentence with the corresponding character names. The sentence that is checked for the relevancy and processed by the anaphora resolution is considered a passage.

Out of 6,014 sentences collected from the plot summaries, 2,994 of them got turned into passages; in other words, about a half of the sentences could not be paired with relevant scenes by Elasticsearch. In addition to these pseudo-generated passages, two more sets of passages are created. For the first set, crowd workers are asked to generate new passages including factual descriptions different from the ones that are pseudo-generated. This produced additional 615 passages; however, passages in this set could be biased toward the dominant characters. To increase the diversity of the character entities in the passages, crowd workers are asked to generate the second set of passages that include factual descriptions related to only non-dominant characters. A total of 1,037 passages are generated in this set, which makes passage completion even more challenging since the chance of the dominant characters being the answers becomes much lower with this second set. Figure 1 shows the overview of passage generation. Note that Amazon Mechanical Turk is used for all crowdsourcing.

3.2 Mention Annotation

For all dialogs and their passages, mentions are first detected automatically by the named entity recognizer in NLP4J (Choi, 2016) using the PERSON entity, then manually corrected. For each passage including multiple mentions, a query is created for every mention by replacing it with the variable x:

Rachel misses dating, so *Joey* offers to take *Rachel* out.
⇒ $\boldsymbol{x}$ misses dating, so *Joey* offers to take *Rachel* out.
⇒ *Rachel* misses dating, so $\boldsymbol{x}$ offers to take *Rachel* out.
⇒ *Rachel* misses dating, so *Joey* offers to take $\boldsymbol{x}$ out.

Following Hermann et al. (2015), all mentions implying the same character are encoded by the same entity ID. A different set of entity IDs are randomly generated for each dialog; for the above example, *Joey* and *Rachel* may be encoded by **@ent01** and **@ent02** in this dialog (Table 1), although they can be encoded by different entity IDs in other dialogs. This random encoding prevents learning models from overfitting to certain types of entities. On the other hand, the same set of entity IDs are applied to the passages associated with the dialog.

One issue still remains that characters in this dataset are often mentioned by several aliases (e.g., nicknames, honorifics) such that it is not trivial to cluster mentions implying the same character using simple string matching. Thus, an entity dictionary is created for each character whose key is the name of the character and the value is a list of aliases for the character, manually inspected throughout the entire show. This entity dictionary is then used to link mentions in both the dialogs and the passages to their character entities.

Type	Count
# of dialogs	1,681
# of passages	4,646
# of queries	13,487
Avg. # of utterances per dialog	15.8
Avg. # of tokens per dialog/passage	290.8 / 19.9
Avg. # of mentions per dialog/passage	24.4 / 3.0
Avg. # of entities per dialog/passage	5.4 / 2.2
Max # of mentions per dialog/passage	117 / 15
Max # of entities per dialog/passage	16 / 7

Table 2: The overall statistics of our corpus.

Table 2 shows the overall statistics of our corpus. It is relatively smaller than the other corpora (Section 2). However, it is the largest, if not the only, corpus for the evaluation of passage completion on multiparty dialog that still gives enough instances to develop meaningful models using deep learning.

4 Approach

This section presents our deep learning architecture that integrates rich feature extraction from convolutional neural networks (CNN) into robust sequence modeling in recurrent neural networks (RNN) (Section 4.1). The combination of CNN and RNN has been adapted by several NLP tasks such as text summarization (Cheng and Lapata, 2016), essay scoring (Dong et al., 2017), sentiment analysis (Wang et al., 2016), or even reading comprehension (Dhingra et al., 2017). Unlike previous works that feed a sequence of sentences encoded by CNN to RNN, a sequence of utterances is encoded by CNN in our model, where each utterance is spoken by a distinct speaker and contains one or more sentences that are coherent in topics. Our best model is optimized by both the utterance (Section 4.2) and the dialog (Section 4.3) level attentions, showing significant improvement over the pure CNN+RNN model.

4.1 CNN + LSTM

Each utterance comes with a speaker label encoded by the entity ID in our corpus (Table 1). This entity ID is treated as the first word of the utterance in our models. Before training, random embeddings are generated for all entity IDs and the variable x with the same dimension d as word embeddings. All utterances and queries are zero-padded to their maximum lengths m and n, respectively.

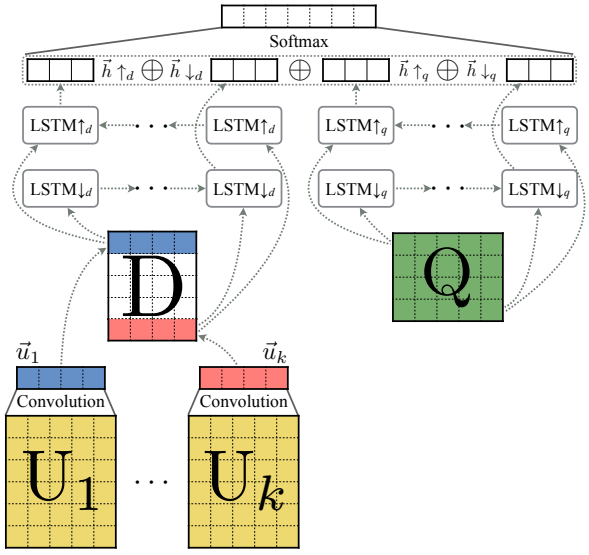


Figure 2: The overview of the CNN+LSTM model.

Given a query and a dialog comprising k-number of utterances, the query matrix $\mathrm{Q} \in \mathcal{R}^{n \times d}$ and the utterance matrix $\mathrm{U}_i \in \mathcal{R}^{m \times d}$ are created using the

word, entity, and variable embeddings $\forall i \in [1, k]$. For each U_i, 2D convolutions are performed for 2-5 grams, where each convolution takes f-number of filters and the output of every filter is max-pooled, resulting a vector of the size f. These vectors are concatenated to create the utterance embedding $\vec{u}_i \in \mathcal{R}^{1\times 4\cdot f}$, then the utterance embeddings are stacked to generate the dialog matrix $\mathrm{D} \in \mathcal{R}^{k\times 4\cdot f}$. This dialog matrix is fed into a bidirectional LSTM consisting of two networks, $\texttt{LSTM}{\downarrow_d}$ and $\texttt{LSTM}{\uparrow_d}$, that process the sequence of utterance embeddings in both directions. In parallel, Q is fed into another bidirectional LSTM with $\texttt{LSTM}{\downarrow_q}$ and $\texttt{LSTM}{\uparrow_q}$ that process the sequence of word embeddings in Q. Each LSTM returns two vectors from the last hidden states of $\texttt{LSTM}{\downarrow_*}$ and $\texttt{LSTM}{\uparrow_*}$:

$$\vec{h}\downarrow_d = \texttt{LSTM}\downarrow_d(\mathrm{D}) \quad \vec{h}\uparrow_d = \texttt{LSTM}\uparrow_d(\mathrm{D})$$
$$\vec{h}\downarrow_q = \texttt{LSTM}\downarrow_q(\mathrm{Q}) \quad \vec{h}\uparrow_q = \texttt{LSTM}\uparrow_q(\mathrm{Q})$$

All the outputs of LSTMs are concatenated and fed into the softmax layer that predicts the most likely entity for x in the query, where each dimension of the output layer represents a separate entity:

$$\mathrm{O} = \mathrm{softmax}(\vec{h}\downarrow_d \oplus \vec{h}\uparrow_d \oplus \vec{h}\downarrow_q \oplus \vec{h}\uparrow_q)$$
$$\mathrm{predict}(\mathrm{U}_1, \ldots, \mathrm{U}_k, \mathrm{Q}) = \mathrm{argmax}(\mathrm{O})$$

Figure 2 demonstrates our CNN+LSTM model that shows significant advantage over the pure bidirectional LSTM model as dialogs get longer.

4.2 Utterance-level Attention

Inspired by Yin et al. (2016), attention is applied to every word pair in the utterances and the query. First, the similarity matrix $\mathrm{S}_i \in \mathcal{R}^{m\times n}$ is created for each utterance matrix U_i by measuring the similarity score between every word in U_i and Q:

$$\mathrm{S}_i[r, c] = \mathrm{sim}(\mathrm{U}_i[r, :], \mathrm{Q}[c, :])$$
$$\mathrm{sim}(x, y) = 1/(1+\|x-y\|)$$

The similarity matrix is then multiplied by the attention matrix $\mathrm{A} \in \mathcal{R}^{n\times d}$ learned during the training. The output of this multiplication produces another utterance embedding $\mathrm{U}'_i \in \mathcal{R}^{m\times d}$, which is channeled to the original utterance embedding U_i and generates the 3D matrix $\mathrm{V}_i \in \mathcal{R}^{2\times m\times d}$ (Figure 3):

$$\mathrm{U}'_i = \mathrm{S}_i \cdot \mathrm{A}$$
$$\mathrm{V}_i = \mathrm{U}_i \oslash \mathrm{U}'_i$$

V_i is fed into the CNN in Section 4.1 instead of U_i and constructs the dialog matrix D.

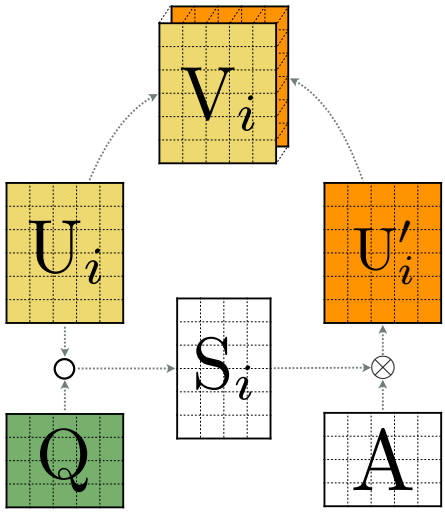


Figure 3: The overview of the utterance-level attention.

4.3 Dialog-level Attention

The utterance-level attention is for the optimization of local contents through word similarities between the query and the utterances. To give a global view to the model, dialog-level attention is applied to the query matrix Q and the dialog matrix D. First, 1D convolutions are applied to each row in Q and D, generating another query matrix $\mathrm{Q}' \in \mathcal{R}^{n\times e}$ and dialog matrix $\mathrm{D}' \in \mathcal{R}^{m\times e}$, where e is the number of filters used for the convolutions.

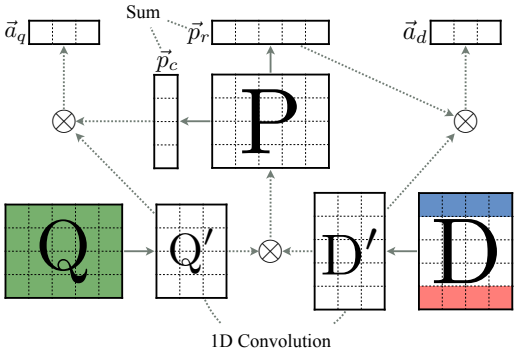


Figure 4: The overview of the dialog-level attention.

Q' is then multiplied to D'^T, resulting another similarity matrix $\mathrm{P} \in \mathcal{R}^{n\times m}$. Furthermore, the sum of each row in P is concatenated to create $\vec{p}_c \in \mathcal{R}^{n\times 1}$, and the sum of each column in P is also concatenated to create $\vec{p}_r \in \mathcal{R}^{1\times m}$:

$$\mathrm{P} = \mathrm{Q}' \cdot \mathrm{D}'^T$$
$$\vec{p}_c[r] = \sum_{j=1}^{m} P[r, j]$$
$$\vec{p}_r[c] = \sum_{j=1}^{n} P[j, c]$$

$\vec{p}_c^T$ is multiplied to Q' and $\vec{p}_r$ is multiplied to D', producing the attention embeddings $\vec{a}_q \in \mathcal{R}^{1\times e}$

Model	Development Set				Evaluation Set			
	Org.	25	50	100	Org.	25	50	100
Human Evaluation	-	-	-	-	**74.02**	-	-	-
Majority	28.61	27.65	21.57	19.79	30.08	28.23	21.58	17.59
Entity Centric	52.28	45.29	45.82	42.17	47.36	43.83	45.56	42.47
Bi-LSTM	72.24	68.90	64.51	55.17	71.21	67.37	62.95	53.76
CNN+LSTM	70.97	70.24	69.40	65.43	70.28	69.20	68.35	64.13
CNN+LSTM+UA	**72.42**	71.73	70.67	66.46	71.84	69.88	69.18	**66.99**
CNN+LSTM+DA	72.24	71.30	70.21	66.37	71.46	69.88	69.30	65.51
CNN+LSTM+UA+DA	72.21	**72.14**	**71.45**	**67.86**	**72.42**	**71.01**	**69.98**	**66.99**

Table 3: Results on the development and the evaluation sets from all models.

and $\vec{a}_d \in \mathcal{R}^{1\times e}$, respectively. Finally, these attention embeddings are concatenated with the outputs of the LSTMs in Section 4.1 then fed into the softmax layer to make the prediction:

$$\vec{a}_q = \vec{p}_c^{\,T} \cdot \mathrm{Q}'$$
$$\vec{a}_d = \vec{p}_r \cdot \mathrm{D}'$$
$$\mathrm{O} = \mathrm{softmax}(\vec{h}\downarrow_d \oplus \vec{h}\uparrow_d \oplus \vec{h}\downarrow_q \oplus \vec{h}\uparrow_q \oplus \vec{a}_d \oplus \vec{a}_q)$$
$$\mathrm{predict}(\mathrm{U}_1, \ldots, \mathrm{U}_k, \mathrm{Q}) = \mathrm{argmax}(\mathrm{O})$$

Similar attentions have been proposed by Yin et al. (2016) and evaluated on NLP tasks such as answer selection, paraphrase identification, and textual entailment; however, they have not been adapted to passage completion. It is worth mentioning that we have tried many other kinds of attention mechanisms and empirically found that the combination of these two attentions yields the best result for the passage completion task.

5 Experiments

The Glove 100-dimensional pre-trained word embeddings (Pennington et al., 2014) are used for all experiments ($d = 100$). The maximum lengths of utterances and queries are $m = 92$ and $n = 126$, and the maximum number of utterances is $k = 25$. For the 2/1D convolutions in Sections 4.1 and 4.3, $f = e = 50$ filters are used, and the `ReLu` activation is applied to all convolutional layers. The dimension of the LSTM outputs $\vec{h}\downarrow\uparrow_*$ is 32, and the `tanh` activation is applied to all hidden states of LSTMs. Finally, the `Adam` optimizer with the learning rate of 0.001 is used to learn the weights of all models. Table 4 shows the dataset split for our experiments that roughly gives 80/10/10% for training/development/evaluation sets.

	Train	Develop	Evaluate	Total
Queries	10,785	1,349	1,353	13,487

Table 4: Dataset split for our experiments, where each query is considered a separate instance.

5.1 Utterance Pruning

Most utterances in our corpus are relatively short except for a few ones so that padding all utterances to their maximum length is practically inefficient. Thus, pruning is used for those long utterances. For any utterance containing more than 80 words, that is about 1% of the entire dataset, stopwords are removed. If the utterance still has over 80 words, all words whose document frequencies are among the top 5% in the training set are removed. If the length is still greater than 80, all words whose document frequencies are among the top 30% in the training set are removed. By doing so, we reduce down the maximum length of utterances from 1,066 to 92, which dramatically speeds up the modeling without compromising the accuracy.

5.2 Datasets with Longer Dialogs

The average number of utterances per dialog is 15.8 in our corpus, which is relatively short. To demonstrate the model robustness for longer dialogs, three more datasets are created in which all dialogs have the fixed lengths of 25, 50, and 100 by borrowing utterances from their consecutive scenes. The same sets of queries are used although models need to search through much longer dialogs in order to answer the queries for these new datasets. The three pseudo-generated datasets as well as the original dataset are used for all our experiments.

5.3 Human Evaluation

Human performance is examined on the evaluation set of the original length using crowdsourcing. The workers are presented with passages and the corresponding dialogs and asked to choose the answer from the list of entities that appear in the dialog. For fair comparisons, the encoded input where the character names are replaced with the entity IDs are used for this evaluation as well, to minimize the bias from external knowledge.

5.4 Baselines

Three models are used to establish comprehensible baseline results:

Majority This model picks the dominant entity in the dialog as the answer for each query.

Entity Centric This is our reimplementation of Chen et al. (2016)'s entity centric model. Our entity centric model was evaluated on the CNN/Daily Mail dataset and showed a comparable result to the previous work.

Bi-LSTM This is the bidirectional LSTM model introduced by Chen et al. (2016), which outperforms their entity centric model by a large margin. We use their implementation of this model;[3] the input to this model is a list of words across all utterances within the dialog. All hyperparameters are tuned using the development set.

5.5 Results

Table 3 shows the results from all models. The human performance on the evaluation set is only 1.6+% higher than the best performing model, which on part shows the difficulty of the task. It should be noted that character anonymization process makes it harder to for people to find the answer. However, it also possible that some participants of the evaluation may enter the answer randomly (i.e the results may not truly reflect human performance). Notice that the performance of the majority model on our dataset is similar to the ones in the CNN/Daily Mail dataset, which validates the level of difficulty in our corpus. As expected, the entity centric model sets its performance in between the majority model and the other deep learning models. For all of our models and Bi-LSTM, experiments are run three times with different random seeds and the accuracies are averaged. The accuracy of Bi-LSTM reported on the CNN dataset is 72.4, which is similar to its performance on our dataset. Our models coupled with both the utterance-level and the dialog-level attentions (CNN+LSTM+UA+DA) outperform all the other models except for the one on the development set of the original dataset. Our models show significant advantage over Bi-LSTM as the length of the dialog gets larger.

Figure 5 shows the learning curves from Bi-LSTM and CNN+LSTM+UA+DA on the original dataset. The red circle and the black star mark the peaks of CNN+LSTM+UA+DA and Bi-LSTM, respectively. Although the accuracies between these models are very similar, our model converges in fewer epochs. Figure 6 shows the learning curves from both models in 3 trials on the length-100 dataset. Our models take fewer epochs to converge and the variance of performance across trials is smaller, implying that our models are not as sensitive to the hyperparameter tuning as Bi-LSTM.

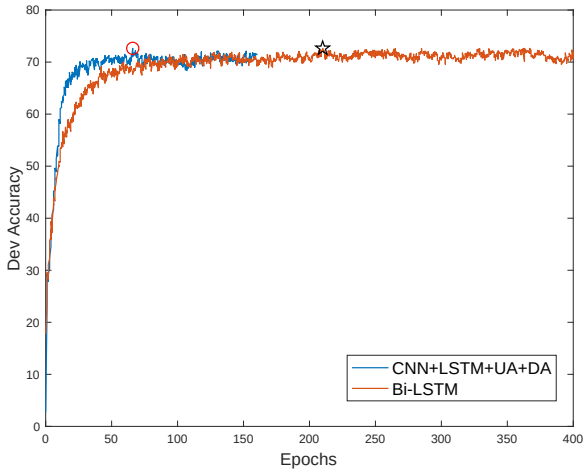


Figure 5: Training curves on the original dataset.

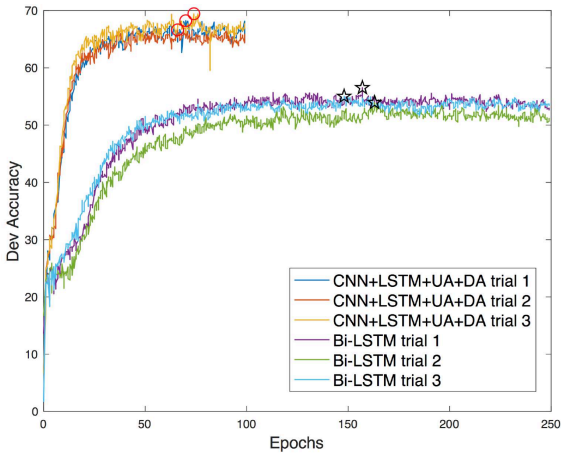


Figure 6: Training curves on the length-100 dataset.

6 Analysis

6.1 Attention Visualization

Figure 7 depicts the dialog-level attention matrix, that is P in Section 4.3, for the example in Table 1. The x-axis and y-axis denote utterances and words in the query, respectively. Each cell represents the attention value between a word in the query and an utterance. From this visualization, we see that query words such as *misses*, *take*, *good*, and *time*

[3] github.com/danqi/rc-cnn-dailymail

have the most attention from utterances as they are the keywords to find the answer entity. The utterances 14, 15 and 17 that give out the answer also get relatively high attention from the query words. This illustrates the effectiveness of the dialog-level attention in our model.

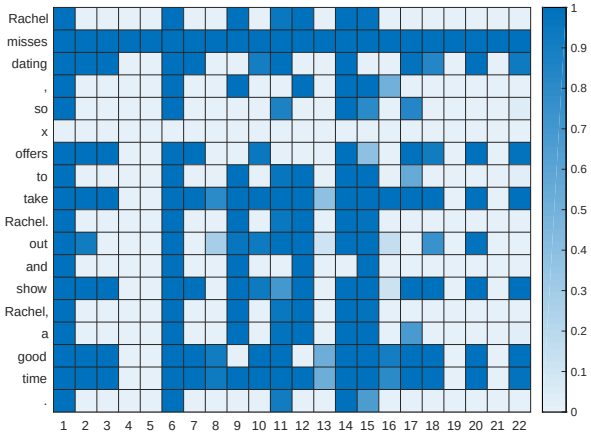


Figure 7: Visualization of the dialog-level attention matrix P for the example in Table 1.

6.2 Comparisons

Table 5 shows the confusion matrix between Bi-LSTM and CNN+LSTM +UA+DA on the original dataset. During the error analysis, it is noticed that Bi-LSTM is better at capturing exact string matches or paraphrases. As shown by the first two examples in Table 6, it is clear that those queries can be answered by capturing just the snippets of the dialogs. In the first example, "$\boldsymbol{x}$ makes up his mind about something" in the query matches "@ent06 sets his mind on something" in the dialog.

Model	Bi-LSTM: T	Bi-LSTM: F
C+L+U+D: T	850	133
C+L+U+D: F	118	252

Table 5: The confusion matrix between Bi-LSTM and CNN+LSTM+UA+DA.

In the second example, query phrase "the closet that $\boldsymbol{x}$ and @ent03 were in" also has the exact string match"the closet @ent18 and @ent03 were in" in the dialog. Although these cues are usually parts of sentences in long utterances, since Bi-LSTM is based on only words, it still is able to locate them correctly. On the other hand, our model encodes each utterance and then feeds encoded vectors to LSTMs, so the high level representation of the cues are mixed with other information, which hinders the model's ability to find the exact string matches. Our model is better at answering queries that require inference from multiple utterances. As shown by the last two examples in Table 6, the cues to the answers distribute across several utterances and there is no obvious match of words or phrases. In the third example, the model needs to infer that in the sentence "(She reaches over to look at the label on the box)", she refers to @ent18 and connect this information with the later utterance by @ent18 "This is addressed to Mrs. @ent16 downstairs" in order to answer the query. In the last example, finding the correct answer requires the model to interpret that the utterances "What the hell was that?!" and "(They both scream and jump away.)" reflect the outcome of *startles*, which is the verb in the query. As dialogs become longer in the padded datasets, because of the utterance encoding procedure, our model's ability's ability to locate relevant part of dialog is not influenced as much, whereas it becomes much more difficult for Bi-LSTM to find the matches.

6.3 Discussion

It is worth mentioning that besides the models presented in Section 4, the attention-over-attention reader was also experimented with our dataset, which outperformed various neural systems by a large margin on both the CNN news dataset and the Children Book Test dataset (Cui et al., 2017). We first reimplemented their model and experimented on the CNN dataset and achieved similar results as reported in the previous paper. We then experimented this model on our original length dataset. However, even after an extensive hyperparameter turning on the development set, this model did not achieve results comparable to those of either Bi-LSTM or our models, so we did not make a further analysis on this model.

7 Conclusion

We introduce a new corpus consisting of multiparty dialogs and crowdsourced annotation for the task of passage completion. To the best of our knowledge, this is the first corpus that can challenge deep learning models for passage completion on this genre. We also present a deep learning architecture combining convolutional and recurrent neural networks, coupled with utterance-level and dialog-level attentions. Models trained by our architecture significantly outperform the one trained by the pure bidirectional LSTM, especially on longer

Model	Query	Dialog
Bi-LSTM	@ent12 says that once ***x*** makes up his mind about something, @ent06 will have xxx with it.	Because you know as well as I do that once **@ent06** sets his mind on something, more often than not, he 's going to have sex with it.
Bi-LSTM	@ent06 points out that people are screwing in the closet that ***x*** and @ent03 were in.	Oh, by the way. Two people screwing in there (points to the closet **@ent18** and @ent03 were in) if you want to check that out.
CNN+LSTM +UA+DA	***x*** saw on the box that the cheesecake was addressed to Mrs. @ent16.	**@ent18** This is the best cheesecake I have ever had. Where did you get this? (She reaches over to look at the label on the box.) @ent10 It was at the front door. When I got home. Somebody sent it to us. **@ent18** @ent10, this is not addressed to you. This is addressed to Mrs. @ent16 downstairs. ...
CNN+LSTM +UA+DA	@ent17 startles @ent02 and ***x*** in the hallway to prove @ent17' point, which sets off an on-going competition of psuedo-attacks.	@ent17 DANGER !!! DANGER !!!!! @ent02 @ent17 !!! **@ent03** What the hell was that ?! (They both scream and jump away.)

Table 6: Examples for model comparison. The first column denotes the model that makes the correct prediction.

dialogs. Our analysis demonstrates the comprehension of our model using the attention matrix. For the future work, we will expand the annotation for more entity types and automatically link mentions with respect to their entities using an entity linker. All our resources including the annotated corpus and source codes of the models are available at: https://github.com/emorynlp/reading-comprehension.

References

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 2358–2367. http://www.aclweb.org/anthology/P16-1223.

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

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

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 484–494. http://www.aclweb.org/anthology/P16-1046.

Jinho D. Choi. 2016. Dynamic Feature Induction: The Last Gist to the State-of-the-Art. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. NAACL'16.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2017. Attention-over-attention neural networks for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 593–602. http://aclweb.org/anthology/P17-1055.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2017. Gated-attention readers for text comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1832–1846. http://aclweb.org/anthology/P17-1168.

Fei Dong, Yue Zhang, and Jie Yang. 2017. Attention-based recurrent convolutional neural network for automatic essay scoring. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. Association for Computational Linguistics, Vancouver, Canada, pages 153–162. http://aclweb.org/anthology/K17-1017.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching Machines to Read and Comprehend. In *Annual Conference on Neural Information Processing Systems*. NIPS'15, pages 1693–1701.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The Goldilocks Principle: Reading

Children's Books with Explicit Memory Representations. In *Proceedings of the 6th International Conference on Learning Representations*. ICLR'16.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1601–1611. http://aclweb.org/anthology/P17-1147.

Tomasz Jurczyk and J. D. Choi. 2017. Cross-domain Document Retrieval: Matching between Conversational and Formal Writings. In *Proceedings of the EMNLP Workshop on Building Linguistically Generalizable NLP Systems*. Copenhagen, Denmark, BLGNLP'17, pages 48–53. http://generalizablenlp.weebly.com.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 785–794. https://www.aclweb.org/anthology/D17-1082.

Takeshi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2016. Who did what: A large-scale person-centered cloze dataset. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 2230–2235. https://aclweb.org/anthology/D16-1241.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernandez. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1525–1534. http://www.aclweb.org/anthology/P16-1144.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 2383–2392. https://aclweb.org/anthology/D16-1264.

Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. 2013. MCTest: A Challenge Dataset for the Open-Domain Machine Comprehension of Text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. EMNLP'13, pages 193–203.

Adam Trischler, Zheng Ye, Xingdi Yuan, Philip Bachman, Alessandro Sordoni, and Kaheer Suleman. 2016. Natural Language Comprehension with the EpiReader. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 128–137. https://aclweb.org/anthology/D16-1013.

Jin Wang, Liang-Chih Yu, K. Robert Lai, and Xuejie Zhang. 2016. Dimensional sentiment analysis using a regional cnn-lstm model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 225–230. http://anthology.aclweb.org/P16-2037.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks. *arXiv* 1502.05698.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics* 4:259–272. https://transacl.org/ojs/index.php/tacl/article/view/831.

Dialog Generation Using Multi-turn Reasoning Neural Networks

Xianchao Wu[1], **Ander Martínez**[2]*, **Momo Klyen**[1]

[1]A.I.&Research, Microsoft Development Co., Ltd.
[2] Graduate School of Information Science, Nara Institute of Science and Technology
{xiancwu, momokl}@microsoft.com, ander.martinez.zy4@is.naist.jp

Abstract

In this paper, we propose a generalizable dialog generation approach that adapts *multi-turn reasoning*, one recent advancement in the field of document comprehension, to generate responses ("answers") by taking current conversation session context as a "document" and current query as a "question". The major idea is to represent a conversation session into memories upon which attention-based memory reading mechanism can be performed multiple times, so that (1) user's query is properly extended by contextual clues and (2) optimal responses are step-by-step generated. Considering that the speakers of one conversation are not limited to be one, we separate the single memory used for document comprehension into different groups for speaker-specific topic and opinion embedding. Namely, we utilize the queries' memory, the responses' memory, and their unified memory, following the time sequence of the conversation session. Experiments on Japanese 10-sentence (5-round) conversation modeling show impressive results on how multi-turn reasoning can produce more diverse and acceptable responses than state-of-the-art single-turn and non-reasoning baselines.

1 Introduction

Dialogue systems such as chatbots are a thriving topic that is attracting increasing attentions from researchers (Sordoni et al., 2015; Serban et al., 2016; Li et al., 2015; Wen et al., 2016). Recent achievements, such as deep neural networks for text generating, user profiling (Li et al., 2014), and natural language understanding, have accelerated the progresses of this field, which was historically approached by conventional rule-based and/or statistical response ranking strategies.

Response ranking models retrieve the most suitable response(s) from a fixed set of (question, answer) pairs given a dialogue context and current query from a user (Banchs and Li, 2012; Lowe et al., 2015). Learning-to-rank approaches were applied to compute the similarity scores of between (query, context) and indexed candidate (question, answer) pairs to return the optimal "answer" to the user. These ranking-based retrieval strategies have been well-applied as an important approach to dialogue systems, yet the set of scripted responses are limited and are short at generalization. On the other hand, statistical machine translation (SMT) systems have been applied to dialogue systems (Ritter et al., 2011), taking user's query as a source language sentence and the chatbot's response as a target language sentence. Labeled data for learning-to-ranking training will not be necessary anymore and all we need is the large-scale (question, answer) pairs.

The sequence-to-sequence model proposed in (Sutskever et al., 2014) applied end-to-end training of neural networks to text generation. This model, further enhanced by an attention mechanism (Bahdanau et al., 2014), was generic and allowed its application to numerous sequence-to-sequence learning tasks such as neural machine translation (NMT) (Cho et al., 2014; Bahdanau et al., 2014), image captioning (Donahue et al., 2015; Mao et al., 2015), speech recognition (Chan et al., 2015) and constituency parsing (Vinyals et al., 2015). The simplicity of these models makes them attractive, since "translation" and "alignment" are learned jointly on the fly.

Specially, Vinyals and Le (2015) applied the sequence-to-sequence model to conversational modeling and achieved impressive results on various datasets. Their model was trained to predict a response given the previous sentence (s). Shang et al. (2015) combined local and global attentions

*Work done when Ander was an intern in Microsoft. Wu and Ander contributed equally to this paper.

Proceedings of NAACL-HLT 2018, pages 2049–2059
New Orleans, Louisiana, June 1 - 6, 2018.

and reported better results than retrieval based systems. Sordoni et al. (2015) explored three different end-to-end approaches for the problem of predicting the response given a query attached with a single message context.

Multi-turn conversation modeling is considered to be more difficult than machine translation, since there are many more acceptable responses for a given (context, query) input and these often rely on external knowledge and/or contextual reasoning. Dialogue systems trained with a maximum likelihood estimation (MLE) objective function, as most SMT utilizes, often learn to reply generic sentences as "I don't know" or "sounds good", which have a high incidence in the "answer" part of (question, answer) style training datasets. There have been various attempts at *diversifying* the responses (Li et al., 2016a; Yu et al., 2016; Li et al., 2017) but the lack of variations in the responses remains as an essential challenge. We wonder that if this stress can be relieved by modeling the prior context in a rather fine-grained way.

In document comprehension fields, multi-turn reasoning (also called multi-hop reasoning) has delivered impressive results by assimilating various pieces of information to produce an unified answer (Hill et al., 2015; Dhingra et al., 2016). Through multi-turn reading the document's memory using attention models, current question can be extended with much richer knowledge. This makes it easier to figure out the correct answer from that document. Different documents need to be read different times to yield out the correct answer for the input question. Specially, Shen et al. (2016) use a dynamic number of turns by introducing a termination gate to control the number of iterations of reading and reasoning.

Motivated by the reasoning network for document comprehension (Shen et al., 2016), we propose multi-turn reasoning neural networks that generate the proper response (or, "answer") by attention-based reasoning from current conversation session ("document") and current query (identical to "question" in document comprehension) from the user. In particular, our networks utilize conversation context and explicitly separate speakers' interventions into sentence-level and conversation-level memories. Our first model uses plain single-turn attention to integrate all the memories, and the second approach integrates multi-turn reasoning. The formulation of our proposed approach is designed in a generalized way, allowing for inclusion of additional information such as external knowledge bases (Yih and Ma, 2016; Ghazvininejad et al., 2017; Han et al., 2015) or emotional memories (Zhou et al., 2017). Moreover, our approach for two-speaker scenario can be easily extended to group chatting by a further speaker-specific memory splitting.

We evaluate the performances of our methods by comparing three configurations trained on a Japanese twitter conversation session dataset. Each conversation session contains 10 sentences which are 5-round between two real-world speakers. The results provide evidences that multi-turn reasoning neural networks can help improving the consistency and diversity of multi-turn conversation modeling.

This paper is structured as follows: Section 2 gives a general description of multi-turn conversation modeling; Section 3 describes background neural language modeling, text generation, and attention mechanisms; Section 4.1 first introduces a model with multiple attention modules and then explains how the multi-turn reasoning mechanism can be further integrated into the previous models; Sections 5, 6 and 7 describe the experimental settings and results using automatic evaluation metrics, detailed human-evaluation based analysis, and conclusions, respectively.

2 Multi-turn Conversation Modeling

Consider a dataset $\mathcal{D}$ consisting of a list of conversations between two speakers. Each conversation $d \in \mathcal{D}$ is an ordered sequence of sentences s_i, where $i \in [1, T_d]$ and T_d is the number of sentences in d, produced by two speakers alternately. In this way, for $s_i, s_j \in d$, both sentences are from the same speaker if and only if $i \equiv j \pmod 2$. Note that, our definition includes the case that one speaker continuously expresses his/her message through several sentences. We simply concatenate these sentences into one to ensure that the conversation is modeled with alternate speakers.

A multi-turn conversation model is trained to search parameters that maximize the likelihood of every sentence $s_i \in d$ where $i \geq 2$, supposing that the beginning sentence s_1 is always given as a precondition:

$$\theta_M = \arg\max_{\theta} \{\mathcal{L}(\theta, \mathcal{D})\}, \quad (1)$$

where

$$\mathcal{L}(\theta, \mathcal{D}) = \sum_{d \in \mathcal{D}} \prod_{i=2}^{T_d} p(s_i | s_{<i}). \quad (2)$$

Here, $s_{<i}$ are sentences $s_j \in d$ and $j < i$.

The probability of each sentence, $p(s_i|s_{<i})$, is frequently estimated by a conditional language model. Note that, traditional single-turn conversation models or NMT models are a special case of this model by simply setting T_d to be 2. That is, the generation of the next sentence is session-insensitive and is only determined by the former single sentence. Another aspect of understanding this contextual conversation model is that, the number of reference contextual sentences $s_{<i}$ is not limited to be one. Suppose there are already 9 sentences known in one conversation session and we want to generate the 10-th sentence, then from $p(s_1)$ to $p(s_9)$ are all preconditions and we will only need to focus on estimating $p(s_{10}|s_{<10})$.

We adapt sequence-to-sequence neural models (Sutskever et al., 2014) for multi-turn conversation modeling. They are separated into an encoder part and a decoder part. The encoder part applies a RNN on the input sequence(s) $s_{<i}$ to yield prior information. The decoder part estimates the probability of the generated output sequence s_i by employing the last hidden state of the encoder as the initial hidden state of the decoder. Sutskever et al. (2014) applied this technique to NMT and impressive experimental results were reported thereafter.

Using Equation 2, we are modeling two chatbots talking with each other, since all the $s_{2,\ldots,T_d}$ are modeled step-by-step on the fly. However, we can add constraints to determine whose responses to be generated, either one speaker or both of them. That is, when i takes odd integers of 1, 3, 5 and so on, we are modeling the first speaker. Even integers of i indicates a generation of responses for the second speaker.

3 Language Modeling and Text Generation

Language models (LM) are trained to compute the probability of a sequence of tokens (words or characters or other linguistic units) being a linguistical sentence. Frequently, the probability of a sentence s with T_s tokens is computed by the production of the probabilities of each token $y_j \in s$ given its contexts $y_{<j}$ and $y_{>j}$:

$$p(s) = \prod_{j=1}^{T_s} p(y_j | y_{<j}, y_{>j}). \quad (3)$$

When generating a sequence based on a LM, we can generate one word at a time based on the previously predicted words. In this situation, only the previously predicted words are known and the probability of current sequence is approximated by dropping the posterior context. That is,

$$p(y_j | y_{<j}, y_{>j}) \approx p(y_j | y_{<j}). \quad (4)$$

We construct a sequence generation LM using sequence-to-sequence neural network f. The neural network intercalates linear combinations and non-linear activate functions to estimate the probability of mass function. Then, in the encoder part of f, the contextual information is represented by a fixed-size hidden vector h_j:

$$p(y_j | y_{<j}, y_{>j}) \approx f(y_{j-1}, h_j, \theta_f), \quad (5)$$

where θ_f represents f's trainable parameters.

To embed the previous word sequence into a fixed-size vector, recurrent neural networks (RNN) such as long short term memory (LSTM) networks (Hochreiter and Schmidhuber, 1997) or gated recurrent units (GRU) (Cho et al., 2014) are widely used. These networks repeat a recurrent operation on each input word:

$$h_j = g(h_{j-1}, y_{j-1}, \theta_g), \quad (6)$$

where θ_g represents the trainable parameters of a RNN function g, and h_j is the hidden state of the RNN at time j.

3.1 Conditional Language Modeling

The hidden state h of a RNN can accumulate information from previous words ($y_j \in s$ and $j < T_s$) or previous sentences ($s_i \in d$ and $i < T_d$) which ensures the encoding and decoding processes in sequence-to-sequence models. Since the contextual sentences are known already, the encoder can represent them in both forward ($\vec{h}_j$) and backward ($\overleftarrow{h}_j$) directions. The results from both recursions can be combined by a concatenated operation. This is referred to as bidirectional RNN shorted as BiRNN (Schuster and Paliwal, 1997).

$$h_j = [\vec{h}_j^\top ; \overleftarrow{h}_j^\top]^\top. \quad (7)$$

For each sentence $s_i \in d$ ($i < T_d$), we annotate the combination of the final states of each RNN direction as a memory vector $m_i = [(\vec{h}^{(i)}_{T_{s_i}})^{\mathrm{T}}; (\overleftarrow{h}^{(i)}_1)^{\mathrm{T}}]^{\mathrm{T}}$. A projection of annotation m_i can be used as the decoder's initial state t_0 such as $t_0 = \tanh(W_s m_{T_{d'}})$ and $T_{d'} < T_d$. W_s here is a weight matrix that projects $m_{T_{d'}}$ into a vector that shares a same dimension with t_0. In (Bahdanau et al., 2014) for NMT, $\overleftarrow{h}_1$, backward encoding of a single source sentence, was used to initialize $t_0 = \tanh(W_s \overleftarrow{h}_1)$.

3.2 Attention Mechanism

Summarizing all contextual information into one single fixed-length vector becomes weaker to guide the generation of the target sentence as the contextual information grows longer. To tackle this problem, an attention mechanism (Bahdanau et al., 2014) was applied to NMT for learning to align and translate jointly. In this attention-based model, the conditional probability in Equation 4 is defined as:

$$p(y_j|y_{<j}) = f(y_{j-1}, t_j, c_j, \theta_f), \tag{8}$$

where

$$t_j = g(t_{j-1}, y_{j-1}, c_j, \theta_g) \tag{9}$$

is a RNN hidden state in the decoder part for time j and $c_j = \sum_{i=1}^{T_s} \alpha_i^{(j)} h_i$ is a *context vector*, a weighted combination of the annotation set memory $(h_1, ..., h_{T_s})$ produced by encoding a source sentence s with length T_s. The weight $\alpha_i^{(j)}$ of each (source) annotation h_i is computed by

$$\alpha_i^{(j)} = \frac{\exp(e_i^{(j)})}{\sum_{l=1}^{T_s} \exp(e_l^{(j)})}. \tag{10}$$

where $e_i^{(j)}$ is an *alignment model* and is implemented by a feed-forward network a:

$$e_i^{(j)} = a(h_i, t_{j-1}). \tag{11}$$

This method was applied to single-turn conversation modeling (Vinyals and Le, 2015). We use this model, with attention over each immediately previous sentence $s_{i-1} \in d$ for generating s_i, as a baseline for our experiments. We annotate this model as `SIMPLE` subsequently in this paper.

4 Multi-turn Reasoning Network with Multiple Type Memories

The attention mechanism described in Equations 8 and 9 is performed in a single-turn feed-forward fashion. However, for complex context and complex queries, human readers often revisit the given context in order to perform deeper inference after one turn reading. This real-world reading phenomenon motivated the multi-turn reasoning networks for document comprehension (Shen et al., 2016). Considering dialog generation scenario with given rich context, we intuitively think if the attentions can be performed multi-turns so that the conversation session is better understood and the simple query, which frequently omits unknown number of context-sensitive words, can be extended for a better generation of the response. The domain adaptation from document comprehension to dialog generation is feasible by taking the rich context of the speakers as a "document", current user's query as a "question" and the chatbot's response as an "answer".

However, there are still several major challenges for this domain adaptation. First, a document is frequently written by a single author with one (hidden) personality, one writing style, and one distribution of the engagement rates of the topics appearing in that document. These are not the case for conversation scenario in which at least two speakers are involved with different (hidden) personalities, personalized speaking styles, and diverse engagement rate distributions of the topics in that conversation session. Second, for document comprehension, the output is frequently a single named entity (Shen et al., 2016) and thus a single softmax function can satisfy this one-shot ranking problem. However, we will need a RNN decoder utilizing context vectors for generating the target response sentence being a sequence of tokens (words or characters) instead of one single named entity.

We tackle the first challenge by separating the context into multiple type memories upon which attention models are performed. For the second difference, we replace the simple softmax output layer by a GRU decoder employing reasoning-attention context vectors.

4.1 Separation of contextual information

The `SIMPLE` model can use multiple turns of context to infer the response by concatenating them

during decoding, using a separator symbol such as EOS for end-of-sentence. Sordoni et al. (2015) separated the query message and the previous two context messages when conditioning the response. The previous context messages were concatenated and treated as a single message.

In our proposed models, we use more than three turns for the context. We separate the last message (the query) from the previous turns to produce a set of annotations h, one per character[1] in the sentence. While encoding the contextual information, we separate the m_i from each speaker into two sets. The motivation is to capture individual characteristics such as personalized topical information and speaking style (Li et al., 2016b). We refer to the set of annotations from the same speaker as one memory. That is, the sentences for which the probabilities are being predicted as M_r (response memory, specially corresponds to the chatbot's side) and the question set as M_q (query memory, specially corresponds to the user's side). We further apply a RNN on top of m_i to produce one more set of vectors M_c (context memory):

$$M_c = \bigcup_{i=0}^{T_c} \{m_i^{(c)}\}, \tag{12}$$

in which,

$$m_i^{(c)} = \text{RNN}(m_i, m_{i-1}^{(c)}), \tag{13}$$

where T_c is the number of turns (sentences) in the conversation. The initial state $m_0^{(c)}$ is a trainable parameter.

We apply an attention mechanism on each of the memories M_q, M_r, M_c and M_h (of current query) separately. Refer to Figure 1 for an intuitive illustration of these memories. Following (Shen et al., 2016), we choose projected cosine similarity function as the attention module. The attention score $a_{j,i}^q$ on memory $m_i^q \in M_q$ for a RNN hidden state t_j in the decoder part is computed as follows:

$$a_{j,i}^q = \text{softmax}_{i=1,\ldots,|M_q|} \cos(W_1^q m_i^q, W_2^q t_j), \tag{14}$$

where W_1^q and W_2^q are weight vectors associated with m_i^q and t_j, respectively. Consequently, the attention vector on the query sequences is given by:

$$c_j^q = \sum_{i=1}^{|M_q|} a_{j,i}^q m_i^q. \tag{15}$$

[1]Most Japanese characters, such as Kanji, have independent semantic meanings other than English letters

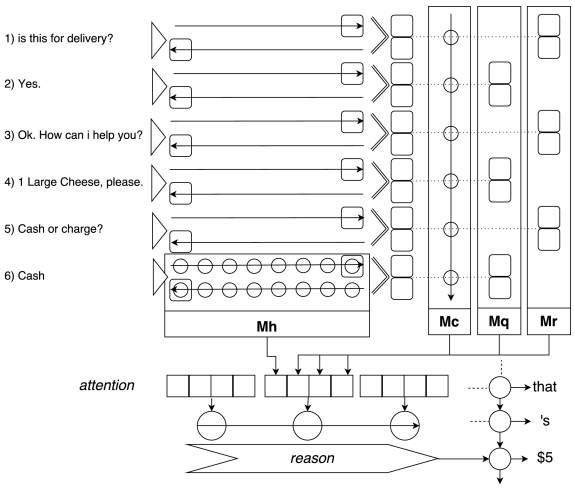


Figure 1: Illustration of the architecture of our REASON model. In the example conversation session, six context sentences are encoded independently by a biRNN. The character-specific annotations from the sixth sentence (i.e., current query) compose the only *sentence-level* M_h memory. $M_{r,q,c}$ are *conversation-level* memories. The last m_i from each sentence are distributed alternately in M_q and M_r memories. Also, $m_{1,\ldots,6}$ are iterated sequentially by a single-direction RNN to produce six context annotations in M_c using Equation 13. In the bottom of the diagram, a reasoning module (Figure 2) is used instead of the plain attention used in MULTI.

Similarly, the attention scores and attention vectors on the other three memories can be derived by replacing q with r, c, and h in Equations 14, 15.

We then concatenate these resulting attention vectors into a final context vector c_j^M, which is consequently applied to Equations 8 and 9. Since the dimension of the updated context vector c_j^M is four times larger, its weight matrix C will need to be enlarged with a same column dimension with the dimension of c_j^M so that Cc_j^M still aligns with the dimension of the hidden layer vector t_j. More details of the GRU function style definition of t_j using c_j can be found in (Bahdanau et al., 2015). We refer to this model that integrates multiple types of memories through separated attention mechanisms as MULTI.

Note that, by separately embedding conversation context into multiple type memories following the number of speakers, we can easily extend this two speaker scenario into group chatting in which tens or hundreds of speakers can be engaged in. The only extension is to further separate M_q by speakers. Consequently, the context vector can be concatenated using the attention vectors by read-

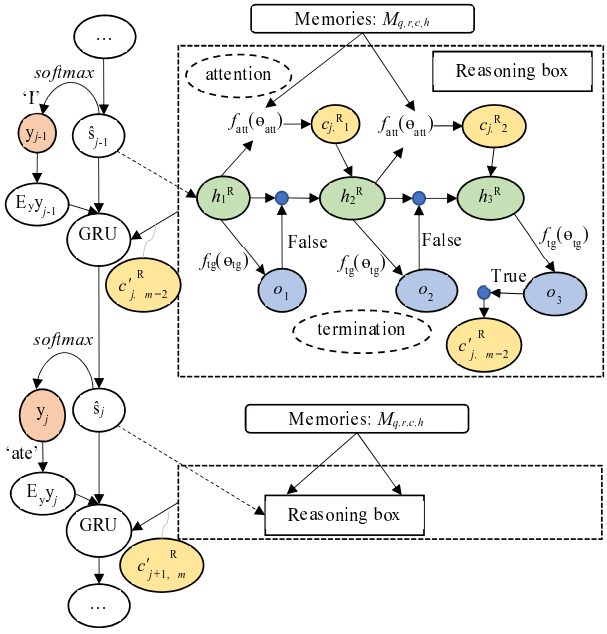


Figure 2: Illustration of the *reason* part of the REASON model. This figure is drawn by partially referring Figure 1 in (Shen et al., 2016).

ing all the memories. The theoretical benefit is that the chatbot can softly keep track of each individual speaker's topics and then make a decision of how to response to that speaker. Another extension will be using a reinforcement learning policy to determine when to let the chatbot to give a response to which speaker in current group chatting.

Generally, the number and type of memories can be enlarged in a reasonable way, such as by introducing external knowledge (Yih and Ma, 2016; Ghazvininejad et al., 2017; Han et al., 2015) or performing sentiment analysis to the "fact memories" to yield emotional memories (Zhou et al., 2017). A detailed description and experimental testifying is out of the scope of this paper.

4.2 Reasoning Neural Dialogue System

As illustrated in Figure 2, we apply a multi-turn reasoning mechanism, following Shen et al. (2016), to the multiple-type annotation memories. This reasoning mechanism replaces the single-turn attention mechanism. We adapt the idea of using a termination state during the inference to dynamically determine how many turns to reason. The termination module can decide whether to continue to infer the next turn (of re-reading the four types of memories) after digesting intermediate topical and speaker-specific information, or to terminate the whole inference process when it concludes that existing information is sufficient to generate the next word in a response. Generally, the idea is to construct a reasoning attention vector that works as a context vector during generating the next word. This idea is included in the "Reasoning box" in Figure 2. Specially, y_{j-1} stands for a former word generated by the hidden state $\hat{s}_{j-1}$ in the GRU decoder. E_y is the embedding matrix. We use a full-connection layer to map from $\hat{s}_{j-1}$ to the initial reasoning hidden state h_1^{R}, since h_m^{R} should be with the same length alike each memory vector in $M_{q,r,c,h}$ and $\hat{s}_{j-1}$'s dimension is smaller than that. Thus, (1) outside the "reasoning box", we use a GRU decoder to yield $\hat{s}_j$ so that a next word y_j can be generated, and (2) inside the "reasoning box", we read the memories to yield the "optimal" contextual vector. The "reasoning box" takes the memories $M_{q,r,c,h}$ and $\hat{s}_{j-1}$ as inputs and finally outputs $c_{j,m}^{\mathrm{R}}$.

The number of reasoning turns for yielding the "reasoning attention vectors" (c_j^{R} which is further indexed by reasoning steps of 1, 2 in Figure 2) during the decoding inference is dynamically parameterized by both the contextual memories and current query, and is generally related to the complexities of the conversation context and current query.

The training steps are performed as per the general framework as described in Equations 8 and 9. For each reasoning hidden state h_m^{R}, the termination probability o_m is estimated by $f_{tg}(h_m^{\mathrm{R}}; \theta_{tg})$, which is

$$o_m = (1 - o_{m-1}) * \sigma(w_t^\top h_m^{\mathrm{R}} + b_t), \qquad (16)$$

where $\theta_{tg} = \{w_t, b_t\}$, w_t is a weight vector, b_t is a bias, and σ is the sigmoid logistic function. Then, different hidden states h_m^{R} are first weighted by their termination probabilities o_m and then summed to produce a reasoning-attention context vector c_j^{R} (using the equations as described previously in Section 3.2), which is consequently used to construct the next reasoning step's h_2^{R} = RNN(h_1^{R}, $c_{j,1}^{\mathrm{R}}$). The final $c_{j,m}^{\mathrm{R}}$ ($m \geq 1$ is the final reasoning step) will be used in Equations 8 and 9 in a way alike former attention vectors. During our experiments, we instead used a sum of from $o_2 \times c_{j,1}^{\mathrm{R}}$ to $o_{m+1} \times c_{j,m}^{\mathrm{R}}$ as the final $c_{j,m}^{\prime\mathrm{R}}$ for next word generation.

During generating each word in the response, our network performs a response action r_m at the m-th step, which implies that the termination gate variables $o_{1:m} = (o_1 = 0, o_2 =$

```
A: i'm bored
B: boooring
A: isn't it? (^_^)
B: everybody is asleep
A: really?
   (^_^) my friends are still awake! :P
B: lucky!
   xD feels lonely with everyone asleep
A: almost everyone is awake (^v^)
B: whyyy?!
   my friends go early to bed.
   Or is it me that's late? xD
A: it's us who are late! xD
B: true, very true xD
```

Figure 3: A 5-round conversation of speakers A and B.

	Conversation sessions	Sentences	Characters (Unique)
Train	253K	2.5M	24M (6,214)
Validation	100	1K	10K (836)
Test	100	1K	9.3K (780)

Table 1: Statistics of the filtered datasets.

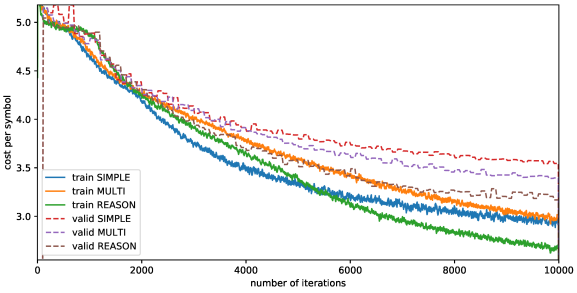


Figure 4: Cost curves of NLL during training.

$0, ..., o_{m-1} = 0, o_m = 1$). A stochastic policy $\pi((o_m, r_m)|h^{\mathrm{R}}_m, t_j; \theta)$ with parameters θ to get a distribution of termination actions, to continue reading the conversation context (i.e., $M_{q,r,c,h}$) or to stop, and of response actions r_m for predicting the next word if the model decides to stop at current step. In our experiments, we set a maximum step parameter T_{max} to be 5 for heuristically avoiding too many reasoning times. We follow (Shen et al., 2016) to compute the expected reward and its gradient for one instance. We refer to this model with multi-turn reasoning attentions as REASON.

5 Experiments

In our experiments, we used a dataset consisting of Japanese twitter conversations. Each conversation contains 10 sentences from two real-world alternating speakers. Given the origin of the dataset, it is quite noisy, containing misspelled words, slang and *kaomoji* (multi-character sequences of facial emoticons) among meaningful words and characters. Preliminary experiments by using a word-based approach resulted in the vocabulary size being too big and with too many word breaking errors, we instead used a character-based approach. Figure 3 shows a sample 10-sentence conversation in which original Japanese sentences were translated into English and similar spelling patterns were kept in a sense (such as booooring for boring and whyyy for why).

We kept the conversations in which all sentences were no more than 20 characters. This filtering strategy resulted in a dataset of 254K conversations from which 100 (1K sentences) where taken out for testing and another 100 for validating and hyper-parameter tuning. The training set contains 6,214 unique characters, which are used as our vocabulary with the addition of two special symbols, an UNK (out-of-vocabulary unknown word) and an EOS (end-of-sentence). Table 1 shows major statistics of the dataset.

The training minimizes negative log-likelihood (NLL) per character on the nine sentences $s_{2,...,10}$ of each conversation. One configuration in MULTI and REASON is that, we respectively use the reference contexts (instead of former automatically generated sentences) to generate current sentence. That is, when generating s_i, we use the golden contextual sentences of from s_1 to s_{i-1}. These three systems were respectively trained 3 epochs (10,000 iterations) on an AdaDelta (Zeiler, 2012) optimizer. Character embedding matrix was shared by both the encoder and the decoder parts. All the hidden layers, in the encoding/decoding parts and the attention models, were of size 200 and the character embeddings were of size 100. The recurrent units that we used were GRU. The gradients were clipped at the maximum gradient norm of 1. The reasoning module's maximum steps $T_{\max}$ was set to be 5. The data was iterated on mini-batches of less than 1,500 symbols each.

We initialized the recurrent weight matrices in GRUs as random orthogonal matrices. Unless specially mentioned, all the elements of the 0-indexed vectors and all bias vectors were initialized to be zero. Any other weight matrices were initialized by sampling from the Gaussian distribution of mean 0 and variance 0.01.

Figure 4 shows the progression of the NLLs per

		SIMPLE	MULTI	REASON
BLEU-4	Train	1.98	1.97	**2.30**
	Validation	1.80	2.12	**2.62**
	Test	2.20	2.13	**2.89**
BLEU-2	Train	6.77	6.78	**7.03**
	Validation	6.67	6.89	**8.14**
	Test	7.19	7.24	**7.97**

Table 2: Character-level BLEU-2/4 (%) scores.

character during training. The validation costs begun converging in the third epoch for the three models. The plot roughly shows lower cost for more complex models.

Galley et al. (2015) obtained better correlation with human evaluation when using BLEU-2 rather than BLEU-4. We thus report both of these scores for automatic evaluation and comparison. The character-level BLEU-4 and BLEU-2 scores for the trained models are reported in Table 2. The REASON model achieved consistently better BLEU-2 and BLEU-4 scores in the three datasets. MULTI performed slightly better than SIMPLE on the validation set yet that performance is less stable than REASON.

Figure 4 also reflects that, (1) the final training costs of SIMPLE and MULTI are quite close with each other at iteration 10,000; (2) there is a big margin of between the final training cost of REASON and that of SIMPLE or MULTI; and (3) the validation costs exactly follows an order of SIMPLE > MULTI > REASON.

6 Analysis

Figure 5 illustrates an English translation of a conversation and the responses suggested by each of the described models. This conversation is extracted from the test set. The three responses are different from the reference response, but the one from REASON looks the most consistent with the given context. The response from MULTI is contradicting the context of speaker B as he/she said *Not at all* in a former sentence.

As it has been shown in (Liu et al., 2016) that BLEU doesn't correlate well with human judgments, we asked three human evaluators to respectively examine 900 responses from each of the models given their reference contexts. The evaluators were asked to judge (1) whether one response is acceptable and (2) whether one response is better than the other two responses. A summary of this evaluation is displayed in Table 3. The *acceptable* column refers to the percentage of responses

A: I feel nostalgic. This was so cute. B: It's gross. What's that picture? A: It's cute! It used to be on TV. B: Isn't that from a children's show? A: Yes B: I know that one!! A: Isn't it cute? B: Not at all A: Uh! I can't believe it...

SIMPLE (B:)	MULTI (B:)	REASON (B:)
Uh! it isn't.	It's cute!	Do you think it's cute?

Reference
B: Are you asking me whether the picture is cute?

Figure 5: Sample responses generated by the three models and the recorded reference response, translated to English. Both MULTI and REASON include *cute*, yet MULTI contradicts a previous response *Not at all*.

	acceptable	best-of-three †	> SIMPLE‡	> MULTI‡
SIMPLE	42%	25%	-	45%
MULTI	52%	29%	55%	-
REASON	**65%**	**46%**	**59%**	**58%**

Table 3: Human evaluation of the responses generated by the three models. † Percentage over the conversations that had at least one response accepted. ‡ From the cases where any of both compared models was acceptable. > = "better than".

that were considered acceptable by at least two of the human evaluators while the *best-of-three* columns refers to the percentage of times that each model's response was considered by at least two evaluators to be better than the other two's, from the contexts that had at least one acceptable response. The last two columns make one-to-one comparisons. In 18% of the contexts, none of the models produced an acceptable response.

This human evaluation shows that complexer models are more likely to produce acceptable responses. The MULTI and REASON models are only different in the attention mechanism of multi-turn reasoning. The reasoning module performed better than single-turn attention 58% of the times.

Table 4 contains the character-level *distinct-n* (Li et al., 2016a) metrics for n-grams where $1 \leq n \leq 5$. This metric measures the number of distinct n-grams divided by the total number of n-grams in the generated responses. The displayed results are computed on the concatenation of all the responses to the test-set contexts. The *Reference* column was computed on the reference responses and represents the optimal human-like ratio.

SIMPLE performed the best at uni-gram diver-

	SIMPLE	MULTI	REASON	Reference
distinct-1	**.039**	.028	.032	.088
distinct-2	.112	.095	**.121**	.407
distinct-3	.199	.180	**.238**	.588
distinct-4	.248	.241	**.310**	.587
distinct-5	.255	.265	**.328**	.530

Table 4: N-gram diversity metrics of between (1) the responses generated to the test set and (2) their reference responses.

sity. For n-grams $n \geq 2$, REASON produced the most diverse outputs. While the results for REASON were consistently better than the other two models, the results for MULTI were not always better than SIMPLE. This indicates MULTI does not always benefit from the augmented context without the multi-turn reasoning attentions.

7 Conclusions

We have presented a novel approach to multi-turn conversation modeling. Our approach uses multiple explicitly separated memories to represent rich conversational contexts. We also presented multi-turn reasoning attentions to integrate various annotation memories. We run experiments on three different models with and without the introduced approaches and measured their performances using automatic metrics and human evaluation.

Experimental results verified that the increased contexts are able to help producing more acceptable and diverse responses. Driven by the depth of the reasoning attention, the diversities of the responses are significantly improved. We argue that the reasoning attention mechanism helps integrating the multiple pieces of information as it can combine them in a more complex way than a simple weighted sum. We further observed that as the accuracy of the conversation model improves, the diversity of the generated responses increases.

The proposed approach of multi-turn reasoning over multiple memory attention networks is presented in a general framework that allows the inclusion of memories of multiple resources and types. Applying to group chatting with more than two speakers and reasoning over emotion embeddings or knowledge vectors included from an external knowledge base/graph are taken as our future directions.

Acknowledgments

The authors thank the anonymous reviewers for their impressive comments and suggestions for improving the earlier version.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv:1409.0473 [cs, stat]* ArXiv: 1409.0473. http://arxiv.org/abs/1409.0473.

Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. 2015. End-to-End Attention-based Large Vocabulary Speech Recognition. *arXiv:1508.04395 [cs]* ArXiv: 1508.04395. http://arxiv.org/abs/1508.04395.

Rafael E. Banchs and Haizhou Li. 2012. IRIS: a Chat-oriented Dialogue System based on the Vector Space Model. In *Proceedings of the ACL 2012 System Demonstrations*. Association for Computational Linguistics, Jeju Island, Korea, pages 37–42. http://www.aclweb.org/anthology/P12-3007.

William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals. 2015. Listen, Attend and Spell. *arXiv:1508.01211 [cs, stat]* ArXiv: 1508.01211. http://arxiv.org/abs/1508.01211.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv:1406.1078 [cs, stat]* ArXiv: 1406.1078. http://arxiv.org/abs/1406.1078.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2016. Gated-Attention Readers for Text Comprehension. *arXiv:1606.01549 [cs]* ArXiv: 1606.01549. http://arxiv.org/abs/1606.01549.

Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term Recurrent Convolutional Networks for Visual Recognition and Description. In *CVPR*.

Michel Galley, Chris Brockett, Alessandro Sordoni, Yangfeng Ji, Michael Auli, Chris Quirk, Margaret Mitchell, Jianfeng Gao, and Bill Dolan. 2015. deltaBLEU: A Discriminative Metric for Generation Tasks with Intrinsically Diverse Targets. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Association for Computational Linguistics, Beijing, China, pages 445–450. http://www.aclweb.org/anthology/P15-2073.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. 2017. A knowledge-grounded neural conversation model. *CoRR* abs/1702.01932. http://arxiv.org/abs/1702.01932.

Sangdo Han, Jeesoo Bang, Seonghan Ryu, and Gary Geunbae Lee. 2015. Exploiting knowledge base to generate responses for natural language dialog listening agents. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, Prague, Czech Republic, pages 129–133. http://aclweb.org/anthology/W15-4616.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The Goldilocks Principle: Reading Children's Books with Explicit Memory Representations. *arXiv:1511.02301 [cs]* ArXiv: 1511.02301. http://arxiv.org/abs/1511.02301.

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

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A Diversity-Promoting Objective Function for Neural Conversation Models. *arXiv:1510.03055 [cs]* ArXiv: 1510.03055. http://arxiv.org/abs/1510.03055.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A Diversity-Promoting Objective Function for Neural Conversation Models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 110–119. http://www.aclweb.org/anthology/N16-1014.

Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016b. A Persona-Based Neural Conversation Model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 994–1003. http://www.aclweb.org/anthology/P16-1094.

Jiwei Li, Will Monroe, Tianlin Shi, Sbastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial Learning for Neural Dialogue Generation. *arXiv:1701.06547 [cs]* ArXiv: 1701.06547. http://arxiv.org/abs/1701.06547.

Jiwei Li, Alan Ritter, and Eduard Hovy. 2014. Weakly supervised user profile extraction from twitter. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 165–174. http://www.aclweb.org/anthology/P14-1016.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT To Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 2122–2132. https://aclweb.org/anthology/D16-1230.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems. *arXiv:1506.08909 [cs]* ArXiv: 1506.08909. http://arxiv.org/abs/1506.08909.

Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. 2015. Deep Captioning with Multimodal Recurrent Neural Networks (m-RNN). *ICLR* .

Alan Ritter, Colin Cherry, and William B. Dolan. 2011. Data-Driven Response Generation in Social Media. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Edinburgh, Scotland, UK., pages 583–593. http://www.aclweb.org/anthology/D11-1054.

M. Schuster and K.K. Paliwal. 1997. Bidirectional Recurrent Neural Networks. *Trans. Sig. Proc.* 45(11):2673–2681. https://doi.org/10.1109/78.650093.

Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building End-to-end Dialogue Systems Using Generative Hierarchical Neural Network Models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press, Phoenix, Arizona, AAAI'16, pages 3776–3783. http://dl.acm.org/citation.cfm?id=3016387.3016435.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural Responding Machine for Short-Text Conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 1577–1586. http://www.aclweb.org/anthology/P15-1152.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2016. ReasoNet: Learning to Stop Reading in Machine Comprehension. *arXiv:1609.05284 [cs]* ArXiv: 1609.05284. https://doi.org/10.1145/3097983.3098177.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015.

A Neural Network Approach to Context-Sensitive Generation of Conversational Responses. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Denver, Colorado, pages 196–205. http://www.aclweb.org/anthology/N15-1020.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, Curran Associates, Inc., pages 3104–3112.

Oriol Vinyals, ukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a Foreign Language. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, Curran Associates, Inc., pages 2773–2781. http://papers.nips.cc/paper/5635-grammar-as-a-foreign-language.pdf.

Oriol Vinyals and Quoc Le. 2015. A Neural Conversational Model. *arXiv:1506.05869 [cs]* ArXiv: 1506.05869. http://arxiv.org/abs/1506.05869.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, and Steve Young. 2016. Multi-domain Neural Network Language Generation for Spoken Dialogue Systems. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 120–129. http://www.aclweb.org/anthology/N16-1015.

Wen-tau Yih and Hao Ma. 2016. Question Answering with Knowledge Base, Web and Beyond. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorial Abstracts*. Association for Computational Linguistics, San Diego, California, pages 8–10. http://www.aclweb.org/anthology/N16-4003.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2016. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. *arXiv:1609.05473 [cs]* ArXiv: 1609.05473. http://arxiv.org/abs/1609.05473.

Matthew D. Zeiler. 2012. ADADELTA: An Adaptive Learning Rate Method. *arXiv:1212.5701 [cs]* ArXiv: 1212.5701. http://arxiv.org/abs/1212.5701.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2017. Emotional chatting machine: Emotional conversation generation with internal and external memory. *CoRR* abs/1704.01074. http://arxiv.org/abs/1704.01074.

Dialogue Learning with Human Teaching and Feedback in End-to-End Trainable Task-Oriented Dialogue Systems

Bing Liu[1]*, **Gokhan Tür**[2], **Dilek Hakkani-Tür**[2], **Pararth Shah**[2], **Larry Heck**[3]†
[1]Carnegie Mellon University, Pittsburgh, PA, USA
[2]Google Research, Mountain View, CA,USA [3]Samsung Research, Mountain View, CA, USA
liubing@cmu.edu, {dilekh,pararth}@google.com,
{gokhan.tur,larry.heck}@ieee.org

Abstract

In this work, we present a hybrid learning method for training task-oriented dialogue systems through online user interactions. Popular methods for learning task-oriented dialogues include applying reinforcement learning with user feedback on supervised pre-training models. Efficiency of such learning method may suffer from the mismatch of dialogue state distribution between offline training and online interactive learning stages. To address this challenge, we propose a hybrid imitation and reinforcement learning method, with which a dialogue agent can effectively learn from its interaction with users by learning from human teaching and feedback. We design a neural network based task-oriented dialogue agent that can be optimized end-to-end with the proposed learning method. Experimental results show that our end-to-end dialogue agent can learn effectively from the mistake it makes via imitation learning from user teaching. Applying reinforcement learning with user feedback after the imitation learning stage further improves the agent's capability in successfully completing a task.

1 Introduction

Task-oriented dialogue systems assist users to complete tasks in specific domains by understanding user's request and aggregate useful information from external resources within several dialogue turns. Conventional task-oriented dialogue systems have a complex pipeline (Rudnicky et al., 1999; Raux et al., 2005; Young et al., 2013) consisting of independently developed and modularly connected components for natural language understanding (NLU) (Mesnil et al., 2015; Liu and Lane, 2016; Hakkani-Tür et al., 2016), dialogue state tracking (DST) (Henderson et al., 2014c; Mrkšić et al., 2016), and dialogue policy learning (Gasic and Young, 2014; Shah et al., 2016; Su et al., 2016, 2017). These system components are usually trained independently, and their optimization targets may not fully align with the overall system evaluation criteria (e.g. task success rate and user satisfaction). Moreover, errors made in the upper stream modules of the pipeline propagate to downstream components and get amplified, making it hard to track the source of errors.

To address these limitations with the conventional task-oriented dialogue systems, recent efforts have been made in designing end-to-end learning solutions with neural network based methods. Both supervised learning (SL) based (Wen et al., 2017; Bordes and Weston, 2017; Liu and Lane, 2017a) and deep reinforcement learning (RL) based systems (Zhao and Eskenazi, 2016; Li et al., 2017; Peng et al., 2017) have been studied in the literature. Comparing to chit-chat dialogue models that are usually trained offline using single-turn context-response pairs, task-oriented dialogue model involves reasoning and planning over multiple dialogue turns. This makes it especially important for a system to be able to learn from users in an interactive manner. Comparing to SL models, systems trained with RL by receiving feedback during users interactions showed improved model robustness against diverse dialogue scenarios (Williams and Zweig, 2016; Liu and Lane, 2017b).

A critical step in learning RL based task-oriented dialogue models is dialogue policy learning. Training dialogue policy online from scratch typically requires a large number of interactive learning sessions before an agent can reach a satisfactory performance level. Recent works (Henderson et al., 2008; Williams et al., 2017; Liu et al., 2017) explored pre-training the dialogue model using human-human or human-machine dialogue

* Work done while the author was an intern at Google.
† Work done while at Google Research.

Proceedings of NAACL-HLT 2018, pages 2060–2069
New Orleans, Louisiana, June 1 - 6, 2018.

corpora before performing interactive learning with RL to address this concern. A potential drawback with such pre-training approach is that the model may suffer from the mismatch of dialogue state distributions between supervised training and interactive learning stages. While interacting with users, the agent's response at each turn has a direct influence on the distribution of dialogue state that the agent will operate on in the upcoming dialogue turns. If the agent makes a small mistake and reaches an unfamiliar state, it may not know how to recover from it and get back to a normal dialogue trajectory. This is because such recovery situation may be rare for good human agents and thus are not well covered in the supervised training corpus. This will result in compounding errors in a dialogue which may lead to failure of a task. RL exploration might finally help to find corresponding actions to recover from a bad state, but the search process can be very inefficient.

To ameliorate the effect of dialogue state distribution mismatch between offline training and RL interactive learning, we propose a hybrid imitation and reinforcement learning method. We first let the agent to interact with users using its own policy learned from supervised pre-training. When an agent makes a mistake, we ask users to correct the mistake by demonstrating the agent the right actions to take at each turn. This user corrected dialogue sample, which is guided by the agent's own policy, is then added to the existing training corpus. We fine-tune the dialogue policy with this dialogue sample aggregation (Ross et al., 2011) and continue such user teaching process for a number of cycles. Since asking for user teaching at each dialogue turn is costly, we want to reduce this user teaching cycles as much as possible and continue the learning process with RL by collecting simple forms of user feedback (e.g. a binary feedback, positive or negative) only at the end of a dialogue.

Our main contributions in this work are:

- We design a neural network based task-oriented dialogue system which can be optimized end-to-end for natural language understanding, dialogue state tracking, and dialogue policy learning.
- We propose a hybrid imitation and reinforcement learning method for end-to-end model training in addressing the challenge with dialogue state distribution mismatch between offline training and interactive learning.

The remainder of the paper is organized as follows. In section 2, we discuss related work in building end-to-end task-oriented dialogue systems. In section 3, we describe the proposed model and learning method in detail. In Section 4, we describe the experiment setup and discuss the results. Section 5 gives the conclusions.

2 Related Work

Popular approaches in learning task-oriented dialogue include modeling the task as a partially observable Markov Decision Process (POMDP) (Young et al., 2013). RL can be applied in the POMDP framework to learn dialogue policy online by interacting with users (Gašić et al., 2013). The dialogue state and system action space have to be carefully designed in order to make the policy learning tractable (Young et al., 2013), which limits the model's usage to restricted domains.

Recent efforts have been made in designing end-to-end solutions for task-oriented dialogues, inspired by the success of encoder-decoder based neural network models in non-task-oriented conversational systems (Serban et al., 2015; Li et al., 2016). Wen et al. (Wen et al., 2017) designed an end-to-end trainable neural dialogue model with modularly connected system components. This system is a supervised learning model which is evaluated on fixed dialogue corpora. It is unknown how well the model performance generalizes to unseen dialogue state during user interactions. Our system is trained by a combination of supervised and deep RL methods, as it is shown that RL may effectively improve dialogue success rate by exploring a large dialogue action space (Henderson et al., 2008; Li et al., 2017).

Bordes and Weston (2017) proposed a task-oriented dialogue model using end-to-end memory networks. In the same line of research, people explored using query-regression networks (Seo et al., 2016), gated memory networks (Liu and Perez, 2017), and copy-augmented networks (Eric and Manning, 2017) to learn the dialogue state. These systems directly select a final response from a list of response candidates conditioning on the dialogue history without doing slot filling or user goal tracking. Our model, on the other hand, explicitly tracks user's goal for effective integration with knowledge bases (KBs). Robust dialogue state tracking has been shown (Jurčíček et al., 2012) to

be critical in improving dialogue success in task completion.

Dhingra et al. (2017) proposed an end-to-end RL dialogue agent for information access. Their model focuses on bringing differentiability to the KB query operation by introducing a "soft" retrieval process in selecting the KB entries. Such soft-KB lookup is prone to entity updates and additions in the KB, which is common in real world information systems. In our model, we use symbolic queries and leave the selection of KB entities to external services (e.g. a recommender system), as entity ranking in real world systems can be made with much richer features (e.g. user profiles, location and time context, etc.). Quality of the generated symbolic query is directly related to the belief tracking performance. In our proposed end-to-end system, belief tracking can be optimized together with other system components (e.g. language understanding and policy) during interactive learning with users.

Williams et al. (2017) proposed a hybrid code network for task-oriented dialogue that can be trained with supervised and reinforcement learning. They show that RL performed with a supervised pre-training model using labeled dialogues improves learning speed dramatically. They did not discuss the potential issue of dialogue state distribution mismatch between supervised pre-training and RL interactive learning, which is addressed in our dialogue learning framework.

3 Proposed Method

Figure 1 shows the overall system architecture of the proposed end-to-end task-oriented dialogue model. We use a hierarchical LSTM neural network to encode a dialogue with a sequence of turns. User input to the system in natural language format is encoded to a continuous vector via a bidirectional LSTM utterance encoder. This user utterance encoding, together with the encoding of the previous system action, serves as the input to a dialogue-level LSTM. State of this dialogue-level LSTM maintains a continuous representation of the dialogue state. Based on this state, the model generates a probability distribution over candidate values for each of the tracked goal slots. A query command can then be formulated with the state tracking outputs and issued to a knowledge base to retrieve requested information. Finally, the system produces a dialogue action, which is conditioned on information from the dialogue state, the estimated user's goal, and the encoding of the query results . This dialogue action, together with the user goal tracking results and the query results, is used to generate the final natural language system response via a natural language generator (NLG). We describe each core model component in detail in the following sections.

3.1 Utterance Encoding

We use a bidirectional LSTM to encode the user utterance to a continuous representation. We refer to this LSTM as the utterance-level LSTM. The user utterance vector is generated by concatenating the last forward and backward LSTM states. Let $\mathbf{U}_k = (w_1, w_2, ..., w_{T_k})$ be the user utterance at turn k with T_k words. These words are firstly mapped to an embedding space, and further serve as the step inputs to the bidirectional LSTM. Let $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ represent the forward and backward LSTM state outputs at time step t. The user utterance vector U_k is produced by: $U_k = [\overrightarrow{h_{T_k}}, \overleftarrow{h_1}]$, where $\overrightarrow{h_{T_k}}$ and $\overleftarrow{h_1}$ are the last states in the forward and backward LSTMs.

3.2 Dialogue State Tracking

Dialogue state tracking, or belief tracking, maintains the state of a conversation, such as user's goals, by accumulating evidence along the sequence of dialogue turns. Our model maintains the dialogue state in a continuous form in the dialogue-level LSTM ($\mathrm{LSTM_D}$) state s_k. s_k is updated after the model processes each dialogue turn by taking in the encoding of user utterance U_k and the encoding of the previous turn system output A_{k-1}. This dialogue state serves as the input to the dialogue state tracker. The tracker updates its estimation of the user's goal represented by a list of slot-value pairs. A probability distribution $P(l_k^m)$ is maintained over candidate values for each goal slot type $m \in M$:

$$s_k = \mathrm{LSTM_D}(s_{k-1},\ [U_k,\ A_{k-1}]) \quad (1)$$

$$P(l_k^m \mid \mathbf{U}_{\leq k},\ \mathbf{A}_{<k}) = \mathrm{SlotDist}_m(s_k) \quad (2)$$

where $\mathrm{SlotDist}_m$ is a single hidden layer MLP with $\mathrm{softmax}$ activation over slot type $m \in M$.

3.3 KB Operation

The dialogue state tracking outputs are used to form an API call command to retrieve information from a knowledge base. The API call command is

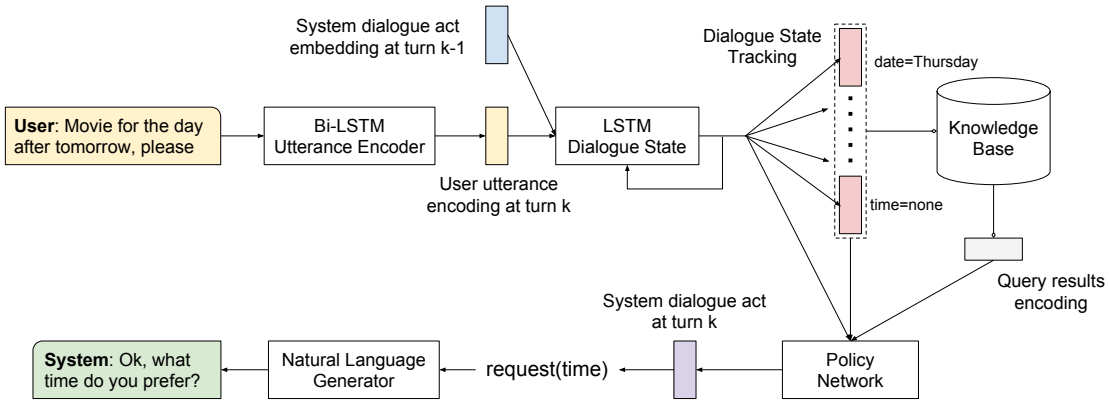


Figure 1: Proposed end-to-end task-oriented dialogue system architecture.

produced by replacing the tokens in a query command template with the best hypothesis for each goal slot from the dialogue state tracking output. Alternatively, an n-best list of API calls can be generated with the most probable candidate values for the tracked goal slots. In interfacing with KBs, instead of using a soft KB lookup as in (Dhingra et al., 2017), our model sends symbolic queries to the KB and leaves the ranking of the KB entities to an external recommender system. Entity ranking in real world systems can be made with much richer features (e.g. user profiles, local context, etc.) in the back-end system other than just following entity posterior probabilities conditioning on a user utterance. Hence ranking of the KB entities is not a part of our proposed neural dialogue model. In this work, we assume that the model receives a ranked list of KB entities according to the issued query and other available sources, such as user models.

Once the KB query results are returned, we save the retrieved entities to a queue and encode the result summary to a vector. Rather then encoding the real KB entity values as in (Bordes and Weston, 2017; Eric and Manning, 2017), we only encode a summary of the query results (i.e. item availability and number of matched items). This encoding serves as a part of the input to the policy network.

3.4 Dialogue Policy

A dialogue policy selects the next system action in response to the user's input based on the current dialogue state. We use a deep neural network to model the dialogue policy. There are three inputs to the policy network, (1) the dialogue-level LSTM state s_k, (2) the log probabilities of candidate values from the belief tracker v_k, and (3) the

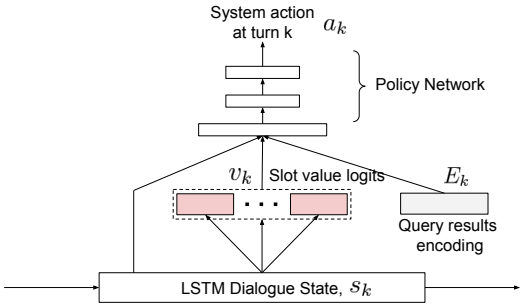


Figure 2: Dialogue state and policy network.

encoding of the query results summary E_k. The policy network emits a system action in the form of a dialogue act conditioning on these inputs:

$$P(a_k \mid U_{\leq k},\ A_{<k},\ E_{\leq k}) = \text{PolicyNet}(s_k, v_k, E_k) \quad (3)$$

where v_k represents the concatenated log probabilities of candidate values for each goal slot, E_k is the encoding of query results, and PolicyNet is a single hidden layer MLP with softmax activation function over all system actions.

The emitted system action is finally used to produce a system response in natural language format by combining the state tracker outputs and the retrieved KB entities. We use a template based NLG in this work. The delexicalised tokens in the NLG template are replaced by the values from either the estimated user goal values or the KB entities, depending on the emitted system action.

3.5 Supervised Pre-training

By connecting all the system components, we have an end-to-end model for task-oriented dialogue. Each system component is a neural network that takes in underlying system component's outputs

in a continuous form that is fully differentiable, and the entire system (utterance encoding, dialogue state tracking, and policy network) can be trained end-to-end.

We first train the system in a supervised manner by fitting task-oriented dialogue samples. The model predicts the true user goal slot values and the next system action at each turn of a dialogue. We optimize the model parameter set θ by minimizing a linear interpolation of cross-entropy losses for dialogue state tracking and system action prediction:

$$\min_{\theta} \sum_{k=1}^{K} -\Big[\sum_{m=1}^{M} \lambda_{l^m} \log P(l_k^{m*} | \mathbf{U}_{\leq k}, \mathbf{A}_{<k}, \mathbf{E}_{<k}; \theta) + \lambda_a \log P(a_k^* | \mathbf{U}_{\leq k}, \mathbf{A}_{<k}, \mathbf{E}_{\leq k}; \theta) \Big] \quad (4)$$

where λs are the linear interpolation weights for the cost of each system output. l_k^{m*} is the ground truth label for the tracked user goal slot type $m \in M$ at the kth turn, and a_k^* is the true system action in the corpus.

3.6 Imitation Learning with Human Teaching

Once obtaining a supervised training dialogue agent, we further let the agent to learn interactively from users by conducting task-oriented dialogues. Supervised learning succeeds when training and test data distributions match. During the agent's interaction with users, any mistake made by the agent or any deviation in the user's behavior may lead to a different dialogue state distribution than the one that the supervised learning agent saw during offline training. A small mistake made by the agent due to this covariate shift (Ross and Bagnell, 2010; Ross et al., 2011) may lead to compounding errors which finally lead to failure of a task. To address this issue, we propose a dialogue imitation learning method which allows the dialogue agent to learn from human teaching. We let the supervised training agent to interact with users using its learned dialogue policy $\pi_\theta(a|s)$. With this, we collect additional dialogue samples that are guided by the agent's own policy, rather than by the expert policy as those in the supervised training corpora. When the agent make mistakes, we ask users to correct the mistakes and demonstrate the expected actions and predictions for the agent to make. Such user teaching precisely addresses the limitations of the currently learned dialogue model, as these newly collected dialogue samples are driven by the agent's own policy. Specifically, in this study we let an expert user to correct the mistake made by the agent in tracking the user's goal at the end of each dialogue turn. This new batch of annotated dialogues are then added to the existing training corpus. We start the next round of supervised model training on this aggregated corpus to obtain an updated dialogue policy, and continue this dialogue imitation learning cycles.

Algorithm 1 Dialogue Learning with Human Teaching and Feedback

1: Train model end-to-end on dialogue samples D with MLE and obtain policy $\pi_\theta(a|s)$ ▷ eq 4
2: **for** learning iteration $k = 1 : K$ **do**
3: Run $\pi_\theta(a|s)$ with user to collect new dialogue samples D_π
4: Ask user to correct the mistakes in the tracked user's goal for each dialogue turn in D_π
5: Add the newly labeled dialogue samples to the existing corpora: $D \leftarrow D \cup D_\pi$
6: Train model end-to-end on D and obtain an updated policy $\pi_\theta(a|s)$ ▷ eq 4
7: **end for**
8: **for** learning iteration $k = 1 : N$ **do**
9: Run $\pi_\theta(a|s)$ with user for a new dialogue
10: Collect user feedback as reward r
11: Update model end-to-end and obtain an updated policy $\pi_\theta(a|s)$ ▷ eq 5
12: **end for**

3.7 Reinforcement Learning with Human Feedback

Learning from human teaching can be costly, as it requires expert users to provide corrections at each dialogue turn. We want to minimize the number of such imitation dialogue learning cycles and continue to improve the agent via a form of supervision signal that is easier to obtain. After the imitation learning stage, we further optimize the neural dialogue system with RL by letting the agent to interact with users and learn from user feedback. Different from the turn-level corrections in the imitation dialogue learning stage, the feedback is only collected at the end of a dialogue. A positive reward is collected for successful tasks, and a zero reward is collected for failed tasks. A step penalty is applied to each dialogue turn to encour-

age the agent to complete the task in fewer steps. In this work, we only use task-completion as the metric in designing the dialogue reward. One can extend it by introducing additional factors to the reward functions, such as naturalness of interactions or costs associated with KB queries.

To encourage the agent to explore the dialogue action space, we let the agent to follow a softmax policy during RL training by sampling system actions from the policy network outputs. We apply REINFORCE algorithm (Williams, 1992) in optimizing the network parameters. The objective function can be written as $J_k(\theta) = \mathbb{E}_\theta[R_k] = \mathbb{E}_\theta\left[\sum_{t=0}^{K-k}\gamma^t r_{k+t}\right]$, with $\gamma \in [0,1)$ being the discount factor. With likelihood ratio gradient estimator, the gradient of the objective function can be derived as:

$$\begin{aligned}\nabla_\theta J_k(\theta) &= \nabla_\theta \mathbb{E}_\theta[R_k] \\ &= \sum_{a_k} \pi_\theta(a_k|s_k)\nabla_\theta \log \pi_\theta(a_k|s_k) R_k \\ &= \mathbb{E}_\theta[\nabla_\theta \log \pi_\theta(a_k|s_k) R_k]\end{aligned} \tag{5}$$

This last expression above gives us an unbiased gradient estimator.

4 Experiments

4.1 Datasets

We evaluate the proposed method on DSTC2 (Henderson et al., 2014a) dataset in restaurant search domain and an internally collected dialogue corpus[1] in movie booking domain. The movie booking dialogue corpus has an average number of 8.4 turns per dialogue. Its training set has 100K dialogues, and the development set and test set each has 10K dialogues.

The movie booking dialogue corpus is generated (Shah et al., 2018) using a finite state machine based dialogue agent and an agenda based user simulator (Schatzmann et al., 2007) with natural language utterances rewritten by real users. The user simulator can be configured with different personalities, showing various levels of randomness and cooperativeness. This user simulator is also used to interact with our end-to-end training agent during imitation and reinforcement learning stages. We randomly select a user profile when conducting each dialogue simulation. During model evaluation, we use an extended set of natural language surface forms over the ones used during training time to evaluate the generalization capability of the proposed end-to-end model in handling diverse natural language inputs.

[1]The dataset can be accessed via https://github.com/google-research-datasets/simulated-dialogue

4.2 Training Settings

The size of the dialogue-level and utterance-level LSTM state is set as 200 and 150 respectively. Word embedding size is 300. Embedding size for system action and slot values is set as 32. Hidden layer size of the policy network is set as 100. We use Adam optimization method (Kingma and Ba, 2014) with initial learning rate of 1e-3. Dropout rate of 0.5 is applied during supervised training to prevent the model from over-fitting.

In imitation learning, we perform mini-batch model update after collecting every 25 dialogues. System actions are sampled from the learned policy to encourage exploration. The system action is defined with the act and slot types from a dialogue act (Henderson et al., 2013). For example, the dialogue act "$confirm(date = monday)$" is mapped to a system action "$confirm\_date$" and a candidate value "$monday$" for slot type "$date$". The slot types and values are from the dialogue state tracking output.

In RL optimization, we update the model with every mini-batch of 25 samples. Dialogue is considered successful based on two conditions: (1) the goal slot values estimated from dialogue state tracking fully match to the user's true goal values, and (2) the system is able to confirm with the user the tracked goal values and offer an entity which is finally accepted by the user. Maximum allowed number of dialogue turn is set as 15. A positive reward of +15.0 is given at the end of a successful dialogue, and a zero reward is given to a failed case. We apply a step penalty of -1.0 for each turn to encourage shorter dialogue for task completion.

4.3 Supervised Learning Results

Table 4.3 and Table 4.3 show the supervised learning model performance on DSTC2 and the movie booking corpus. Evaluation is made on DST accuracy. For the evaluation on DSTC2 corpus, we use the live ASR transcriptions as the user input utterances. Our proposed model achieves near state-of-the-art dialogue state tracking results on DSTC2 corpus, on both individual slot tracking and joint slot tracking, comparing to the recent published

results using RNN (Henderson et al., 2014b) and neural belief tracker (NBT) (Mrkšić et al., 2016). In the movie booking domain, our model also achieves promising performance on both individual slot tracking and joint slot tracking accuracy. Instead of using ASR hypothesis as model input as in DSTC2, here we use text based input which has much lower noise level in the evaluation of the movie booking tasks. This partially explains the higher DST accuracy in the movie booking domain comparing to DSTC2.

Model	Area	Food	Price	Joint
RNN	92	86	86	69
RNN+sem. dict	92	86	92	71
NBT	90	84	94	72
Our SL model	90	84	92	72

Table 1: Dialogue state tracking results on DSTC2

Goal slot	Accuracy
Num of Tickets	98.22
Movie	91.86
Theater Name	97.33
Date	99.31
Time	97.71
Joint	84.57

Table 2: DST results on movie booking dataset

4.4 Imitation and RL Results

Evaluations of interactive learning with imitation and reinforcement learning are made on metrics of (1) task success rate, (2) dialogue turn size, and (3) DST accuracy. Figures 3, 4, and 5 show the learning curves for the three evaluation metrics. In addition, we compare model performance on task success rate using two different RL training settings, the end-to-end training and the policy-only training, to show the advantages of performing end-to-end system optimization with RL.

Task Success Rate As shown in the learning curves in Figure 3, the SL model performs poorly. This might largely due to the compounding errors caused by the mismatch of dialogue state distribution between offline training and interactive learning. We use an extended set of user NLG templates during interactive evaluation. Many of the test NLG templates are not seen by the supervised training agent. Any mistake made by the agent in understanding the user's request may lead to compounding errors in the following dialogue

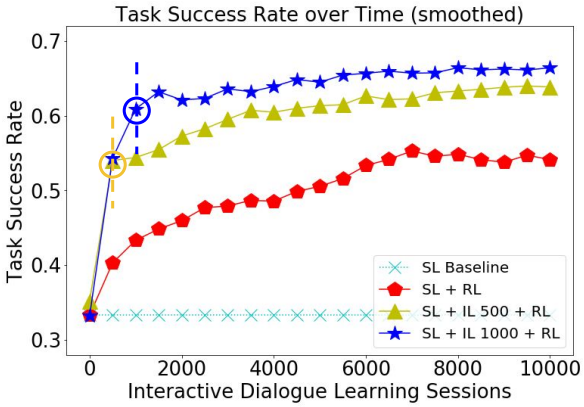


Figure 3: Interactive learning curves on task success rate.

turns, which cause final task failure. The red curve (`SL + RL`) shows the performance of the model that has RL applied on the supervised pre-training model. We can see that interactive learning with RL using a weak form of supervision from user feedback continuously improves the task success rate with the growing number of user interactions. We further conduct experiments in learning dialogue model from scratch using only RL (i.e. without supervised pre-training), and the task success rate remains at a very low level after 10K dialogue simulations. We believe that it is because the dialogue state space is too complex for the agent to learn from scratch, as it has to learn a good NLU model in combination with a good policy to complete the task. The yellow curve (`SL + IL 500 + RL`) shows the performance of the model that has 500 episodes of imitation learning over the SL model and continues with RL optimization. It is clear from the results that applying imitation learning on supervised training model efficiently improves task success rate. RL optimization after imitation learning increases the task success rate further. The blue curve (`SL + IL 1000 + RL`) shows the performance of the model that has 1000 episodes of imitation learning over the SL model and continues with RL. Similarly, it shows hints that imitation learning may effectively adapt the supervised training model to the dialogue state distribution during user interactions.

Average Dialogue Turn Size Figure 4 shows the curves for the average turn size of successful dialogues. We observe decreasing number of dialogue turns in completing a task along the growing number of interactive learning sessions. This shows that the dialogue agent learns better strategies in successfully completing the task with fewer

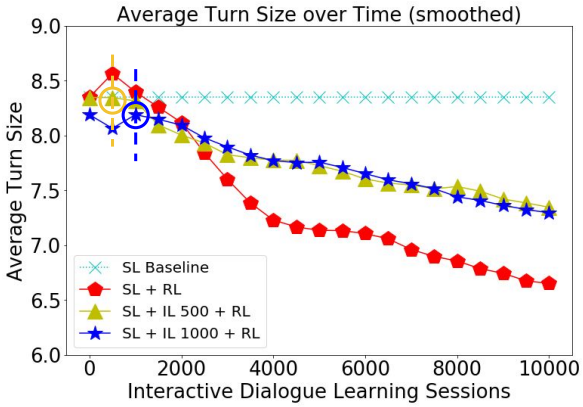


Figure 4: Interactive learning curves on average dialogue turn size.

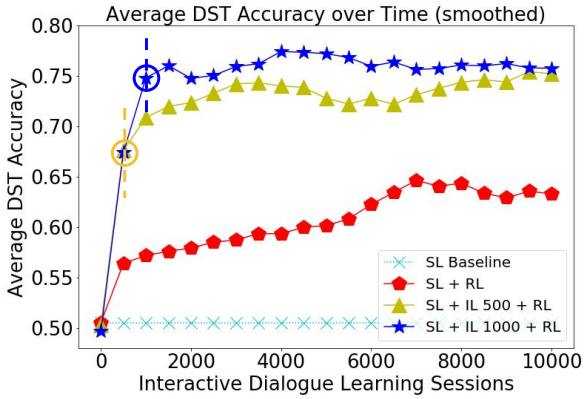


Figure 5: Interactive learning curves on dialogue state tracking accuracy.

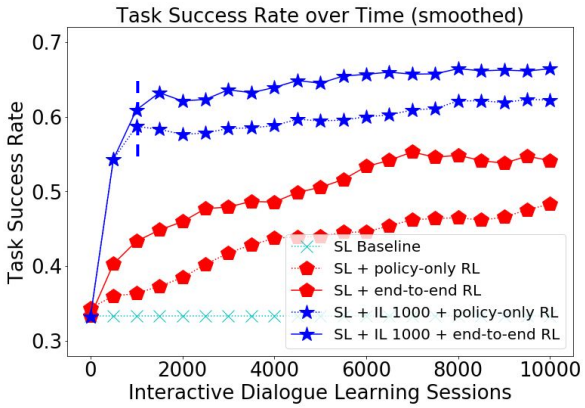


Figure 6: Interactive learning curves on task success rate with different RL training settings.

number of dialogue turns. The red curve with RL applied directly after supervised pre-training model gives the lowest average number of turns at the end of the interactive learning cycles, comparing to models with imitation dialogue learning. This seems to be contrary to our observation in Figure 3 that imitation learning with human teaching helps in achieving higher task success rate. By looking into the generated dialogues, we find that the `SL + RL` model can handle easy tasks well but fails to complete more challenging tasks. Such easy tasks typically can be handled with fewer number of turns, which result in the low average turn size for the `SL + RL` model. On the other hand, the imitation plus RL models attempt to learn better strategies to handle those more challenging tasks, resulting in higher task success rates and also slightly increased dialogue length comparing to `SL + RL` model.

Dialogue State Tracking Accuracy Similar to the results on task success rate, we see that imitation learning with human teaching quickly improves dialogue state tracking accuracy in just a few hundred interactive learning sessions. The joint slots tracking accuracy in the evaluation of SL model using fixed corpus is 84.57% as in Table 4.3. The accuracy drops to 50.51% in the interactive evaluation with the introduction of new NLG templates. Imitation learning with human teaching effectively adapts the neural dialogue model to the new user input and dialogue state distributions, improving the DST accuracy to 67.47% after only 500 imitation dialogue learning sessions. Another encouraging observation is that RL on top of SL model and IL model not only improves task success rate by optimizing dialogue policy, but also further improves dialogue state tracking performance. This shows the benefits of performing end-to-end optimization of the neural dialogue model with RL during interactive learning.

End-to-End RL Optimization To further show the benefit of performing end-to-end optimization of dialogue agent, we compare models with two different RL training settings, the end-to-end training and the policy-only training. End-to-end RL training is what we applied in previous evaluation sections, in which the gradient propagates from system action output layer all the way back to the natural language user input layer. Policy-only training refers to only updating the policy network parameters during interactive learning with RL, with all the other underlying system parameters fixed. The evaluation results are shown in Figure 6. From these learning curves, we see clear advantage of performing end-to-end model update in achieving higher dialogue task success rate during interactive learning comparing to only updating the policy network.

4.5 Human User Evaluations

We further evaluate the proposed method with human judges recruited via Amazon Mechanical Turk. Each judge is asked to read a dialogue between our model and user simulator and rate each system turn on a scale of 1 (frustrating) to 5 (optimal way to help the user). Each turn is rated by 3 different judges. We collect and rate 100 dialogues for each of the three models: (i) SL model, (ii) SL model followed by 1000 episodes of IL, (iii) SL and IL followed by RL. Table 3 lists the mean and standard deviation of human scores overall system turns. Performing interactive learning with imitation and reinforcement learning clearly improves the quality of the model according to human judges.

Model	Score
SL	3.987 ± 0.086
SL + IL 1000	4.378 ± 0.082
SL + IL 1000 + RL	4.603 ± 0.067

Table 3: Human evaluation results. Mean and standard deviation of crowd worker scores (between 1 to 5).

5 Conclusions

In this work, we focus on training task-oriented dialogue systems through user interactions, where the agent improves through communicating with users and learning from the mistake it makes. We propose a hybrid learning approach for such systems using end-to-end trainable neural network model. We present a hybrid imitation and reinforcement learning method, where we firstly train a dialogue agent in a supervised manner by learning from dialogue corpora, and continuously to improve it by learning from user teaching and feedback with imitation and reinforcement learning. We evaluate the proposed learning method with both offline evaluation on fixed dialogue corpora and interactive evaluation with users. Experimental results show that the proposed neural dialogue agent can effectively learn from user teaching and improve task success rate with imitation learning. Applying reinforcement learning with user feedback after imitation learning with user teaching improves the model performance further, not only on the dialogue policy but also on the dialogue state tracking in the end-to-end training framework.

References

Antoine Bordes and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. In *International Conference on Learning Representations*.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. Towards end-to-end reinforcement learning of dialogue agents for information access. In *ACL*.

Mihail Eric and Christopher D Manning. 2017. A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue. In *EACL*.

Milica Gašić, Catherine Breslin, Matthew Henderson, Dongho Kim, Martin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Steve Young. 2013. Online policy optimisation of bayesian spoken dialogue systems via human interaction. In *ICASSP*.

Milica Gasic and Steve Young. 2014. Gaussian processes for pomdp-based dialogue manager optimization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* .

Dilek Hakkani-Tür, Gökhan Tür, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Interspeech*.

James Henderson, Oliver Lemon, and Kallirroi Georgila. 2008. Hybrid reinforcement/supervised learning of dialogue policies from fixed data sets. *Computational Linguistics* .

Matthew Henderson, Blaise Thomson, and Jason Williams. 2013. Dialog state tracking challenge 2 & 3. http://camdial.org/~mh521/dstc/.

Matthew Henderson, Blaise Thomson, and Jason Williams. 2014a. The second dialog state tracking challenge. In *SIGDIAL*.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014b. Robust dialog state tracking using delexicalised recurrent neural networks and unsupervised gate. In *IEEE SLT*.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014c. Word-based dialog state tracking with recurrent neural networks. In *SIGDIAL*.

Filip Jurčíček, Blaise Thomson, and Steve Young. 2012. Reinforcement learning for parameter estimation in statistical spoken dialogue systems. *Computer Speech & Language* 26(3):168–192.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P Spithourakis, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. In *ACL*.

Xuijun Li, Yun-Nung Chen, Lihong Li, and Jianfeng Gao. 2017. End-to-end task-completion neural dialogue systems. *arXiv preprint arXiv:1703.01008* .

Bing Liu and Ian Lane. 2016. Joint online spoken language understanding and language modeling with recurrent neural networks. In *SIGDIAL*.

Bing Liu and Ian Lane. 2017a. An end-to-end trainable neural network model with belief tracking for task-oriented dialog. In *Interspeech*.

Bing Liu and Ian Lane. 2017b. Iterative policy learning in end-to-end trainable task-oriented neural dialog models. In *Proceedings of IEEE ASRU*.

Bing Liu, Gokhan Tur, Dilek Hakkani-Tur, Pararth Shah, and Larry Heck. 2017. End-to-end optimization of task-oriented dialogue model with deep reinforcement learning. In *NIPS Workshop on Conversational AI*.

Fei Liu and Julien Perez. 2017. Gated end-to-end memory networks. In *EACL*.

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. 2015. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)* .

Nikola Mrkšić, Diarmuid O Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2016. Neural belief tracker: Data-driven dialogue state tracking. *arXiv preprint arXiv:1606.03777* .

Baolin Peng, Xiujun Li, Lihong Li, Jianfeng Gao, Asli Celikyilmaz, Sungjin Lee, and Kam-Fai Wong. 2017. Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. In *Proceedings of EMNLP*.

Antoine Raux, Brian Langner, Dan Bohus, Alan W Black, and Maxine Eskenazi. 2005. Lets go public! taking a spoken dialog system to the real world. In *Interspeech*.

Stéphane Ross and Drew Bagnell. 2010. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. pages 661–668.

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

Alexander I Rudnicky, Eric H Thayer, Paul C Constantinides, Chris Tchou, R Shern, Kevin A Lenzo, Wei Xu, and Alice Oh. 1999. Creating natural dialogs in the carnegie mellon communicator system. In *Eurospeech*.

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 2007. Agenda-based user simulation for bootstrapping a pomdp dialogue system. In *NAACL-HLT*.

Minjoon Seo, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Query-regression networks for machine comprehension. *arXiv preprint arXiv:1606.04582* .

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2015. Building end-to-end dialogue systems using generative hierarchical neural network models. *arXiv preprint arXiv:1507.04808* .

Pararth Shah, Dilek Hakkani-Tür, Liu Bing, and Gokhan Tür. 2018. Bootstrapping a neural conversational agent with dialogue self-play, crowdsourcing and on-line reinforcement learning. In *NAACL-HLT*.

Pararth Shah, Dilek Hakkani-Tür, and Larry Heck. 2016. Interactive reinforcement learning for task-oriented dialogue management. In *NIPS 2016 Deep Learning for Action and Interaction Workshop*.

Pei-Hao Su, Pawel Budzianowski, Stefan Ultes, Milica Gasic, and Steve Young. 2017. Sample-efficient actor-critic reinforcement learning with supervised data for dialogue management. In *SIGDIAL*.

Pei-Hao Su, Milica Gasic, Nikola Mrksic, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. On-line active reward learning for policy optimisation in spoken dialogue systems. In *ACL*.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *EACL*.

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *ACL*.

Jason D Williams and Geoffrey Zweig. 2016. End-to-end lstm-based dialog control optimized with supervised and reinforcement learning. *arXiv preprint arXiv:1606.01269* .

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* .

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE* 101(5):1160–1179.

Tiancheng Zhao and Maxine Eskenazi. 2016. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. In *SIGDIAL*.

LSDSCC: A Large Scale Domain-Specific Conversational Corpus for Response Generation with Diversity Oriented Evaluation Metrics

Zhen Xu[1], **Nan Jiang**[2], **Bingquan Liu**[1], **Wenge Rong**[2],
Bowen Wu[3], **Baoxun Wang**[3], **Zhuoran Wang**[3], **and Xiaolong Wang**[1]
[1]School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China
[2]School of Computer Science and Engineering, Beihang University, Beijing, China
[3]Tricorn (Beijing) Technology Co., Ltd, Beijing, China
{zxu,liubq,wangxl}@insun.hit.edu.cn
{nanjiang, w.rong}@buaa.edu.cn
{wubowen, wangbaoxun, wangzhuoran}@trio.ai

Abstract

It has been proven that automatic conversational agents can be built up using the End-to-End Neural Response Generation (NRG) framework, and such a data-driven methodology requires a large number of dialog pairs for model training and reasonable evaluation metrics for testing. This paper proposes a Large Scale Domain-Specific Conversational Corpus (LSDSCC) composed of high-quality query-response pairs extracted from the domain-specific online forum, with thorough pre-processing and cleansing procedures. Also, a testing set, including multiple diverse responses annotated for each query, is constructed, and on this basis, the metrics for measuring the diversity of generated results are further presented. We evaluate the performances of neural dialog models with the widely applied diversity boosting strategies on the proposed dataset. The experimental results have shown that our proposed corpus can be taken as a new benchmark dataset for the NRG task, and the presented metrics are promising to guide the optimization of NRG models by quantifying the diversity of the generated responses reasonably.

1 Introduction

Conversational agents (a.k.a. Chat-bots) are effective media to establish communications with human beings and have received much attention from academic and industrial experts in recent years (Serban et al., 2017). One essential fact promoting the research work on conversational agents is the explosive growth of human interaction data accumulated in the social network services, such as Twitter[1] and Reddit[2]. So, it is possible to build Chat-bots based on data-driven approaches (Serban and Pineau, 2015).

[1]https://twitter.com/
[2]https://www.reddit.com/

Nevertheless, there still remains a great challenge for building such conversational agents: at present, the automatic evaluation metrics of NRG models can hardly afford to measure the semantic relevance and diversity of generated results reasonably, and even the latter evaluation aspect has been paid little attention. The widely accepted evaluating methods employed by the existing NRG models can be categorized as: a) metrics inherited from Machine Translation, e.g., BLEU, Perplexity, etc. (Yao et al., 2015; Lowe et al., 2017; Wu et al., 2018); b) discrete scores measuring the quality of generated results by human labeling (Shang et al., 2015; Serban et al., 2016; Xu et al., 2017); and c) case study comparing the generated results of different NRG models (Shang et al., 2015; Wang et al., 2017). The disappointing situation is that these evaluating methods have not revealed tangible difference among NRG models, the reasons for which can be reflected by the example given in Table 1.

Query: Where did you get that from?
Ground-truth responses: I got it from her.
– I do not know. – Cloverfield wiki. – New York Times. – From movie theatre.
Query: Airplane is now available on Netflix!
Ground-truth responses: Thank you!
– Is it worth watching? – Thank you for that I'll add it to my list. – Awesome, I haven't watched it!

Table 1: Cases of queries with diverse responses.

For each query in Table 1, one response from the testing set is taken as the ground truth, together with responses with more morphological and semantic variations, marked with the symbol "–". These samples indicate that the numerical metrics inherited from NMT which discard the diver-

Proceedings of NAACL-HLT 2018, pages 2070–2080
New Orleans, Louisiana, June 1 - 6, 2018.

sity among responses, cannot reflect marginal differences among generative models, which is supported by the research work of Liu et al. (2016). Thus, an NRG model with good capability to produce diverse and meaningful responses is possible to be judged as a poor one by the BLEU/Perplexity based evaluations. Meanwhile, the metrics based on human labeling are still promising, yet the expensive cost and inconsistency among labelers limit the scale of human-annotation. Therefore, it becomes a necessity to develop reasonable automatic evaluation metrics, that can be taken to measure both candidate response's diversity and its relevance to the given query, to effectively guide the training of NRG models towards the state of promoting meaningful and diverse responses (Li et al., 2016a; Shao et al., 2017; Freitag and Al-Onaizan, 2017).

In order to evaluate the performance of NRG models automatically and reasonably, a well-annotated testing set should be built first. But then, building such a high-quality testing set is a non-trivial task indeed. On one hand, most existing source datasets cover various domains, making it difficult to evaluate the generated results in case that the domain of the generated response is different from that of the reference. On the other hand, a large number of noises, typos, and slangs distribute in existing large-scale datasets, such as Twitter corpus (Ritter et al., 2011) and Ubuntu dialog corpus (Kadlec et al., 2015). For instance, there are many file directories with computer names in Ubuntu dialog corpus. Therefore, qualified domain-specific datasets are desperately required to evaluate NRG models with different architectures reasonably.

To address the above issues, we build a high-quality and domain-specific dialog corpus composed of a carefully prepared training set, and meanwhile, a testing set is constructed by collecting multiple reference responses for each query and conducting group-aware human annotation on collected responses. On this basis, we proposed three discriminative metrics: `MaxBLEU`, Mean Diversity Score (`MDS`), and Probabilistic Diversity Score (`PDS`), to primarily evaluate the diversity of generated responses with relevance also considered. To further assess the performance and effectiveness of the test set cooperating with the proposed metrics, the widely applied Sequence-to-Sequence (Seq2Seq) (Bahdanau et al., 2014; Sutskever et al., 2014) based models with the available diversity promotion methods are implemented, and experiments are conducted on the proposed Large Scale Domain-Specific Conversational Corpus (LSDSCC) dataset. The experimental results stay consistent with the previous experience acquired from human-labeled sets, and the performance of these models suggests that the LSDSCC corpus and discriminative metrics will provide insights for future research in the field of NRG.

2 Related Work

Seq2Seq based conversation modeling approaches have been proven to be able to generate response directly (Vinyals and Le, 2015; Shang et al., 2015). However, these models tend to produce generic responses to any given queries, namely the *deficient diversity problem* (Shao et al., 2017). Recent studies attempt to constrain these universal replies and promote more diverse responses with various strategies during the procedure of training or inference (Li et al., 2016a,b; Mou et al., 2016; Xing et al., 2017; Shao et al., 2017). Besides, there still exits another meaningful option, that is, to employ reasonable diversity oriented evaluation metrics to guide the optimization of models.

The quality of testing sets is a primary factor for such evaluation of NRG models. Existing large-scale corpora contain the Movie Dialogue, Ubuntu, Twitter, and Reddit corpus (Banchs, 2012; Uthus and Aha, 2013; Ritter et al., 2010; Schrading et al., 2015). The Ubuntu corpus is built by scraping a large scale tech support dialogues from Ubuntu IRC forum for building response ranking models (Kadlec et al., 2015). Similarly, Sordoni et al. (2015) provide external context information for message response pairs from Twitter FireHose. Besides, Dodge et al. (2016) and Schrading et al. (2015) collect real conversations from movie categories of Reddit community, which are integrated into a multi-task corpus on movie for the ranking task and discourse analysis. In the above corpora, there are only one or two reference responses for most query, which is completely unlike that of the practical conversation scenario (Li et al., 2017b). By contrast, this paper construct a high-quality testing set, including multi-references for each query. In this regard, our testing set is more close to the real-world setting.

Besides the testing set, evaluation metrics are also important for the performance measurement of NRG models. Most frequently applied evaluation metrics for NRG models are inherited from NMT to measure the fluency and relevance of generated responses, such as Perplexity, BLEU (Papineni et al., 2002) and deltaBLEU (Galley et al., 2015). Although these metrics demonstrate the relevance between the given query and the generated responses, they overlook the reply's diversity that is of great importance in conversation setting. Thus, efforts are devoted to simulate the human subjective judgment, which is similar with the response ranking task in retrieval-based chat agents (Lowe et al., 2017; Tao et al., 2018), but unavoidable uncertainty and errors are brought into the systems (Hu et al., 2014). In addition, automatic evaluation metrics (e.g. BLEU, deltaBLEU, etc.) are limited by the fact that each query only has references with the exact same meaning and many overlapped phrases, which is unreasonable in the conversational scenario.

3 Data Processing and Analysis

Previous studies indicate that more focused topics and less diverged domain is helpful to guide NRG models away from the state of producing universal responses (Mou et al., 2016; Xing et al., 2017), so we compose a domain-specific corpora by constraining the domain of crawled dialogues from Reddit to its movie discussion board[3]. The quality of the data in Reddit movie category has been discussed by Stoddard (2015) and Jamnik and Lane (2017), who point out that the popularity is a good indication of relative quality and the movie category is one of the most popular boards in Reddit. Thus, the data in Reddit movie category is originally high-quality. In this section, the pipeline for building the LSDSCC dataset will be discussed in detail, and necessary statistical indicators are collected to demonstrate its distribution. Moreover, human evaluation is conducted to measure the quality of the obtained training set.

3.1 Data Processing

Data Cleansing. We crawl threads from the movie discussion board of Reddit that includes human-to-human conversations as the raw dataset, and conduct the following cleansing operations:

a) For each thread, we strip away the *markdown* and *html* syntax tokens, e.g., "[word](url)" is transformed to "word", ">" is reformed to ">", etc. Meanwhile, all forms of urls, emails and digits withing the paragraphs are normalized as "url", "email" and "digits" tokens respectively;

b) As emoticons in the data originated from social media services always provide essential emotional information of users, we propose to convert the same groups of emoticons into corresponding words (e.g., ":-)" will be reverted to "happy") to preserve such emotion knowledge;

c) Finally, replicated words or characters (e.g., "cooool" and "ahahaha", etc.) are substituted with its normal form using regular expressions.

Query Vocabulary	Size	Coverage (%)
overlong words	17,084	10.32 %
non-ascii words	58,720	35.46 %
Response Vocabulary	**Size**	**Coverage (%)**
overlong words	15,914	7.94 %
non-ascii words	71,997	35.94 %

Table 2: Composition of noise words in the query and response vocabulary of the raw data.

Vocabulary Truncation. After the above preprocessing operations, there still exist redundant unformatted slang and noisy strings (e.g., "Iloveyou"), which have low-frequency in the crawled raw data. Consequently, the vocabulary size of the dataset is exceeding $160K$ as shown in Table 2. Keeping a such large vocabulary for Seq2Seq based models will consume excessive memory and make those models difficult to converge, while pruning low-frequency unformatted slang and noisy strings into "UNK" symbols would directly harm the performance of the model since excessive knowledge hidden in these strings are ignored in the training process. To address this issue, we break these slangs and noises into several frequent words in our corpus and eliminate non-ASCII tokens. In this way, sufficient information of the dataset is maintained for model training. Finally, the vocabulary sizes of the dataset are reduced to around $50K$.

Dialog Pruning. Statistical results on the sentence length of query-response pairs in the cleaned corpus are illustrated in Fig. 1. Concerning the fact that recurrent neural networks can not efficiently capture the semantics of over-long sentences and previous studies indicate that such re-

[3]https://www.reddit.com/r/movies/, selected from https://www.reddit.com/r/datasets

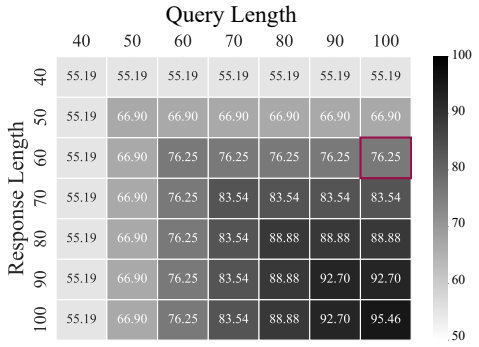


Figure 1: Sentence length coverage of queries and responses within the dataset.

sponses would make the decoder hard to converge (Greff et al., 2017), it is necessary to prune the pairs containing very long sentences. After the sentences are tokenized using the *NLTK toolkits*[4], the cases with queries longer than 100 or responses exceeding 60 words are pruned directly, and 76.25% of the dataset are finally reserved.

After pruning the corpus, there remain 738,095 single-turn and 346,543 multi-turn conversations. Since this paper focuses on the single-turn dialogs, the evaluative testing set and detailed experiments in the following sections are designed for single-turn corpus. As the testing set will select pairs from the preprocessed data, the corresponding tuples will be deleted to avoid coverage.

3.2 Query-Response Relevance of the Dataset

As one of the most important qualities of the conversational corpus, the query-response relevance demonstrates the overall quality of the dataset. Human evaluations of the query-response relevance are conducted to validate the quality of the dataset used in this paper. Nine experienced annotators are invited to evaluate the query-response relevance of 500 single-turn dialogs uniformly sampled from the whole dataset obtained in Subsection 3.1. In the evaluation, we ask each annotator to label whether the response is appropriate to the corresponding query in the given query-response pair. A pair is tagged as "Unsure" if the annotator could not confirm the degrees of relevance without related context and background movie knowledge. The labeled result is shown in Table 3. It is observed that 85% samples in the query-response relevance task are confirmed to keep high relevance between the query and the corresponding responses. Moreover, there exist only about 6.6% irrelevant noises. So, the resource can be considered as a high-quality one and can be used in the practical task.

[4]http://www.nltk.org/

Category	Relevant	Unsure	Irrelevant
Numbers	427	40	33
Percentage	85.4%	8%	6.6%

Table 3: Query-Response Relevance on the single-turn training set.

4 Testing Set and Evaluation Metrics

Existing evaluation metrics of dialog agents measure the quality of the generated sentences only by referring to the existing responses, which obeys the same principle with NMT models' metrics. However, one essential difference between NRG and NMT lies in the fact that, a large group of responses can be considered as relevant to a given query in conversations, while the number of references to a translation result is quite limited for NMT models. So the diversity degree of candidates which have not covered by NMT oriented evaluation metrics, is supposed to be quantified and measured in NRG models.

Currently, few studies focus on the evaluation based on the group of references, which is more meaningful and reasonable for NRG models. Therefore, we proposed three metrics: `MaxBLEU`, Mean Diversity Score, and Probabilistic Diversity Score, to quantify both the relevance and diversity of the generated responses. Since these metrics are based on the multi-reference, we first describe the procedure of building testing set, with multi-references for each query. Then, the metrics for NRG models are detailed based on the multi-reference testing set.

4.1 Multi-Reference Testing Set Construction

Fig. 2 illustrates the response quantity distribution of queries in the preprocessed data. While the testing set is randomly sampled from the preprocessed data, the response quantity distribution of the testing set is the same as that in Fig. 2. In this case, the multi-reference testing set for NRG evaluation is difficult to construct by directly extracting samples from the dialog corpus, since there are too few queries that contain more than three responses. Roughly choosing samples from such data is possible to bring topic bias into the testing set, and manually filtering suitable candidate pairs from

them is also time-consuming and expensive. Nevertheless, there exist large amounts of queries that are highly semantically similar or correlated with each other. This indicates that the multiple references can be obtained by selecting responses of queries that are semantically identical to the original query. What's more, the human-annotation is involved to proofread the filtered pairs' quality and complete the final labeling.

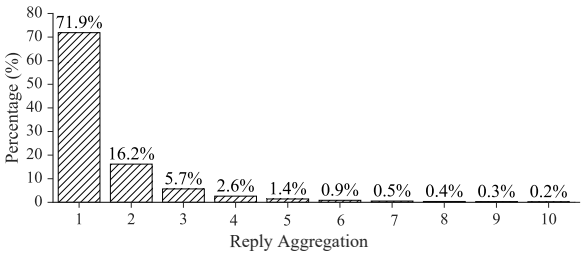


Figure 2: Distribution of reply quantities in training set.

When constructing the testing set, the very first step is getting semantically similar (or even identical) queries with the given ones. For this purpose, this paper adopts the TF-IDF similarity and semantic embedding based distance to measure the similarity between queries. The procedure of gaining similar queries is divided into two stage: In the first stage, we employ Apache Lucene[5] to exploit the word-level TF-IDF patterns within queries, and then extract the top 100 similar queries with highest scores given by Lucene for each query. Yet, these candidates only capture n-gram level similarity with the probably diverged semantics. Thus, in the second stage, we utilize paragraph vector algorithm (a.k.a Doc2vec[6]) (Le and Mikolov, 2014) to resort the selected similar queries in the semantic space and only queries of similarity score higher than a certain threshold (i.e., 0.9) are reserved. Table 4 lists several identical queries filtered by Lucene and Doc2vec methods with the given query. It should be noted that the Lucene index and Doc2vec need to be initialized by feeding all the sentences in the dialogue corpus.

To reserve as much information as possible and balance the distribution of the composed testing set, we divide the dataset into several subsets based on the response number of queries, and then sample testing data from each subset uniformly. Concretely, according to the response number, queries of the dataset are divided into three subsets: a) queries with less than 3 responses, b) queries with 3 to 5 responses, and c) queries with more than 5 responses. We randomly sample 100 queries from each subset, and thus 300 queries are obtained as the testing set. Aiming at building a multiple references testing set, each query in the testing set is assigned with 15 responses, including the original responses and the ones of the most similar queries obtained by the procedure of last paragraph.

Similar Queries	Similarity
If you haven't already, watch the animatrix.	0.97
Do not watch the animatrix, you may leave you house.	0.95
I don't have much to ad except, that people really should watch animatrix.	0.94
I recommend you watch the matrix.	0.91

Table 4: Filtered queries identical to the original query: "**You should watch the animatrix.**"

Afterwards, three skilled and experienced labelers familiar with movies are employed and carefully trained to crosswise annotate the filtered testing set. In addition, labelers can also obtain some background of the corresponding query since there are additional details for most queries in Reddit. In this case, the quality of selected samples can be guaranteed. Besides, the annotators are asked not only to label the relevance of query and reference responses, but also reorganize the independent references into groups by the semantic similarity subjectively. The grouping strategy is introduced for the purpose of evaluating the diversity of responses generated by different models.

In the relevance oriented annotation procedure, the labelers are first asked to judge whether a candidate response is appropriate and natural to the input query. If a candidate response is grammatically correct and semantically relevant with the corresponding query from the annotators' perspective, it should be labeled as "**1**". Otherwise, the annotators have to give "**0**" label to the candidate. Then, for each query, the annotators need to split responses labeled with "**1**" into different groups based on word overlapping between them, with stop-word overlapping ignored. Finally, the groups with the similar semantics are merged into a larger group by the annotators, so as to get the final grouped responses.

[5]https://lucene.apache.org/

[6]https://radimrehurek.com/gensim/models/doc2vec.html

At last, we obtain a high-quality testing set, in which each query is assigned with different numbers of reference responses. Fig. 3 shows the distribution of the response numbers in the testing set. Comparing to the original response number distribution in Fig. 2, the replies distribution of the testing set is much more appropriate for the conversational scenario. Furthermore, responses to the corresponding query are categorized into several groups. In this case, NRG models can be evaluated reasonably using such a testing set. One sampled case in the testing set is shown in the left phase of Fig. 4, and there are eight responses in the labeled data divided into four groups. The different metrics in this figure will be introduced in the following sections. It should be noted that both the single-turn dialogs and the annotated testing set are released[7].

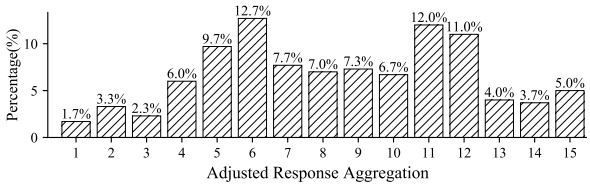


Figure 3: Aggregated responses per query in testing set.

4.2 The Metric on Response Relevance

Since the NRG architecture is analogous to the NMT models, introducing the BLEU scores to evaluate the semantic relevance of the generated results is acceptable. However, it is not reasonable to average the BLEU scores of the generated response to each reference, because the semantic of each reference varies significantly. Aiming at revealing the variation and diversity among responses, which have not yet covered at the NMT models, we propose a `MaxBLEU` metric customized for response generation based on the Multi-BLEU metric (Madnani et al., 2008). Noticing that the metrics inherited from SMT, like BLEU, is not able to evaluate the diversity of responses, we propose the specified metrics for diversity evaluation, which will be described in the next subsection.

Given an input query, the NRG model is able to generate a set of hypothesis $\{h_i\}$[8]. Meanwhile, according to the human-annotation strategy described in Subsection 4.1, the set of references can be reorganized, based on their semantic similarity, into the groups with the format of $\{r_{ij}\}$, where r_{ij} denotes the j-th reference in the i-th group. On this basis, the `MaxBLEU` metric is defined as:

$$\texttt{MaxBLEU}(h_i) = \arg\max_k \text{Multi-BLEU}(h_i, r_{k\cdot}) \quad (1)$$

where $r_{k\cdot}$ denotes all the references in the k-th group. That is, we begin by calculating all the multi-BLEU scores between each hypothesis and grouped references, and pick the score for the sentence with the highest BLEU as the score for this set of hypothesis, so that we make an alignment between generated hypothesis h to the group-aware references r. For simplicity, one response can only be aligned to one group reference, and multi-group references are not considered in this work.

4.3 Metrics on Response Diversity

Given a query, the diversity degree of candidate responses is an essential criterion for evaluating the performances of NRG models. Currently, most studies tend to demonstrate the diversities of different models by sampling and comparing the generated results, or labeling the diversity of the generated samples, which makes it difficult to benchmark and automatically evaluate different models. Although Li et al. (2017a) propose to calculate the number of distinct unigrams and bigrams of generated responses, such scores do not align well with human inspection (Serban et al., 2017).

Algorithm 1 Two Response Diversity Metrics.

Input:
hypothesis set `H` and reference set `R` ;
Output:
Mean Diversity Score (`MDS`);
Probabilistic Diversity Score (`PDS`);
for all $r_{i\cdot} \in$ `R` **do** ▷ Initialize
  $p_i = 1/|\texttt{R}|$,
  $p'_i = |r_{i\cdot}|/\sum_j |r_{j\cdot}|$.
end for
for all $h_i \in$ `H` **do** ▷ Compute alignment
  $k = \arg\max_j$ Multi-BLEU$(h_i, r_{j\cdot})$.
  $\texttt{MDS}(k) = p_k$,
  $\texttt{PDS}(k) = p'_k$.
end for
return $\sum_k \texttt{MDS}(k), \sum_k \texttt{PDS}(k)$

Therefore, we propose two evaluative metrics based on the `MaxBLEU` metric for diversity measurement: a) Mean Diversity Score (`MDS`) and b) Probabilistic Diversity Score (`PDS`). Basically, the

[7] https://drive.google.com/file/d/1nbpbnhwNP14xAc4SAc1-NN5lvEr01dQb/view?usp=sharing

[8] Following the terms in machine translation, this part takes "hypothesis" to represent "response".

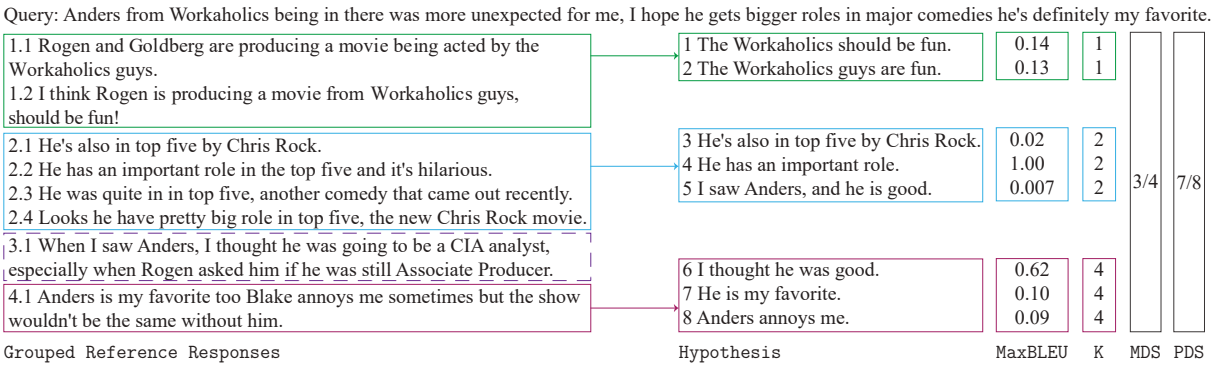


Figure 4: A sampled testing case including a given query, the grouped reference responses and the generated ones (hypothesis) with the proposed metrics, among which "K" is obtained by performing $\arg\max$ upon MaxBLEU scores. The MDS metric is calculated with the partitioned groups and PDS metric is calculated with the weight of each group in the overall the candidate set.

two metrics aim at measuring the overall diversity of the whole set of generation results (hypothesis) by taking them as an entirety, and the detailed calculation steps of the proposed metrics are illustrated in Algorithm 1. According to the algorithm, the PDS metric assumes that the weight of each reference group is distributed uniformly, regardless of the reference number in each group. Similarly, the MDS metric takes the count of the members in each group as the weight of the corresponding group, and actually compute the weighted coverage upon the reference group.

5 Experiments and Analysis

In this section, we present the detailed experiments on the single-turn dialog dataset and analysis on generated results, in accordance to the proposed metrics.

5.1 Baselines and Experimental Setups

Experiments are conducted using the popular Seq2Seq based models with the currently available diversity prompting strategies as follows:

1) **Basic Seq2Seq.** We employ the basic Seq2Seq to build the encoder-decoder architecture running on the proposed dataset, by taking the bidirectional LSTM cell as the encoder to address the input sentences ordering problem and classic LSTM cell as the decoder (Vinyals and Le, 2015).

2) **Attention-Seq2Seq.** As proposed by Vinyals and Le (2015); Luong et al. (2015), a concatenated version of attention mechanism is applied upon the basic Seq2Seq model.

3) **Greedy-Seq2Seq.** Based on the basis Seq2Seq model, the diversity promotion strategy proposed by Li et al. (2016b) is applied in the generating procedure, and the training procedure stays the same. Hyper parameter γ, a.k.a. *diversity rate*, are set with empirical experiments (i.e., $\gamma = 0.1, 0.8$) to reveal the efforts.

4) **Greedy-Attn-Seq2Seq.** Following the work of Li et al. (2016b), the greedy diversity promotion strategy is applied on the Seq2Seq model with attention mechanism similar with model 2, and we set hyper parameter $\gamma = 0.1, 0.8$.

5) **MMI-Seq2Seq.** In the generation procedure, Maximum Mutual Information (MMI) model is applied in the decoder to prune generic answers on the basic Seq2Seq model (Li et al., 2016a).

In our research, we implement these models on the TensorFlow platform[9], and Adam optimizer (Kingma and Ba, 2015) is employed for gradient optimization during training. Besides, we choose to prune the words whose frequencies are below 2, so the source and target vocabulary are set to $42,257$ and $46,865$ respectively.

In addition, we set the batch size to 50, hidden size of encoder to 256, hidden size of decoder to 512 and learning rate to $2e-4$. The gradients are clipped within $[-3.0, 3.0]$ to avoid the gradient explosion problem. Every model runs on a single GPU separately for at least one week before convergence. Afterwards, for all these methods, we generate a set of hypothesis sentences with beam size set to $k = 50$, and the evaluation scores are obtained using the proposed metrics.

After running through 25 epochs on the dataset, the training log-loss of the basic Seq2Seq mod-

[9]https://www.tensorflow.org

Models	MaxBLEU	MDS	PDS
Seq2Seq (Vinyals and Le, 2015)	1.30	0.230	0.253
Attention-Seq2Seq (Luong et al., 2015)	1.42	0.235	0.262
Greedy-Seq2Seq ($\gamma = 0.1$) (Li et al., 2016b)	1.88	0.249	0.243
Greedy-Seq2Seq ($\gamma = 0.8$)	1.72	0.297	0.291
Greedy-Attn-Seq2Seq ($\gamma = 0.1$) (Li et al., 2016b)	**2.27**	0.252	0.248
Greedy-Attn-Seq2Seq ($\gamma = 0.8$)	2.05	0.285	0.287
MMI-Seq2Seq (Li et al., 2016a)	2.15	**0.311**	**0.329**

Table 5: Performances of different models trained on the LSDSCC dataset with three metrics: MaxBLEU, MDS, PDS.

els converge to about 4.2 and the Seq2Seq models augmented with attention converge to 3.1. Also, we set the dropout rate to 0.5, which enables us to tune the models though much more epochs and avoid the over-fitting problems.

5.2 Relevance Analysis

The semantic relevance of the generated responses is represented by the MaxBLEU scores, which are listed in the corresponding column of Table 5. From this benchmarking table, it can be observed that the attention mechanism is helpful for decoders to improve the relevance of the generated responses, since the *Attention-Seq2Seq* performs better than the basic Seq2Seq on the dataset, in terms of all the three metrics. However, the relative gain of the attention layer is limited, indicating that modeling relation of query and response by attention module is not able to directly solve the learning paradigm of conversations.

In accordance to the results of Greedy-Seq2Seq ($\gamma = 0.1$) and Greedy-Seq2Seq ($\gamma = 0.8$), the hyper-parameter γ actually plays an important role in the generation steps of the decoder. Since γ is introduced to constrain the selection probability of the next-step word by performing the re-ranking process, and the larger value of this parameter will lead to the greater impact upon generating steps and produce more diverse sentences, we evaluate this greedy strategy with γ set with two empirical value. It can be seen that the model with the smaller γ performs better than the one with the larger parameter, which can be attributed to the fact that responses with more diversity are less similar to references. Similar observation can be get from the results of models Greedy-Attn-Seq2Seq ($\gamma = 0.1$) and Greedy-Attn-Seq2Seq ($\gamma = 0.8$). Besides, the reason for setting $\gamma = 0.1, 0.8$ in this part is that they are well represented for the poor diversity and good diversity, which the exact score of γ will vary under different configurations and structures of model.

In addition, the MMI model is proved to be promising to enhance the generation models, by improving both the relevant and diversity of generated responses. Even though the *MMI-Seq2Seq* model has not got the highest MaxBLEU, it outperforms the other ones on diversity, which will be discussed in the following subsection.

5.3 Diversity Analysis

Table 5 also illustrates the MDS and PDS score of each benchmark. It is observed that the greedy strategies in the generating procedure with the greater parameter γ can boost the diversity of generated responses obviously. This phenomenon is attributed to the inter-sibling ranking policy in the decoding procedure, which tends to choose hypotheses from diverse parents. In addition, the MMI strategy gets the highest MDS and PDS, because the MMI criterion relieves the constraint of the language model, under which general responses always get a higher generative probability.

Meanwhile, the PDS metric aligns well with the basic MDS, but the relative gap becomes larger within the Greedy-Seq2Seq ($\gamma = 0.1$) and Greedy-Seq2Seq ($\gamma = 0.8$) models. The reason for enlarging relative gap between different models, is to distinguish the performance of similar models and evaluate the performance of specific module inside the models. When comparing Seq2Seq and Attention-Seq2Seq, relative gain of applied attention module to the overall model in terms of MDS was 2.1%, while it became 3.6% considering the PDS metric.

Practically, it is reasonable to make a trade-off between the relevance and diversity. The PDS is more suitable for choosing the systems with stringent diversity requirement, and the MDS is a softer metric, which should be taken into consideration when measuring the diversity improvements by integrating some new modules into NRG models.

Moreover, it can be observed that the relevance oriented metric MaxBLEU gets improvement along

with the increasing of the diversity oriented PDS and MDS. This phenomenon indicates a relationship between relevance and diversity against that in some of text generation tasks (e.g., image caption (Yao et al., 2017)). Since there are generally many references for a given query, the relevance and diversity are possible to be improved simultaneously for the response generation task (Li et al., 2016a). And thus, the topic changing on the generated results is tolerable.

5.4 Human Correlation Analysis

To validate the correlations between human ratings and the proposed metrics, we further invite 9 annotators with rich movie knowledge to judge the relevance and diversity of the generated responses from benchmark methods. Each baseline model generate 10 responses for each query in the test dataset. The annotators are first asked to judge whether a generated response is relevant to the query (labeled with 1) or not (labeled with 0). After that, the annotators estimate the diversity of relevant responses of each query with a scale of 1 to 3. The final Fleiss Kappa (Fleiss, 1971) score is 0.46, which denotes moderate agreement of the annotators.

Model	Spearman	p-value	Pearson	p-value
MaxBLEU	0.31	0.022	0.29	0.036
MDS	0.36	0.041	0.33	0.028
PDS	0.39	0.038	0.35	0.040

Table 6: Correlation between the proposed metrics and human judgments for the Reddit dataset.

The Pearson and Spearman correlation between the human evaluations and each metric are given in Table 6. It can be observed that the proposed metrics correlate with human judgments moderately with $p - value < 0.05$, which is quite different from the correlation test in Liu et al. (2016). This can be attributed to the fact that there are multiple references for each query in our test dataset. Although the proposed metrics are derived from the word-overlap based BLEU scores, expanding references of each query makes such scores much more reasonable for evaluating the relevance and diversity of generated responses.

6 Conclusion and Future Work

In this paper, we have proposed the Large Scale Domain-Specific Conversational Corpus (LSDSCC), collected from the movie discuss threads in the Reddit community, for training and testing the Neural Response Generation (NRG) models. In addition, necessary data cleansing and pruning works are done to remove noises in the utterances. Moreover, we employ volunteers to annotate a diverse query-responses testing set, with reference groups taken into consideration for objectively quantifying the diversity of generated results. On the basis of the testing set, we propose two evaluative diversity metrics (mean diversity score and probabilistic diversity score) calculated according to the standard MaxBLEU score.

Furthermore, we investigate the performance of popular Seq2Seq based models with various diversity promotion strategies, and the score of them are collected to validate the effectiveness of the proposed metrics. The proposed dataset and evaluation metrics are expected to be used for the effective training and reasonable testing of NRG models.

In the future studies, we would explore the possibility of promoting diversity on the learning procedure, by directly optimizing diversity loss in the cost function. Besides, injecting external information during response's generation would be another challenging work.

Acknowledgments

We thank the anonymous reviewers for their insightful comments. This research is partially supported by National Natural Science Foundation of China (No.61572151, No.61602131, No.61672192) and the National High Technology Research and Development Program ("863" Program) of China (No.2015AA015405).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.

Rafael E. Banchs. 2012. Movie-dic: a movie dialogue corpus for research and development. In *Proc. of ACL*, pages 203–207.

Jesse Dodge, Andreea Gane, Xiang Zhang, Antoine Bordes, Sumit Chopra, Alexander Miller, Arthur Szlam, and Jason Weston. 2016. Evaluating prerequisite qualities for learning end-to-end dialog systems. In *Proc. of ICLR*.

J. L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382.

Markus Freitag and Yaser Al-Onaizan. 2017. Beam search strategies for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 56–60.

Michel Galley, Chris Brockett, Alessandro Sordoni, Yangfeng Ji, Michael Auli, Chris Quirk, Margaret Mitchell, Jianfeng Gao, and Bill Dolan. 2015. deltableu: A discriminative metric for generation tasks with intrinsically diverse targets. In *Proc. of ACL*, pages 445–450.

K. Greff, R. K. Srivastava, J. Koutnk, B. R. Steunebrink, and J. Schmidhuber. 2017. Lstm: A search space odyssey. *IEEE Trans. Neural Netw. Learn. Syst.*, 28(10):2222–2232.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Proc. of NIPS*, pages 2042–2050.

Matthew R. Jamnik and David J. Lane. 2017. The use of reddit as an inexpensive source for high-quality data. *Practical Assessment Research & Evaluation*, 22(5).

Rudolf Kadlec, Martin Schmid, and Jan Kleindienst. 2015. Improved deep learning baselines for ubuntu corpus dialogs. In *Machine Learning for SLU Interaction Workshop, NIPS*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proc. of ICLR*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proc. of ICML*, pages 1188–1196.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proc. of NAACL-HLT*, pages 110–119.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016b. A simple, fast diverse decoding algorithm for neural generation. *CoRR*, abs/1611.08562.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2017a. Learning to decode for future success. *CoRR*, abs/1701.06549.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017b. Dailydialog: A manually labelled multi-turn dialogue dataset. In *Proc. of IJCNLP*, pages 986–995.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proc. of EMNLP*, pages 2122–2132.

Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic turing test: Learning to evaluate dialogue responses. In *Proc. of ACL*, pages 1116–1126.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proc. of EMNLP*, pages 1412–1421.

Nitin Madnani, Philip Resnik, Bonnie J Dorr, and Richard Schwartz. 2008. Are multiple reference translations necessary? investigating the value of paraphrased reference translations in parameter optimization. In *Proc. of AMTA*, pages 143–152.

Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2016. Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. In *Proc. of COLING*, pages 3349–3358.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. of ACL*, pages 311–318. Association for Computational Linguistics.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of twitter conversations. In *Proc. of NAACL-HLT*, pages 172–180.

Alan Ritter, Colin Cherry, and William B. Dolan. 2011. Data-driven response generation in social media. In *Proc. of EMNLP*, pages 583–593.

Nicolas Schrading, Cecilia Ovesdotter Alm, Ray Ptucha, and Christopher Homan. 2015. An analysis of domestic abuse discourse on reddit. In *Proc. of EMNLP*, pages 2577–2583.

Iulian V. Serban and Joelle Pineau. 2015. Text-based speaker identification for multi-participant open-domain dialogue systems. In *Machine Learning for Spoken Language Understanding and Interaction, NIPS 2015 Workshop*.

Iulian Vlad Serban, Ryan Lowe, Laurent Charlin, and Joelle Pineau. 2017. A survey of available corpora for building data-driven dialogue systems. *CoRR*, abs/1703.05742.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proc. of AAAI*, pages 3776–3784.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proc. of ACL*, pages 1577–1586.

Yuanlong Shao, Stephan Gouws, Denny Britz, Anna Goldie, Brian Strope, and Ray Kurzweil. 2017.

Generating high-quality and informative conversation responses with sequence-to-sequence models. In *Proc. of EMNLP*, pages 2210–2219.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proc. of HLT-NAACL*, pages 196–205.

Greg Stoddard. 2015. Popularity dynamics and intrinsic quality in reddit and hacker news. In *ICWSM*, pages 416–425.

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

Chongyang Tao, Lili Mou, Dongyan Zhao, and Rui Yan. 2018. Ruber: An unsupervised method for automatic evaluation of open-domain dialog systems. In *Proc. of AAAI*, pages 1–8.

David C. Uthus and David W. Aha. 2013. The ubuntu chat corpus for multiparticipant chat analysis. In *AAAI Spring Symposium: Analyzing Microtext*.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. In *Proc. of ICLR*.

Jianan Wang, Xin Wang, Fang Li, Zhen Xu, Zhuoran Wang, and Baoxun Wang. 2017. Group linguistic bias aware neural response generation. In *Proceedings of the 9th SIGHAN Workshop on Chinese Language Processing*, pages 1–10.

Yu Wu, Wei Wu, Dejian Yang, Can Xu, Zhoujun Li, and Ming Zhou. 2018. Neural response generation with dynamic vocabularies. In *Proc. of AAAI*.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *Proc. of AAAI*, pages 3351–3357.

Zhen Xu, Bingquan Liu, Baoxun Wang, SUN Chengjie, Xiaolong Wang, Zhuoran Wang, and Chao Qi. 2017. Neural response generation via gan with an approximate embedding layer. In *Proc. of EMNLP*, pages 617–626.

Kaisheng Yao, Geoffrey Zweig, and Baolin Peng. 2015. Attention with intention for a neural network conversation model. In *Machine Learning for SLU Interaction Workshop, NIPS*, pages 1–7.

Ting Yao, Yingwei Pan, Yehao Li, and Tao Mei. 2017. Incorporating copying mechanism in image captioning for learning novel objects. In *Proc. of CVPR*, pages 5263–5271. IEEE.

EMR Coding with Semi-Parametric Multi-Head Matching Networks

Anthony Rios
Department of Computer Science
University of Kentucky
Lexington, KY
anthony.rios1@uky.edu

Ramakanth Kavuluru
Division of Biomedical Informatics
University of Kentucky
Lexington, KY
ramakanth.kavuluru@uky.edu

Abstract

Coding EMRs with diagnosis and procedure codes is an indispensable task for billing, secondary data analyses, and monitoring health trends. Both speed and accuracy of coding are critical. While coding errors could lead to more patient-side financial burden and misinterpretation of a patient's well-being, timely coding is also needed to avoid backlogs and additional costs for the healthcare facility. In this paper, we present a new neural network architecture that combines ideas from few-shot learning matching networks, multi-label loss functions, and convolutional neural networks for text classification to significantly outperform other state-of-the-art models. Our evaluations are conducted using a well known de-identified EMR dataset (MIMIC) with a variety of multi-label performance measures.

1 Introduction

Electronic medical record (EMR) coding is the process of extracting diagnosis and procedure codes from the digital record (the EMR) pertaining to a patient's visit. The digital record is mostly composed of multiple textual narratives (e.g., discharge summaries, pathology reports, progress notes) authored by healthcare professionals, typically doctors, nurses, and lab technicians. Hospitals heavily invest in training and retaining professional EMR coders to manually annotate all patient visits by reviewing EMRs. Proprietary commercial software tools often termed as computer-assisted coding (CAC) systems are already in use in many healthcare facilities and were found to be helpful in increasing medical coder productivity (Dougherty et al., 2013). Thus progress in automated EMR coding methods is expected to directly impact real world operations.

In the US, the diagnosis and procedure codes used in EMR coding are from the International Classification of Diseases (ICD) terminology (specifically the ICD-10-CM variant) as required by the Health Insurance Portability and Accountability Act (HIPPA). ICD codes facilitate billing activities, retrospective epidemiological studies, and also enable researchers to aggregate health statistics and monitor health trends. To code EMRs effectively, medical coders are expected to have thorough knowledge of ICD-10-CM and follow a complex set of guidelines to code EMRs. For example, if a coder accidentally uses the code "heart failure" (ICD-10-CM code I50) instead of "acute systolic (congestive) heart failure" (ICD-10-CM code I50.21), then the patient may be charged substantially more[1] causing significant unfair burden. Therefore, it is important for coders to have better tools at their disposal to find the most appropriate codes. Additionally, if coders become more efficient, hospitals may hire fewer coders to reduce their operating costs. Thus automated coding methods are expected to help with expedited coding, cost savings, and error control.

In this paper, we treat medical coding of EMR narratives as a multi-label text classification problem. Multi-label classification (MLC) is a machine learning task that assigns a set of labels (typically from a fixed terminology) to an instance. MLC is different from multi-class problems, which assign a single label to each example from a set of labels. Compared to general multi-label problems, EMR coding has three distinct challenges. First, with thousands of ICD codes, the label space is large and the label distribution is extremely unbalanced – most codes occur very infrequently with a few codes occurring several orders of magnitude more than others. Second and more importantly, a patient may have a large number of diagnoses and procedures.

[1] https://nyti.ms/2oxrjCv

Proceedings of NAACL-HLT 2018, pages 2081–2091
New Orleans, Louisiana, June 1 - 6, 2018.

On average, coders annotate an EMR with more than 20 such codes and hence predicting the top one or two codes is not sufficient for EMR coding. Third, EMR narratives may be very long (e.g., discharge summaries may have over 1000 words), which may result in a needle in a haystack situation when attempting to seek evidence for particular codes.

Recent advances in *extreme multi-label classification* have proven to work well for large label spaces. Many of these methods (Yu et al., 2014; Bhatia et al., 2015; Liu et al., 2017) focus on creating efficient multi-label models that can handle 10^4 to 10^6 labels. While these models perform well in large label spaces, they don't necessarily focus on improving prediction of infrequent labels. Typically, they optimize for the top 1, 3, or 5 ranked labels by focusing on the P@1, P@3, and P@5 evaluation measures. The labels ranked at the top usually occur frequently in the dataset and it is not obvious how to handle infrequent labels. One solution would be to ignore the rare labels. However, when the majority of medical codes are infrequent, this solution is unsatisfactory.

While neural networks have shown great promise for text classification (Kim, 2014; Yang et al., 2016; Johnson and Zhang, 2017), the label imbalances associated with EMR coding hinder their performance. Imagine if a dataset contains only one training example for every class leading to *one-shot learning*, a subtask of *few-shot learning*. How can we classify a new instance? A trivial solution would be to use a non-parametric 1-NN (1 nearest neighbor) classifier. 1-NN does not require learning any label specific parameters and we only need to define features to represent our data and a distance metric. Unfortunately, defining good features and picking the best distance metric is nontrivial. Instead of manually defining the feature set and distance metric, neural network training procedures have been developed to learn them automatically (Koch et al., 2015). For example, matching networks (Vinyals et al., 2016) can automatically learn discriminative feature representations and a useful distance metric. Therefore, using a 1-NN prediction method, matching networks work well for infrequent labels. However, researchers typically evaluate matching networks on multi-class problems without label imbalance. For EMR coding with extreme label imbalance with several labels occurring thousands of times, traditional parametric neural networks (Kim, 2014) should work very well on the frequent labels. In this paper, we introduce a new variant of matching networks (Vinyals et al., 2016; Snell et al., 2017) to address the EMR coding problem. Specifically, we combine the non-parametric idea of k-NN and matching networks with traditional neural network text classification methods to handle both frequent and infrequent labels encountered in EMR coding.

Overall, we make the following contributions in this paper:

- We propose a novel semi-parametric neural matching network for diagnosis/procedure code prediction from EMR narratives. Our architecture employs ideas from matching networks (Vinyals et al., 2016), multiple attention (Lin et al., 2017), multi-label loss functions (Nam et al., 2014a), and convolutional neural networks (CNNs) for text classification (Kim, 2014) to produce a state-of-the-art EMR coding model.
- We evaluate our model on publicly available EMR datasets to ensure reproducibility and benchmarking; we also compare against prior state-of-the-art methods in EMR coding and demonstrate robustness across multiple standard evaluation measures.
- We analyze and measure how each component of our model affects the performance using ablation experiments.

2 Related Work

In this section we cover recent methodologies that are either relevant to our approach and problem or form the main ingredients of our contribution.

2.1 Extreme Multi-label Classification

Current methods for extreme MLC fall into two categories: embedding and tree-based methods. Embedding-based methods aim to reduce the training complexity. They effectively reduce the label space by assuming the training label matrix is low rank. Intuitively, rather than learning independent classifiers for each label (binary relevance) (Tsoumakas et al., 2010), classifiers are learned in a reduced label space $\hat{L} \ll L$ where L is the total number of labels. Likewise, a projection matrix is learned to convert predictions from the reduced label space back to the original label space. In general, embedding methods vary

based on how they reduce the label space and how the projection operation is optimized. Tai and Lin (2012) use principal component analysis (PCA) to reduce the label space. Low-rank Empirical risk minimization for Multi-Label Learning (LEML) (Yu et al., 2014) jointly optimizes the label space reduction and the projection processes. RobustXML (Xu et al., 2016) is similar to LEML but it treats infrequent labels as outliers and models them separately. Liu et al. (2017) employ neural networks for extreme multi-label problems using a funnel-like architecture that reduces the label vector dimensionality. Tree-based multi-label methods work by recursively splitting the feature space. These methods usually differ based on the node splitting criterion. FastXML (Prabhu and Varma, 2014) partitions the feature space using the nDCG measure as the splitting criterion. PfastreXML (Jain et al., 2016) improves on FastXML by using a propensity scored nDCG splitting criterion and re-ranking the predicted labels to optimize various ranking measures.

2.2 Memory Augmented Neural Networks

Memory networks (Weston et al., 2014) have access to external memory, typically consisting of information the model may use to make predictions. Intuitively, informative memories concerning a given instance are found by the memory network to improve its predictive power. Kamra et al. (2017) use memory networks to fix issues of catastrophic forgetting. They show that external memory can be used to learn new tasks without forgetting previous tasks. Memory networks are now applied to a wide variety of natural language processing tasks, including question answering and language modeling (Sukhbaatar et al., 2015; Bordes et al., 2015; Miller et al., 2016).

Matching networks (Vinyals et al., 2016; Snell et al., 2017) have recently been developed for few/one-shot learning problems. We can interpret matching networks as a key-value memory network (Miller et al., 2016). The "keys" are training instances, while the "values" are the labels associated with each training example. Intuitively, the concept is similar to a hashmap. The model will search for the most similar training instance to find its respective "value". Also, matching networks can be interpreted as a k-NN based model that automatically learns an informative distance metric. Finally, Altae-Tran et al. (2017) used matching networks for drug discovery, a problem where data is limited.

2.3 Diagnosis Code Prediction

The 2007 shared task on coding radiology reports (Pestian et al., 2007) was the first effort that popularized automated EMR coding. Traditionally, linear methods have been used for diagnosis code prediction. Perotte et al. (2013) developed a hierarchical support vector machine (SVM) model that takes advantage of the ICD-9-CM hierarchy. In our prior work, we train a linear model for every label (Rios and Kavuluru, 2013) and re-rank the labels using a learning-to-rank procedure (Kavuluru et al., 2015). Zhang et al. (2017) supplement the diagnosis code training data with data from PubMed (biomedical article corpus and search system) to train linear models using both the original training data and the PubMed data.

Recent advances in neural networks have also been put to use for EMR coding: Baumel et al. (2018) trained a CNN with multiple sigmoid outputs using binary cross-entropy. Duarte et al. (2017) use hierarchical recurrent neural networks (RNNs) to annotate death reports with ICD-10 codes. Vani et al. (2017) introduced grounded RNNs for EMR coding. They found that iteratively updating their predictions at each time step significantly improved the performance. Finally, similar to our work, memory networks (Prakash et al., 2017) have recently been used for diagnosis coding. However, we would like to note two significant differences between the memory network from Prakash et al. (2017) and our model. First, they don't use a matching network and their memories rely on extracting information about each label from Wikipedia. In contrast, our model does not use any auxiliary information. Second, they only evaluate on the 50 most frequent labels, while we evaluate on all the labels in the dataset.

3 Our Architecture

An overview of our model is shown in Figure 1. Our model architecture has two main components.

1. We augment a CNN with external memory over a support set S, which consists of a small subset of the training dataset. The model searches the support set to find similar examples with respect to the input instance. We make use of the homophily assumption that similar instances in the support set are coded

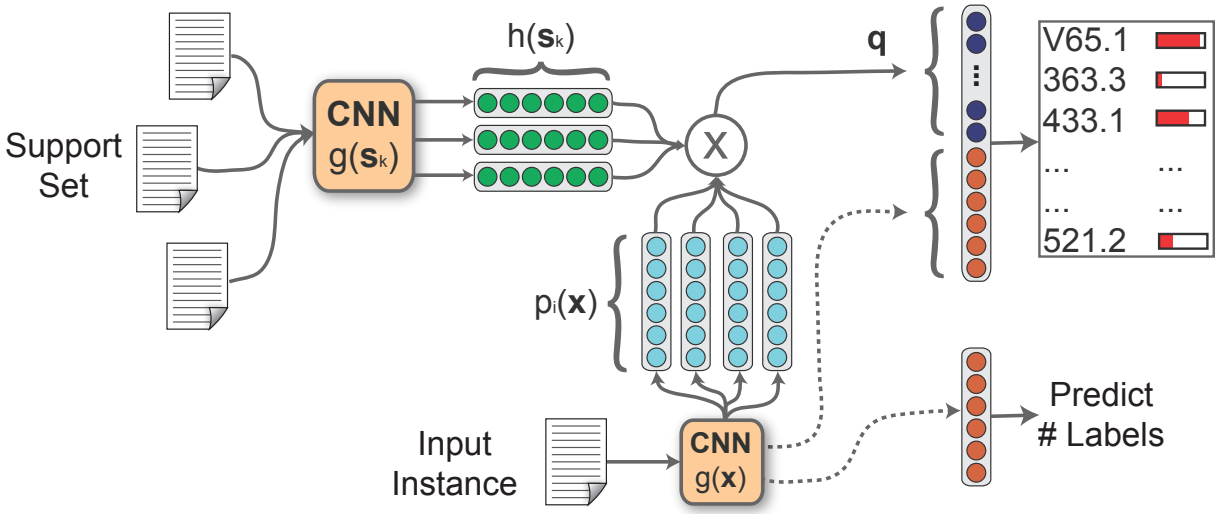


Figure 1: The matching CNN architecture. For each input instance, $\mathbf{x}$, we search a support set using different representations of $\mathbf{x}$ and use the similar support instances and auxiliary features to the output layer.

with similar labels. Therefore, we use the related support set examples as auxiliary features. The similar instances are chosen automatically by combining ideas from metric learning and neural attention. We emphasize that unlike in a traditional k-NN setup, we do NOT explicitly use the labels of the support set instances. The support set essentially enriches and complements the features derived from the input instance.

2. Rather than predicting labels by thresholding, we rank them and select the top k labels specific to each instance where k is predicted using an additional output unit (termed MetaLabeler). We train the MetaLabeler along with the classification loss using a multi-task training scheme.

Before we go into more specific details of our architecture, we introduce some notation. Let X represent the set of all training documents and $\mathbf{x}$ be an instance of X. Likewise, let S represent the set of support instances and s be an instance of S. We let L be the total number of unique labels. Our full model is described in following subsections.

3.1 Convolutional Neural Networks

We use a CNN to encode each document following what is now a fairly standard approach consisting of an embedding layer, a convolution layer, a max-pooling layer, and an output layer (Collobert et al., 2011; Kim, 2014). However, in our architecture, the CNN additionally aids in getting intermediate representations for the multi-head matching network component (Section 3.2).

Intuitively, CNNs make use of the sequential nature of text, where a non-linear function is applied to *region vectors* formed from vectors of words in short adjacent word sequences. Formally, we represent each document as a sequence of word vectors, $[\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n]$, where $\mathbf{w}_i \in \mathbb{R}^d$ represents the vector of the i-th word in the document. The region vectors are formed by concatenating each window of s words, $\mathbf{w}_{i-s+1}|| \ldots ||\mathbf{w}_i$, into a local region vector $\mathbf{c}_j \in \mathbb{R}^{sd}$. Next, $\mathbf{c}_j$ is passed to a non-linear function

$$\hat{\mathbf{c}}_j = ReLU(\mathbf{W}\,\mathbf{c}_j + \mathbf{b}),$$

where $\mathbf{W} \in \mathbb{R}^{v \times sd}$, $\mathbf{b} \in \mathbb{R}^v$, and ReLU is a rectified linear unit (Glorot et al., 2011; Nair and Hinton, 2010). Each row of $\mathbf{W}$ represents a convolutional filter; so v is the total number of filters.

After processing each successive region vector, we obtain a document representation $\mathbf{D} = [\hat{\mathbf{c}}_1, \hat{\mathbf{c}}_2, \ldots, \hat{\mathbf{c}}_{n+s-1}]$ by concatenating each $\hat{\mathbf{c}}_j$ forming a matrix $\mathbf{D} \in \mathbb{R}^{v \times (n+s-1)}$. Each row of $\mathbf{D}$ is referred to as a *feature map*, formed by different convolutional filters. Unfortunately, this representation is dependent on the length of the document and we cannot pass it to an output layer. We use max-over-time pooling to create a fixed size vector

$$g(\mathbf{s}) = [\hat{c}^1_{max}, \hat{c}^2_{max}, \ldots, \hat{c}^q_{max}],$$

where $\hat{c}^j_{max} = \max(\hat{c}^j_1, \hat{c}^j_2, \ldots, \hat{c}^j_{n+s-1})$.

3.2 Multi-Head Matching Network

Using the support set and the input instance, our goal is to estimate $P(\mathbf{y}|\mathbf{x}, S)$. The support set S is chosen based on nearest neighbors and its selection process is discussed in Section 3.4. Among instances in S, our model finds informative support instances with respect to $\mathbf{x}$ and creates a feature vector using them. This feature vector is combined with the input instance to make predictions.

First, each support instance $\mathbf{s}_k \in S$ is projected into the support space using a simple single-layer feed forward NN as

$$h(g(\mathbf{s}_k)) = ReLU(\mathbf{W}_s\, g(\mathbf{s}_k) + \mathbf{b}_s),$$

where $\mathbf{W}_s \in \mathbb{R}^{z \times v}$ and $\mathbf{b}_s \in \mathbb{R}^z$. Likewise, we project each input instance $\mathbf{x}$ into the input space using a different feed forward neural network,

$$p_i(g(\mathbf{x})) = ReLU(\mathbf{W}^i_\alpha\, g(\mathbf{x}) + \mathbf{b}^i_\alpha),$$

where $\mathbf{W}^i_\alpha \in \mathbb{R}^{z \times v}$ and $\mathbf{b}^i_\alpha \in \mathbb{R}^z$. Compared to the support set neural network where we use only a single network, for the input instance we have u projection neural networks. This means we have u versions of $\mathbf{x}$, an idea that is similar to self-attention (Lin et al., 2017), where the model learns multiple representations of an instance. Here each $p_i(g(\mathbf{x}))$ represents a single "head" or representation of the input $\mathbf{x}$. Using different weight matrices, $[\mathbf{W}^1_\alpha, \ldots, \mathbf{W}^u_\alpha]$ and $[\mathbf{b}^1_\alpha, \ldots, \mathbf{b}^u_\alpha]$, we create different representations of $\mathbf{x}$ (multiple heads). For both the input multi-heads and the support instance projection, we note that the same CNN is used (also indicated in Figure 1) whose output is subject to the feed forward neural nets outlined thus far in this section.

Rather than searching for a single informative support instance, we search for multiple relevant support instances. For each of the u input instance representations, we calculate a normalized attention score

$$A_{i,k} = \frac{\exp(-d(p_i(g(\mathbf{x})),\, h(g(\mathbf{s}_k))))}{\sum_{\mathbf{s}_{k'} \in S} \left[\exp(-d(p_i(g(\mathbf{x})),\, h(g(\mathbf{s}_{k'}))))\right]}$$

where $A_{i,k}$ represents the score of the k-th support example with respect to the i-th input representation $p_i(g(\mathbf{x}))$ and

$$d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2^2,$$

is the square of the Euclidean distance between the input and support representations.

Next, the normalized scores are aggregated into a matrix $\mathbf{A} \in \mathbb{R}^{u \times |S|}$. Then, we create a feature vector

$$\mathbf{q} = vec(\mathbf{A}\,\mathbf{S}) \tag{1}$$

where $\mathbf{q} \in \mathbb{R}^{uz}$, vec is the matrix vectorization operator, and $\mathbf{S} \in \mathbb{R}^{|S| \times z}$ is the support instance CNN feature matrix whose i-th row is $h(g(\mathbf{s}_i))$ for $i = 1, \ldots, |S|$. Intuitively, multiple weighted averages of the support instances are created, one for each of the u input representations. The final feature vector,

$$\mathbf{h} = \mathbf{q}\,||\,g(\mathbf{x}), \tag{2}$$

is formed by concatenating the CNN representation of the input instance $\mathbf{x}$ and the support set feature vector $\mathbf{q}$.

Finally, the output layer for L labels involves computing

$$\hat{\mathbf{y}} = P(\mathbf{y}|\mathbf{x}, S) = \sigma(\mathbf{W}_c\,\mathbf{h} + \mathbf{b}_c) \tag{3}$$

where $\mathbf{W}_c \in \mathbb{R}^{L \times (uz+v)}$, $\mathbf{b}_c \in \mathbb{R}^L$, and σ is the sigmoid function. Because we use a sigmoid activation function, each label prediction ($\hat{y}_i$) is in the range from 0 to 1.

3.3 MetaLabeler

The easiest method to convert $\hat{\mathbf{y}}$ into label predictions is to simply threshold each element at 0.5. However, most large-scale multi-label problems are highly imbalanced. When training using binary cross-entropy, the threshold 0.5 is optimized for accuracy. Therefore, our predictions will be biased towards 0. A simple way to fix this problem is to optimize the threshold value for each label. Unfortunately, searching for the optimal threshold of each label is computational expensive in large label spaces. Here we train a regression based output layer

$$\hat{r} = ReLU(\mathbf{W}_r\, g(\mathbf{x}) + b_r)$$

where $\hat{r}$ estimates the number of labels $\mathbf{x}$ should be annotated with. At test time, we rank each label by its score in $\hat{\mathbf{y}}$. Next, $\hat{r}$ is rounded to the nearest integer and we predict the top $\hat{r}$ ranked labels.

3.4 Training

To train our model, we need to define two loss functions. First, following recent working on multi-label classification with neural net-

works (Nam et al., 2014b), we train using a multi-label cross-entropy loss. The loss is defined as

$$\mathcal{L}_c = \sum_{i=1}^{L} \big[-y_i \log(\hat{y}_i) - (1 - y_i) \log(1 - \hat{y}_i) \big],$$

which sums the binary cross-entropy loss for each label. The second loss is used to train the MetaLabeler for which we use the mean squared error

$$\mathcal{L}_{\mathbf{r}} = \|\mathbf{r} - \hat{\mathbf{r}}\|_2^2$$

where $\mathbf{r}$ is the vector of correct numbers of labels and $\hat{\mathbf{r}}$ is our estimate. We train these two losses using a multi-task learning paradigm (Collobert et al., 2011).

Similar to previous work with matching networks (Vinyals et al., 2016; Snell et al., 2017), "episode" or mini-batch construction can have an impact on performance. In the multi-label setting, episode construction is non-trivial. We propose a simple strategy for choosing the support set S which we find works well in practice. First, at the beginning of the training process we loop over all training examples and store $g(\mathbf{x})$ for every training instance. We will refer to this set of vectors as T. Next, for every step of the training process (for every mini-batch M), we search $T \setminus M$ to find the e nearest neighbors (using Euclidean distance) per instance to form our support set S. Likewise, we add e random examples from $T \setminus M$ to the support set. Therefore, our support set S contains up to $|M|e + e$ instances. The purpose of the random examples is to ensure the distance metric learned during training (captured by improving representations of documents as influenced by all network parameters) is robust to noisy examples.

3.5 Matching Network Interpretation

If we do not use the support set label vectors, then what is our network learning? To answer this question we directly compare the matching network formulation to our method. Matching networks can be expressed as

$$\hat{\mathbf{y}} = \sum_{\mathbf{s}_k \in S} a(\mathbf{x}, \mathbf{s}_k)\, \mathbf{y}_{\mathbf{s}_k}$$

where $a(,)$ is the attention/distance learned between two instances, k indexes each support instance, and $\mathbf{y}_k$ is a one-hot encoded vector. $a(,)$ is equivalent to $A_{1,k}$ assuming we use a single head. Traditional matching networks use one-hot encoded vectors because they are evaluated on multi-class problems. EMR coding is a multi-label problem. Hence, $\mathbf{y}_k$ is a multi-hot encoded vector. Moreover, with thousands of labels, it is unlikely even for neighboring instance pairs to share many labels; this problem is not encountered in the multi-class setting. We overcome this issue by learning new output label vectors for each support set instance. Assuming a single head, our method can be re-written as

$$\hat{\mathbf{y}} = \sigma(\mathbf{W}_c^1\, g(\mathbf{x}) + \mathbf{b}_c + \sum_{\mathbf{s}_k \in S} a(\mathbf{x}, \mathbf{s}_k)\, \tilde{\mathbf{y}}_{\mathbf{s}_k}), \quad (4)$$

where $\tilde{\mathbf{y}_k}$ is the learned label vector for support instance s. Next, we define $\tilde{\mathbf{y}_k}$, the learned support set vectors, as

$$\tilde{\mathbf{y}}_{\mathbf{s}_k} = \mathbf{W}_c^2\, h(g(\mathbf{s}_k)) \quad (5)$$

where both $\mathbf{W}_c^1$ and $\mathbf{W}_c^2$ are submatrices of $\mathbf{W}_c$. Using this reformulation, we can now see that our method's main components (equations (1)-(3)) are equivalent to this more explicit matching network formulation (equations (4)–(5)). Intuitively, our method combines a traditional output layer – the first half of equation 4 – with a matching network where the support set label vectors are learned to better match the labels of their nearest neighbors.

4 Experiments

In this section we compare our work with prior state-of-the-art medical coding methods. In Section 4.1 we discuss the two publicly available datasets we use. Next, Section 4.2 describes the implementation details of our model. We summarize the various baselines and models we compare against in Section 4.3. The evaluation metrics are described in Section 4.4. Finally, we discuss how our method performs in Section 4.5.

4.1 Datasets

EMR data is generally not available for public use especially if it involves textual notes. Therefore, we focus on the publicly available Medical Information Mart for Intensive Care (MIMIC) datasets for benchmarking purposes. We evaluate using two versions of MIMIC: MIMIC II (Lee et al., 2011) and MIMIC III (Johnson et al., 2016), where the former is a relatively smaller and older dataset

	# Train	# Test	# Labels	LC	AI/L
MIMIC II	18822	2282	7042	36.7	118.8
MIMIC III	37016	2755	6932	13.6	80.8

Table 1: This table presents the number of training examples (# Train), the number of test examples (# Test), label cardinality (LC), and the average number of instances per label (AI/L) for the MIMIC II and MIMIC III datasets.

and the latter is the most recent version. Following prior work (Perotte et al., 2013; Vani et al., 2017), we use the free text discharge summaries in MIMIC to predict the ICD-9-CM[2] codes. The dataset statistics are shown in Table 1.

For comparison purposes, we use the same MIMIC II train/test splits as Perotte et al. (2013). Specifically, we use discharge reports collected from 2001 to 2008 from the intensive care unit (ICU) of the Beth Israel Deaconess Medical Center. Following Perotte et al. (2013), the labels for each discharge summary are extended using the parent of each label in label set. The parents are based on the ICD-9-CM hierarchy. We use the hierarchical label expansion to maximize the prior work we can compare against.

The MIMIC III dataset has been extended to include health records of patients admitted to the Beth Israel Deaconess Medical Center from 2001 to 2012 and hence provides a test bed for more advanced learning methods. Unfortunately, it does not have a standard train/test split to compare against prior work given we believe we are the first to look at it for this purpose. Hence, we use both MIMIC II and MIMIC III for comparison purposes. Furthermore, we do not use the hierarchical label expansion on the MIMIC III dataset.

Before we present our results, we discuss an essential distinction between the MIMIC II and MIMIC III datasets. Particularly, we are interested in the differences concerning label imbalance. From Table 1, we find that MIMIC III has almost twice as many examples compared to MIMIC II in the dataset. However, MIMIC II on average has more instances per label. Thus, although MIMIC III has more examples, each label is used fewer times on average compared to MIMIC II. The reason for this is because of how the label sets for each instance were extended using the ICD-9 hierarchy in MIMIC II.

[2]In 2015, a federal mandate was issued that requires the use of ICD-10 instead of ICD-9. However because of this recent change, ICD-10 training data is limited. Therefore, we use publicly available ICD-9 datasets for evaluation.

4.2 Implementation Details

Preprocessing: Each discharge summary was tokenized using a simple regex tokenization scheme (\w\w+). Also, each word/token that occurs less than five times in the training dataset was replaced with the *UNK* token.

Model Details: For our CNN, we used convolution filters of size 3, 4 and 5 with 300 filters for each filter size. We used 300 dimensional skip-gram (Mikolov et al., 2013) word embeddings pre-trained on PubMed. The Adam optimizer (Kingma and Ba, 2015) was used for training with the learning rate 0.0001. The mini-batch size was set to 4, e – the number of nearest neighbors per instance – was set to 16, and the number of heads (u) is set to 8. Our code is available at: `https://github.com/bionlproc/med-match-cnn`

4.3 Baseline Methods

In this paper, we focused on comparing our method to state-of-the-art methods for diagnosis code prediction such as grounded recurrent neural networks (Vani et al., 2017) (GRNN) and multi-label CNNs (Baumel et al., 2018). We also compare against traditional binary relevance methods where independent binary classifiers (L1-regularized linear models) are trained for each label. Next, we compare against hierarchical SVM (Perotte et al., 2013), which incorporates the ICD-9-CM label hierarchy. Finally, we also report the results of the traditional matching network with one modification: We train the matching network with the multi-label loss presented in Section 3.4 and threshold using the MetaLabeler described in Section 3.3.

We also present two versions of our model: *Match-CNN* and *Match-CNN Ens*. Match-CNN is the multi-head matching network introduced in Section 3. Match-CNN Ens is an ensemble that averages three Match-CNN models, each initialized using a different random seed.

4.4 Evaluation Metrics

We evaluate our method using a wide variety of standard multi-label evaluation metrics. We use the popular micro and macro averaged F1 measures to assess how our model (with the MetaL-

			F1		AUC (PR)		AUC (ROC)		P@k		R@k	
	Prec.	Recall	Micro	Macro	Micro	Macro	Micro	Macro	8	40	8	40
Flat SVM (Perotte et al., 2013)	**0.867**	0.164	0.276	–	–	–	–	–	–	–	–	–
Hier. SVM (Perotte et al., 2013)	0.577	0.301	0.395	–	–	–	–	–	–	–	–	–
Logistic (Vani et al., 2017)	0.774	0.395	0.523	0.042	0.541	0.125	0.919	0.704	0.913	0.572	0.169	0.528
Attn BoW (Vani et al., 2017)	0.745	0.399	0.52	0.027	0.521	0.079	0.975	0.807	0.912	0.549	0.169	0.508
GRU-128 (Vani et al., 2017)	0.725	0.396	0.512	0.027	0.523	0.082	0.976	**0.827**	0.906	0.541	0.168	0.501
BiGRU-64 (Vani et al., 2017)	0.715	0.367	0.485	0.021	0.493	0.071	0.973	0.811	0.892	0.522	0.165	0.483
GRNN-128 (Vani et al., 2017)	0.753	0.472	0.58	0.052	0.587	0.126	0.976	0.815	0.93	0.592	0.172	0.548
BiGRNN-64 (Vani et al., 2017)	0.761	0.466	0.578	0.054	0.589	0.131	0.975	0.798	0.925	**0.596**	0.172	0.552
CNN (Baumel et al., 2018) *	0.810	0.403	0.538	0.031	0.599	0.127	0.971	0.759	0.931	0.585	0.207	0.586
Matching Network *	0.439	0.388	0.412	0.014	0.394	0.034	0.893	0.551	0.793	0.427	0.172	0.425
Match-CNN (Ours)	0.605	0.561	0.582	0.064	0.612	0.148	**0.977**	0.792	0.930	0.586	0.207	0.590
Match-CNN Ens. (Ours)	0.616	**0.567**	**0.591**	**0.066**	**0.623**	**0.157**	**0.977**	0.793	**0.935**	0.594	**0.208**	**0.598**

Table 2: Results for the MIMIC II dataset. Models marked with * represent our custom implementations.

			F1		AUC (PR)		AUC (ROC)		P@k		R@k	
	P	R	Micro	Macro	Micro	Macro	Micro	Macro	8	40	8	40
Logistic (Vani et al., 2017) *	0.711	0.242	0.361	0.026	0.419	**0.147**	0.961	0.751	0.554	**0.211**	0.414	**0.686**
CNN (Baumel et al., 2018) *	**0.726**	0.246	0.367	0.021	0.376	0.095	0.942	0.697	0.534	0.196	0.395	0.636
Matching Network *	0.248	0.237	0.242	0.008	0.183	0.028	0.823	0.554	0.310	0.128	0.231	0.431
Match-CNN (Ours)	0.466	0.447	0.456	0.041	0.421	0.119	0.963	0.726	0.557	0.206	0.413	0.670
Match-CNN Ens. (Ours)	0.488	**0.449**	**0.468**	**0.043**	**0.441**	0.129	**0.965**	**0.760**	**0.570**	**0.211**	**0.422**	0.683

Table 3: Results for the MIMIC III dataset. Models marked with * represent our custom implementations.

abeler) performs when thresholding predictions. For problems with large labels spaces that suffer from significant imbalances in label distributions, the default threshold of 0.5 generally performs poorly (hence our use of MetaLabeler). To remove the thresholding effect bias, we also report different versions of the area under the precision-recall (PR) and receiver operating characteristic (ROC) curves. Finally, in a real-world setting, our system would not be expected to replace medical coders. We would expect medical coders to use our system to become more efficient in coding EMRs. Therefore, we would rank the labels based on model confidence and medical coders would choose the correct labels from the top few. To understand if our system would be useful in a real-world setting, we evaluate with precision at k (P@k) and recall at k (R@k). Having high P@k and R@k are critical to effectively encourage the human coders to use and benefit from the system.

4.5 Results

We show experimental results on MIMIC II in Table 2. Overall, our method improves on prior work across a variety of metrics. With respect to micro F1, we improve upon GRNN-128 by over 1%. Also, while macro-F1 is still low in general, we also improve macro F1 compared to state-of-the-art neural methods by more than 1%. In general, both micro and macro F1 are highly dependent on the thresholding methodology. Rather than thresholding at 0.5, we rank the labels and pick the top k based on a trained regression output layer. Can we do better than using a MetaLabeler? To measure this, we look at the areas under PR/ROC curves. Regarding micro and macro PR-AUC, we improve on prior work by $\approx 2.5\%$. This suggests that via better thresholding, the chances of improving both micro and macro F1 are higher for Match-CNN compared to other methods. Finally, we are also interested in metrics that evaluate how this model would be used in practice. We perform comparably with prior work on P@k. We show strong improvements in R@k with over a 4% improvement in R@40 compared to grounded RNNs and over 1% improvement when compared with Baumel et al. (2018). Our method also outperforms matching networks across every evaluation measure.

We present MIMIC III results in Table 3. We reiterate that MIMIC III does not have a standard train/test split. Hence we compare our model to our implementations of methods from prior ef-

	F1		P@k		R@k		AUC (PR)	
	Micro	Macro	8	40	8	40	Micro	Macro
Match-CNN	0.456	0.041	0.557	0.206	0.413	0.670	0.421	0.119
- Matching	0.429	0.034	0.534	0.196	0.395	0.636	0.376	0.095
- MetaLabler	0.391	0.026	0.557	0.206	0.413	0.670	0.421	0.119
- Multi-Head	0.450	0.034	0.548	0.202	0.403	0.656	0.417	0.113

Table 4: Ablation results for the MIMIC III dataset.

forts. For MIMIC III also we show improvements in multiple evaluation metrics. Interestingly, our method performs much better than the standard CNN on MIMIC III, compared to the relative performances of the two methods on MIMIC II. Match-CNN improves on CNN in R@40 by almost 5% on the MIMIC III dataset. The gain in R@40 is more than the 1% improvement found on MIMIC II. We hypothesize that the improvements on MIMIC III are because the label imbalance found in MIMIC III is higher than MIMIC II. Increased label imbalances mean more labels occur less often. Therefore, we believe our model works better with less training examples per label compared to the standard CNN model.

In Table 4 we analyze each component of our model using an ablation analysis on the MIMIC III dataset. First, we find that removing the matching component significantly effects our performance by reducing micro PR-AUC by almost 5%. Regarding micro and macro F1, we also notice that the MetaLabeler heuristic substantially improves on default thresholding (0.5). Finally, we see that the multi-head matching component provides reasonable improvements to our model across multiple evaluation measures. For example, P@8 and P@40 decrease by around 1% when we use attention with a single input representation.

5 Conclusion

In this paper, we introduce a semi-parametric multi-head matching network with a specific application to EMR coding. We find that by combining the non-parametric properties of matching networks with a traditional classification output layer, we improve metrics for both frequent and infrequent labels in the dataset. In the future, we plan to investigate three limitations of our current model.

1. We currently use a naive approach to choose the support set. We believe that improving the support set sampling method could substantially improve performance.

2. We hypothesize that a more sophisticated thresholding method could have a significant impact on the micro and macro F1 measures. As we show in Table 4, MetaLabeler outperforms naive thresholding strategies. However, given our method shows non-trivial gains in PR-AUC compared to micro/macro F1, we believe better thresholding strategies are a worthy avenue to seek improvements.

3. Both the MIMIC II and MIMIC III datasets have around 7000 labels but ICD-9-CM contains over 16000 labels and ICD-10-CM has nearly 70,000 labels. In future work, we believe significant attention should be given to zero-shot learning applied to EMR coding. To predict labels that have never occurred in the training dataset, we think it is vital to take advantage of the ICD hierarchy. Baker and Korhonen (2017) improve neural network training by incorporating hierarchical label information to create better weight initializations. However, this does not help with respect to zero-shot learning. If we can better incorporate expert knowledge about the label space, we may be able to infer labels we have not seen before.

Acknowledgments

Thanks to anonymous reviewers for their thorough reviews and constructive criticism that helped improve the clarity of the paper (especially leading to the addition of Section 3.5 in the revision). This research is supported by the U.S. National Library of Medicine through grant R21LM012274. We also gratefully acknowledge the support of the NVIDIA Corporation for its donation of the Titan X Pascal GPU used for this research.

References

Han Altae-Tran, Bharath Ramsundar, Aneesh S Pappu, and Vijay Pande. 2017. Low data drug discovery with one-shot learning. *ACS central science* 3(4):283–293.

Simon Baker and Anna Korhonen. 2017. Initializing neural networks for hierarchical multi-label text classification. *BioNLP 2017* pages 307–315.

Tal Baumel, Jumana Nassour-Kassis, Michael Elhadad, and Noemie Elhadad. 2018. Multi-label classification of patient notes a case study on icd code assignment. In *Proceedings of the 2018 AAAI Joint Workshop on Health Intelligence*.

Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. 2015. Sparse local embeddings for extreme multi-label classification. In *Advances in neural information processing systems*. pages 730–738.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075* .

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

Michelle Dougherty, Sandra Seabold, and Susan E White. 2013. Study reveals hard facts on CAC. *Journal of the American Health Information Management Association* 84(7):54–56.

Francisco Duarte, Bruno Martins, Cátia Sousa Pinto, and Mário J Silva. 2017. A deep learning method for icd-10 coding of free-text death certificates. In *Portuguese Conference on Artificial Intelligence*. Springer, pages 137–149.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier networks. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics. JMLR W&CP Volume*. volume 15, pages 315–323.

Himanshu Jain, Yashoteja Prabhu, and Manik Varma. 2016. Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pages 935–944.

Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. Mimic-iii, a freely accessible critical care database. *Scientific data* 3.

Rie Johnson and Tong Zhang. 2017. Deep pyramid convolutional neural networks for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 562–570.

Nitin Kamra, Umang Gupta, and Yan Liu. 2017. Deep generative dual memory network for continual learning. *arXiv preprint arXiv:1710.10368* .

Ramakanth Kavuluru, Anthony Rios, and Yuan Lu. 2015. An empirical evaluation of supervised learning approaches in assigning diagnosis codes to electronic medical records. *Artificial intelligence in medicine* 65(2):155–166.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1746–1751.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*. volume 2.

Joon Lee, Daniel J Scott, Mauricio Villarroel, Gari D Clifford, Mohammed Saeed, and Roger G Mark. 2011. Open-access mimic-ii database for intensive care research. In *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*. IEEE, pages 8315–8318.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. Automatic discovery and optimization of parts for image classification. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. 2017. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, pages 115–124.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*. pages 3111–3119.

Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *EMNLP*. The Association for Computational Linguistics, pages 1400–1409.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. pages 807–814.

Jinseok Nam, Jungi Kim, Eneldo Loza Mencía, Iryna Gurevych, and Johannes Fürnkranz. 2014a. Large-scale multi-label text classification - revisiting neural networks. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part II*. pages 437–452.

Jinseok Nam, Jungi Kim, Eneldo Loza Mencía, Iryna Gurevych, and Johannes Fürnkranz. 2014b. Large-scale multi-label text classificationrevisiting neural networks. In *Machine Learning and Knowledge Discovery in Databases*, Springer, pages 437–452.

Adler Perotte, Rimma Pivovarov, Karthik Natarajan, Nicole Weiskopf, Frank Wood, and Noémie Elhadad. 2013. Diagnosis code assignment: models and evaluation metrics. *Journal of the American Medical Informatics Association* 21(2):231–237.

John P Pestian, Christopher Brew, Paweł Matykiewicz, Dj J Hovermale, Neil Johnson, K Bretonnel Cohen, and Włodzisław Duch. 2007. A shared task involving multi-label classification of clinical free text. In *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing*. Association for Computational Linguistics, pages 97–104.

Yashoteja Prabhu and Manik Varma. 2014. Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 263–272.

Aaditya Prakash, Siyuan Zhao, Sadid A Hasan, Vivek V Datla, Kathy Lee, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2017. Condensed memory networks for clinical diagnostic inferencing. In *AAAI*. pages 3274–3280.

Anthony Rios and Ramakanth Kavuluru. 2013. Supervised extraction of diagnosis codes from emrs: role of feature selection, data selection, and probabilistic thresholding. In *Healthcare Informatics (ICHI), 2013 IEEE International Conference on*. IEEE, pages 66–73.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, Curran Associates, Inc., pages 4078–4088.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*. pages 2440–2448.

Farbound Tai and Hsuan-Tien Lin. 2012. Multilabel classification with principal label space transformation. *Neural Computation* 24(9):2508–2542.

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis P. Vlahavas. 2010. Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook*, pages 667–685.

Ankit Vani, Yacine Jernite, and David Sontag. 2017. Grounded recurrent neural networks. *arXiv preprint arXiv:1705.08557* .

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*. pages 3630–3638.

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

Chang Xu, Dacheng Tao, and Chao Xu. 2016. Robust extreme multi-label learning. In *KDD*. pages 1275–1284.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 1480–1489.

Hsiang-Fu Yu, Prateek Jain, Purushottam Kar, and Inderjit S. Dhillon. 2014. Large-scale multi-label learning with missing labels. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*. pages 593–601.

Danchen Zhang, Daqing He, Sanqiang Zhao, and Lei Li. 2017. Enhancing automatic icd-9-cm code assignment for medical texts with pubmed. *BioNLP 2017* pages 263–271.

Factors Influencing the Surprising Instability of Word Embeddings

Laura Wendlandt, Jonathan K. Kummerfeld and **Rada Mihalcea**
Computer Science & Engineering
University of Michigan, Ann Arbor
{wenlaura,jkummerf,mihalcea}@umich.edu

Abstract

Despite the recent popularity of word embedding methods, there is only a small body of work exploring the limitations of these representations. In this paper, we consider one aspect of embedding spaces, namely their stability. We show that even relatively high frequency words (100-200 occurrences) are often unstable. We provide empirical evidence for how various factors contribute to the stability of word embeddings, and we analyze the effects of stability on downstream tasks.

1 Introduction

Word embeddings are low-dimensional, dense vector representations that capture semantic properties of words. Recently, they have gained tremendous popularity in Natural Language Processing (NLP) and have been used in tasks as diverse as text similarity (Kenter and De Rijke, 2015), part-of-speech tagging (Tsvetkov et al., 2016), sentiment analysis (Faruqui et al., 2015), and machine translation (Mikolov et al., 2013a). Although word embeddings are widely used across NLP, their stability has not yet been fully evaluated and understood. In this paper, we explore the factors that play a role in the stability of word embeddings, including properties of the data, properties of the algorithm, and properties of the words. We find that word embeddings exhibit substantial instabilities, which can have implications for downstream tasks.

Using the overlap between nearest neighbors in an embedding space as a measure of stability (see section 3 below for more information), we observe that many common embedding spaces have large amounts of instability. For example, Figure 1 shows the instability of the embeddings obtained by training *word2vec* on the Penn Treebank (PTB) (Marcus et al., 1993). As expected, lower frequency words have lower stability and higher frequency words have higher stability. What is surprising however about this graph is the medium-frequency words, which show huge variance in stability. This cannot be explained by frequency, so there must be other factors contributing to their instability.

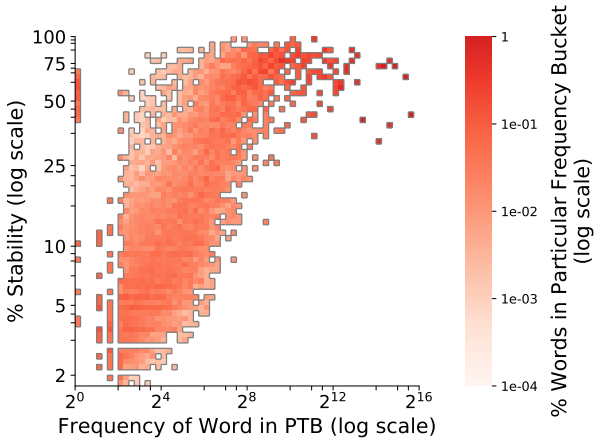


Figure 1: Stability of *word2vec* as a property of frequency in the PTB. Stability is measured across ten randomized embedding spaces trained on the training portion of the PTB (determined using language modeling splits (Mikolov et al., 2010)). Each word is placed in a frequency bucket (x-axis), and each column (frequency bucket) is normalized.

In the following experiments, we explore which factors affect stability, as well as how this stability affects downstream tasks that word embeddings are commonly used for. To our knowledge, this is the first study comprehensively examining the factors behind instability.

2 Related Work

There has been much recent interest in the applications of word embeddings, as well as a small, but growing, amount of work analyzing the properties of word embeddings.

Here, we explore three different embedding methods: PPMI (Bullinaria and Levy, 2007),

Proceedings of NAACL-HLT 2018, pages 2092–2102
New Orleans, Louisiana, June 1 - 6, 2018.

word2vec (Mikolov et al., 2013b), and GloVe (Pennington et al., 2014). Various aspects of the embedding spaces produced by these algorithms have been previously studied. Particularly, the effect of parameter choices has a large impact on how all three of these algorithms behave (Levy et al., 2015). Further work shows that the parameters of the embedding algorithm *word2vec* influence the geometry of word vectors and their context vectors (Mimno and Thompson, 2017). These parameters can be optimized; Hellrich and Hahn (2016) posit optimal parameters for negative sampling and the number of epochs to train for. They also demonstrate that in addition to parameter settings, word properties, such as word ambiguity, affect embedding quality.

In addition to exploring word and algorithmic parameters, concurrent work by Antoniak and Mimno (2018) evaluates how document properties affect the stability of word embeddings. We also explore the stability of embeddings, but focus on a broader range of factors, and consider the effect of stability on downstream tasks. In contrast, Antoniak and Mimno focus on using word embeddings to analyze language (e.g., Garg et al., 2018), rather than to perform tasks.

At a higher level of granularity, Tan et al. (2015) analyze word embedding spaces by comparing two spaces. They do this by linearly transforming one space into another space, and they show that words have different usage properties in different domains (in their case, Twitter and Wikipedia).

Finally, embeddings can be analyzed using second-order properties of embeddings (e.g., how a word relates to the words around it). Newman-Griffis and Fosler-Lussier (2017) validate the usefulness of second-order properties, by demonstrating that embeddings based on second-order properties perform as well as the typical first-order embeddings. Here, we use second-order properties of embeddings to quantify stability.

3 Defining Stability

We define *stability* as the percent overlap between nearest neighbors in an embedding space.[1] Given a word W and two embedding spaces A and B, take the ten nearest neighbors of W in both A and B. Let the stability of W be the percent overlap between these two lists of nearest neighbors. 100% stability indicates perfect agreement between the two embedding spaces, while 0% stability indicates complete disagreement. In order to find the ten nearest neighbors of a word W in an embedding space A, we measure distance between words using cosine similarity.[2] This definition of stability can be generalized to more than two embedding spaces by considering the average overlap between two sets of embedding spaces. Let X and Y be two sets of embedding spaces. Then, for every pair of embedding spaces (x, y), where $x \in X$ and $y \in Y$, take the ten nearest neighbors of W in both x and y and calculate percent overlap. Let the stability be the average percent overlap over every pair of embedding spaces (x, y).

Model 1	Model 2	Model 3
metropolitan	*ballet*	**national**
national	**metropolitan**	*ballet*
egyptian	bard	**metropolitan**
rhode	chicago	institute
society	**national**	glimmerglass
debut	*state*	*egyptian*
folk	*exhibitions*	intensive
reinstallation	*society*	jazz
chairwoman	whitney	*state*
philadelphia	*rhode*	*exhibitions*

Table 1: Top ten most similar words for the word *international* in three randomly intialized *word2vec* models trained on the NYT Arts Domain. Words in all three lists are in bold; words in only two of the lists are italicized.

Consider an example using this metric. Table 1 shows the top ten nearest neighbors for the word *international* in three randomly initialized *word2vec* embedding spaces trained on the NYT Arts domain (see Section 4.3 for a description of this corpus). These models share some similar words, such as *metropolitan* and *national*, but there are also many differences. On average, each pair of models has four out of ten words in common, so the stability of *international* across these three models is 40%.

The idea of evaluating ten best options is also found in other tasks, like lexical substitution (e.g., McCarthy and Navigli, 2007) and word associa-

[1] This metric is concurrently explored in work by Antoniak and Mimno (2018).

[2] We found comparable results for other distance metrics, such as l^1 norm, l^2 norm, and l^∞ norm, but we report results from cosine similarity to be consistent with other word embedding research.

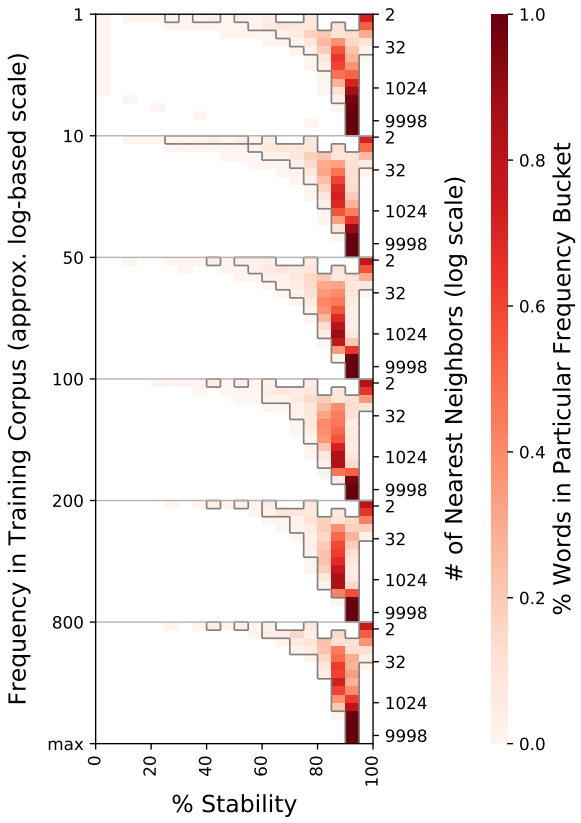


Figure 2: Stability of GloVe on the PTB. Stability is measured across ten randomized embedding spaces trained on the training data of the PTB (determined using language modeling splits (Mikolov et al., 2010)). Each word is placed in a frequency bucket (left y-axis) and stability is determined using a varying number of nearest neighbors for each frequency bucket (right y-axis). Each row is normalized, and boxes with more than 0.01 of the row's mass are outlined.

tion (e.g., Garimella et al., 2017), where the top ten results are considered in the final evaluation metric. To give some intuition for how changing the number of nearest neighbors affects our stability metric, consider Figure 2. This graph shows how the stability of GloVe changes with the frequency of the word and the number of neighbors used to calculate stability; please see the figure caption for a more detailed explanation of how this graph is structured. Within each frequency bucket, the stability is consistent across varying numbers of neighbors. Ten nearest neighbors performs approximately as well as a higher number of nearest neighbors (e.g., 100). We see this pattern for low frequency words as well as for high frequency words. Because the performance does not change substantially by increasing the number of nearest neighbors, it is computationally less intensive to use a small number of nearest neighbors. We choose ten nearest neighbors as our metric throughout the rest of the paper.

4 Factors Influencing Stability

As we saw in Figure 1, embeddings are sometimes surprisingly unstable. To understand the factors behind the (in)stability of word embeddings, we build a regression model that aims to predict the stability of a word given: (1) properties related to the word itself; (2) properties of the data used to train the embeddings; and (3) properties of the algorithm used to construct these embeddings. Using this regression model, we draw observations about factors that play a role in the stability of word embeddings.

4.1 Methodology

We use ridge regression to model these various factors (Hoerl and Kennard, 1970). Ridge regression regularizes the magnitude of the model weights, producing a more interpretable model than non-regularized linear regression. This regularization mitigates the effects of multicollinearity (when two features are highly correlated). Specifically, given N ground-truth data points with M extracted features per data point, let $\mathbf{x}_n \in \mathbb{R}^{1\times M}$ be the features for sample n and let $\mathbf{y} \in \mathbb{R}^{1\times N}$ be the set of labels. Then, ridge regression learns a set of weights $\mathbf{w} \in \mathbb{R}^{1\times M}$ by minimizing the least squares function with l^2 regularization, where λ is a regularization constant:

$$L(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(\mathbf{y}_n - \mathbf{w}^\top \mathbf{x}_n)^2 + \frac{\lambda}{2}||\mathbf{w}||^2$$

We set $\lambda = 1$. In addition to ridge regression, we tried non-regularized linear regression. We obtained comparable results, but many of the weights were very large or very small, making them hard to interpret.

The goodness of fit of a regression model is measured using the coefficient of determination R^2. This measures how much variance in the dependent variable $\mathbf{y}$ is captured by the independent variables $\mathbf{x}$. A model that always predicts the expected value of $\mathbf{y}$, regardless of the input features, will receive an R^2 score of 0. The highest possible R^2 score is 1, and the R^2 score can be negative.

Given this model, we create training instances by observing the stability of a large number of

words across various combinations of two embedding spaces. Specifically, given a word W and two embedding spaces A and B, we encode properties of the *word* W, as well as properties of the *datasets* and the *algorithms* used to train the embedding spaces A and B. The target value associated with this features is the stability of the word W across embedding spaces A and B. We repeat this process for more than 2,500 words, several datasets, and three embedding algorithms.

Specifically, we consider all the words present in all seven of the data domains we are using (see Section 4.3), 2,521 words in total. Using the feature categories described below, we generate a feature vector for each unique word, dataset, algorithm, and dimension size, resulting in a total of 27,794,025 training instances. To get good average estimates for each embedding algorithm, we train each embedding space five times, randomized differently each time (this does not apply to PPMI, which has no random component). We then train a ridge regression model on these instances. The model is trained to predict the stability of word W across embedding spaces A and B (where A and B are not necessarily trained using the same algorithm, parameters, or training data). Because we are using this model to learn associations between certain features and stability, no test data is necessary. The emphasis is on the model itself, not on the model's performance on a specific task.

We describe next each of the three main categories of factors examined in the model. An example of these features is given in Table 2.

Word Properties	
Primary part-of-speech	`Adjective`
Secondary part-of-speech	`Noun`
# Parts-of-speech	`2`
# WordNet senses	`3`
Syllables	`5`
Data Properties	
Raw frequency in corpus A	`106`
Raw frequency in corpus B	`669`
Diff. in raw frequency	`563`
Vocab. size of corpus A	`10,508`
Vocab. size of corpus B	`43,888`
Diff. in vocab. size	`33,380`
Overlap in corpora vocab.	`17%`
Domains present	`Arts, Europarl`
Do the domains match?	`False`
Training position in A	`1.02%`
Training position in B	`0.15%`
Diff. in training position	`0.86%`
Algorithm Properties	
Algorithms present	*`word2vec`*`, PPMI`
Do the algorithms match?	`False`
Embedding dimension of A	`100`
Embedding dimension of B	`100`
Diff. in dimension	`0`
Do the dimensions match?	`True`

Table 2: Consider the word *international* in two embedding spaces. Suppose embedding space A is trained using *word2vec* (embedding dimension 100) on the NYT Arts domain, and embedding space B is trained using PPMI (embedding dimension 100) on Europarl. This table summarizes the resulting features for this word across the two embedding spaces.

4.2 Word Properties

We encode several features that capture attributes of the word W. First, we use the primary and secondary part-of-speech (POS) of the word. Both of these are represented as bags-of-words of all possible POS, and are determined by looking at the primary (most frequent) and secondary (second most frequent) POS of the word in the Brown corpus[3] (Francis and Kucera, 1979). If the word is not present in the Brown corpus, then all of these POS features are set to zero.

To get a coarse-grained representation of the polysemy of the word, we consider the number of different POS present. For a finer-grained representation, we use the number of different WordNet senses associated with the word (Miller, 1995; Fellbaum, 1998).

We also consider the number of syllables in a word, determined using the CMU Pronuncing Dictionary (Weide, 1998). If the word is not present in the dictionary, then this is set to zero.

4.3 Data Properties

Data features capture properties of the training data (and the word in relation to the training data). For this model, we gather data from two sources: New York Times (NYT) (Sandhaus, 2008) and Europarl (Koehn, 2005). Overall, we consider seven domains of data: (1) NYT - U.S., (2) NYT - New York and Region, (3) NYT - Business, (4) NYT - Arts, (5) NYT - Sports, (6) All of the data from

[3]Here, we use the universal tagset, which consists of twelve possible POS: adjective, adposition, adverb, conjunction, determiner / article, noun, numeral, particle, pronoun, verb, punctuation mark, and other (Petrov et al., 2012).

Dataset	Sentences	Vocab. Size	Num. Tokens / Vocab. Size
NYT US	13,923	5,787	64.37
NYT NY	36,792	11,182	80.41
NYT Business	21,048	7,212	75.96
NYT Arts	28,161	10,508	65.29
NYT Sports	21,610	5,967	77.85
All NYT	121,534	24,144	117.98
Europarl	2,297,621	43,888	1,394.28

Table 3: Dataset statistics.

domains 1-5 (denoted "All NYT"), and (7) All of English Europarl. Table 3 shows statistics about these datasets. The first five domains are chosen because they are the top five most common categories of news articles present in the NYT corpus. They are smaller than "All NYT" and Europarl, and they have a narrow topical focus. The "All NYT" domain is more diverse topically and larger than the first five domains. Finally, the Europarl domain is the largest domain, and it is focused on a single topic (European Parliamentary politics). These varying datasets allow us to consider how data-dependent properties affect stability.

We use several features related to domain. First, we consider the raw frequency of word W in both the domain of data used for embedding space A and the domain of data for space B. To make our regression model symmetric, we effectively encode three features: the higher raw frequency (between the two), the lower raw frequency, and the absolute difference in raw frequency.

We also consider the vocabulary size of each corpus (again, symmetrically) and the percent overlap between corpora vocabulary, as well as the domain of each of the two corpora, represented as a bag-of-words of domains. Finally, we consider whether the two corpora are from the same domain.

Our final data-level features explore the role of curriculum learning in stability. It has been posited that the order of the training data affects the performance of certain algorithms, and previous work has shown that for some neural network-based tasks, a good training data order (curriculum learning strategy) can improve performance (Bengio et al., 2009). Curriculum learning has been previously explored for *word2vec*, where it has been found that optimizing training data order can lead to small improvements on common NLP tasks (Tsvetkov et al., 2016). Of the embedding algorithms we consider, curriculum learning only affects *word2vec*. Because GloVe and PPMI use the data to learn a complete matrix before building embeddings, the order of the training data will not affect their performance. To measure the effects of training data order, we include as features the first appearance of word W in the dataset for embedding space A and the first appearance of W in the dataset for embedding space B (represented as percentages of the total number of training sentences)[4]. We further include the absolute difference between these percentages.

4.4 Algorithm Properties

In addition to word and data properties, we encode features about the embedding algorithms. These include the different algorithms being used, as well as the different parameter settings of these algorithms. Here, we consider three embedding algorithms, *word2vec*, GloVe, and PPMI. The choice of algorithm is represented in our feature vector as a bag-of-words.

PPMI creates embeddings by first building a positive pointwise mutual information word-context matrix, and then reducing the dimensionality of this matrix using SVD (Bullinaria and Levy, 2007). A more recent word embedding algorithm, *word2vec* (skip-gram model) (Mikolov et al., 2013b) uses a shallow neural network to learn word embeddings by predicting context words. Another recent method for creating word embeddings, GloVe, is based on factoring a matrix of ratios of co-occurrence probabilities (Pennington et al., 2014).

For each algorithm, we choose common parameter settings. For *word2vec*, two of the parameters that need to be chosen are window size and minimum count. Window size refers to the maximum distance between the current word and the predicted word (e.g., how many neighboring words to consider for each target word). Any word appearing less than the minimum count number of times in the corpus is discarded and not considered in the *word2vec* algorithm. For both of these features, we choose standard parameter settings, namely, a window size of 5 and a minimum count of 5. For GloVe, we also choose standard parameters. We

[4] All *word2vec* experiments reported here are run in a multi-core setting, which means that these percentages are approximate. However, comparable results were achieved using a single-core version of *word2vec*.

Feature	Weight
Lower training data position of word W	-1.52
Higher training data position of W	-1.49
Primary POS = Numeral	1.12
Primary POS = Other	-1.08
Primary POS = Punctuation mark	-1.02
Overlap between corpora vocab.	1.01
Primary POS = Adjective	-0.92
Primary POS = Adposition	-0.92
Do the two domains match?	0.91
Primary POS = Verb	-0.88
Primary POS = Conjunction	-0.84
Primary POS = Noun	-0.81
Primary POS = Adverb	-0.79
Do the two algorithms match?	0.78
Secondary POS = Pronoun	0.62
Primary POS = Determiner	-0.48
Primary POS = Particle	-0.44
Secondary POS = Particle	0.36
Secondary POS = Other	0.28
Primary POS = Pronoun	-0.26
Secondary POS = Verb	-0.25
Number of *word2vec* embeddings	-0.21
Secondary POS = Adverb	0.15
Secondary POS = Determiner	0.14
Number of NYT Arts Domain	-0.14
Number of NYT All Domain	0.14
Number of GloVe embeddings	0.13
Number of syllables	-0.11

Table 4: Regression weights with a magnitude greater than 0.1, sorted by magnitude.

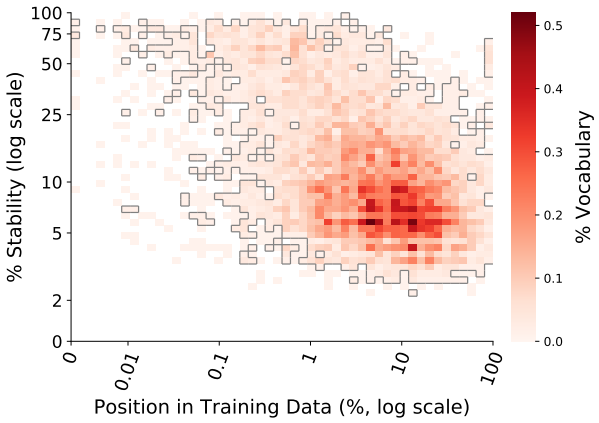
use 50 iterations of the algorithm for embedding dimensions less than 300, and 100 iterations for higher dimensions.

We also add a feature reflecting the embedding dimension, namely one of five embedding dimensions: 50, 100, 200, 400, or 800.

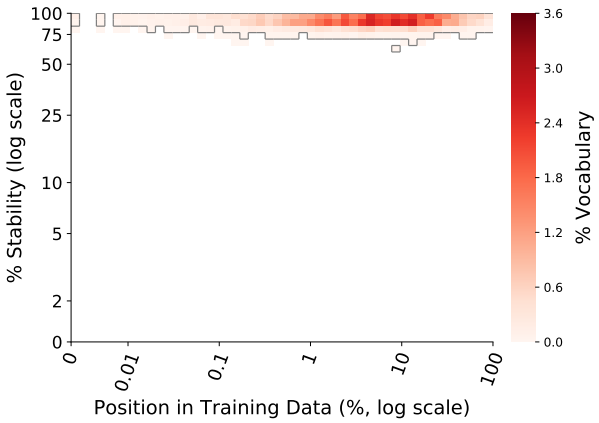
5 Lessons Learned: What Contributes to the Stability of an Embedding

Overall, the regression model achieves a coefficient of determination (R^2) score of 0.301 on the training data, which indicates that the regression has learned a linear model that reasonably fits the training data given. Using the regression model, we can analyze the weights corresponding to each of the features being considered, shown in Table 4. These weights are difficult to interpret, because features have different distributions and ranges. However, we make several general observations about the stability of word embeddings.

Observation 1. Curriculum learning is important. This is evident because the top two features (by magnitude) of the regression model capture where the word first appears in the training data. Figure 3 shows trends between training data position and stability in the PTB. This figure contrasts *word2vec* with GloVe (which is order invariant).



(a) *word2vec*



(b) GloVe

Figure 3: Stability of both *word2vec* and GloVe as properties of the starting word position in the training data of the PTB. Stability is measured across ten randomized embedding spaces trained on the training data of the PTB (determined using language modeling splits (Mikolov et al., 2010)). Boxes with more than 0.02% of the total vocabulary mass are outlined.

To further understand the effect of curriculum learning on the model, we train a regression model with all of the features except the curriculum learning features. This model achieves an R^2 score of 0.291 (compared to the full model's score of 0.301). This indicates that curriculum learning is a factor in stability.

Observation 2. POS is one of the biggest factors in stability. Table 4 shows that many of the top weights belong to POS-related features (both primary and secondary POS). Table 5 compares aver-

Primary POS	Avg. Stability
Numeral	47%
Verb	31%
Determiner	31%
Adjective	31%
Noun	30%
Adverb	29%
Pronoun	29%
Conjunction	28%
Particle	26%
Adposition	25%
Punctuation mark	22%

Table 5: Percent stability broken down by part-of-speech, ordered by decreasing stability.

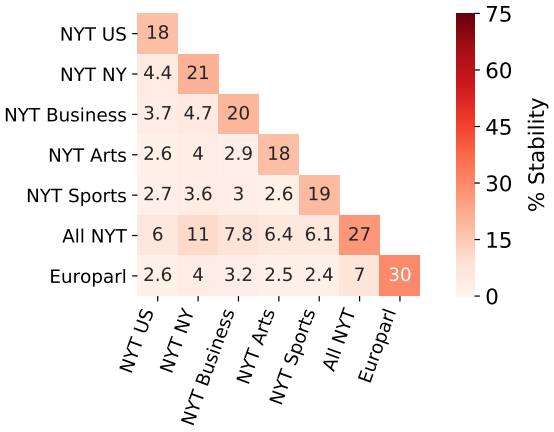


Figure 4: Percent stability broken down by domain.

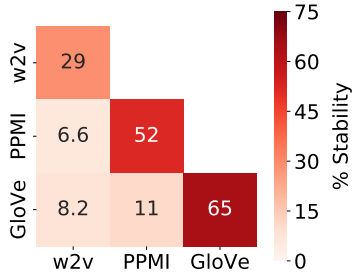


Figure 5: Percent stability broken down between algorithm (in-domain data only).

age stabilities for each primary POS. Here we see that the most stable POS are numerals, verbs, and determiners, while the least stable POS are punctuation marks, adpositions, and particles.

Observation 3. Stability within domains is greater than stability across domains. Table 4 shows that many of the top factors are domain-related. Figure 4 shows the results of the regression model broken down by domain. This figure shows the highest stabilities appearing on the diagonal of the matrix, where the two embedding spaces both belong to the same domain. The stabilities are substantially lower off the diagonal.

Figure 4 also shows that "All NYT" generalizes across the other NYT domains better than Europarl, but not as well as in-domain data ("All NYT" encompasses data from US, NY, Business, Arts, and Sports). This is true even though Europarl is much larger than "All NYT".

Observation 4. Overall, GloVe is the most stable embedding algorithm. This is particularly apparent when only in-domain data is considered, as in Figure 5. PPMI achieves similar stability, while *word2vec* lags considerably behind.

To further compare *word2vec* and GloVe, we look at how the stability of *word2vec* changes with the frequency of the word and the number of neighbors used to calculate stability. This is shown in Figure 6 and is directly comparable to Figure 2. Surprisingly, the stability of *word2vec* varies substantially with the frequency of the word. For lower-frequency words, as the number of nearest neighbors increases, the stability increases approximately exponentially. For high-frequency words, the lowest and highest number of nearest neighbors show the greatest stability. This is different than GloVe, where stability remains reasonably constant across word frequencies, as shown in Figure 2. The behavior we see here agrees with the conclusion of (Mimno and Thompson, 2017), who find that GloVe exhibits more well-behaved geometry than *word2vec*.

Observation 5. Frequency is not a major factor in stability. To better understand the role that frequency plays in stability, we run separate ablation experiments comparing regression models with frequency features to regression models without frequency features. Our current model (using raw frequency) achieves an R^2 score of 0.301. Comparably, a model using the same features, but with normalized instead of raw frequency, achieves a score of 0.303. Removing frequency from either regression model gives a score of 0.301. This indicates that frequency is not a major factor in stability, though normalized frequency is a larger factor than raw frequency.

Finally, we look at regression models using only frequency features. A model using only raw frequency features has an R^2 score of 0.008, while

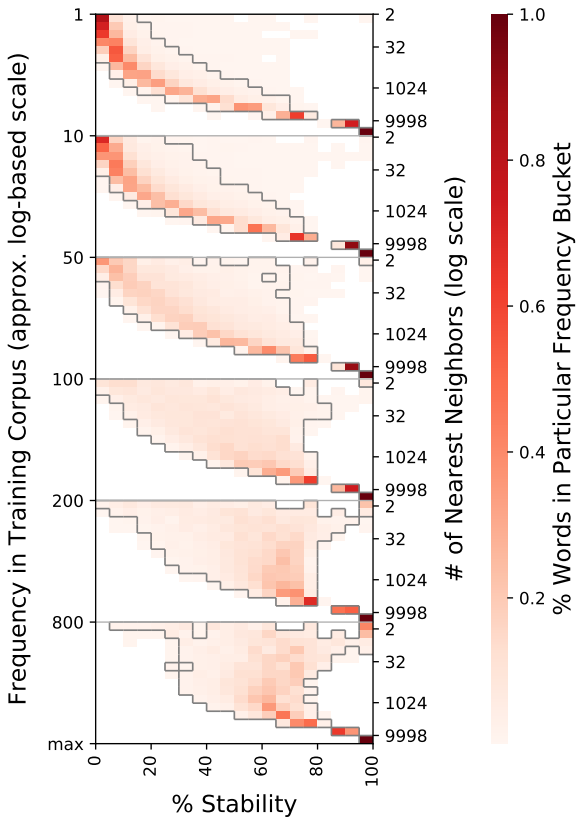


Figure 6: Stability of *word2vec* on the PTB. Stability is measured across ten randomized embedding spaces trained on the training data of the PTB (determined using language modeling splits (Mikolov et al., 2010)). Each word is placed in a frequency bucket (left y-axis) and stability is determined using a varying number of nearest neighbors for each frequency bucket (right y-axis). Each row is normalized, and boxes with more than 0.01 of the row's mass are outlined.

a model with only normalized frequency features has an R^2 score of 0.0059. This indicates that while frequency is not a major factor in stability, it is also not negligible. As we pointed out in the introduction, frequency does correlate with stability (Figure 1). However, in the presence of all of these other features, frequency becomes a minor factor.

6 Impact of Stability on Downstream Tasks

Word embeddings are used extensively as the first stage of neural networks throughout NLP. Typically, embeddings are initalized based on a vector trained with *word2vec* or GloVe and then further modified as part of training for the target task. We study two downstream tasks to see whether stability impacts performance.

Since we are interested in seeing the impact of

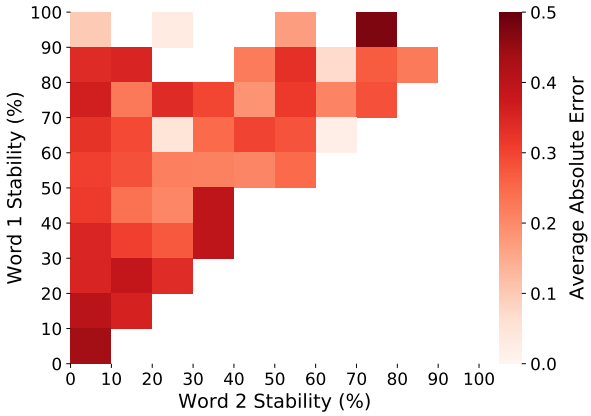


Figure 7: Absolute error for word similarity.

word vector stability, we choose tasks that have an intuitive evaluation at the word level: word similarity and POS tagging.

6.1 Word Similarity

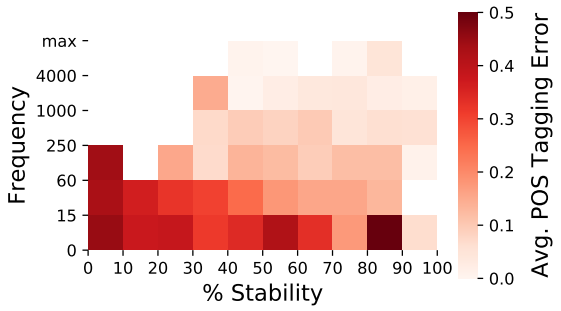
To model word similarity, we use 300-dimensional *word2vec* embedding spaces trained on the PTB. For each pair of words, we take the cosine similarity between those words averaged over ten randomly initialized embedding spaces.

We consider three datasets for evaluating word similarity: WS353 (353 pairs) (Finkelstein et al., 2001), MTurk287 (287 pairs) (Radinsky et al., 2011), and MTurk771 (771 pairs) (Halawi et al., 2012). For each dataset, we normalize the similarity to be in the range $[0, 1]$, and we take the absolute difference between our predicted value and the ground-truth value. Figure 7 shows the results broken down by stability of the two words (we always consider Word 1 to be the more stable word in the pair). Word similarity pairs where one of the words is not present in the PTB are omitted.

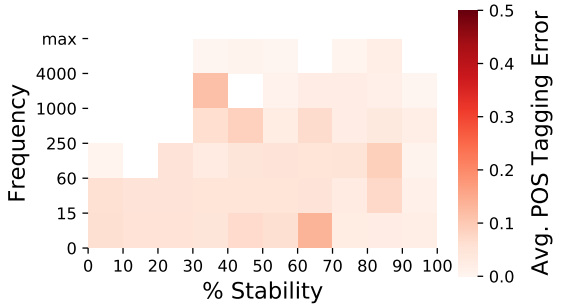
We find that these word similarity datasets do not contain a balanced distribution of words with respect to stability; there are substantially more unstable words than there are stable words. However, we still see a slight trend: As the combined stability of the two words increases, the average absolute error decreases, as reflected by the lighter color of the cells in Figure 7 while moving away from the (0,0) data point.

6.2 Part-of-Speech Tagging

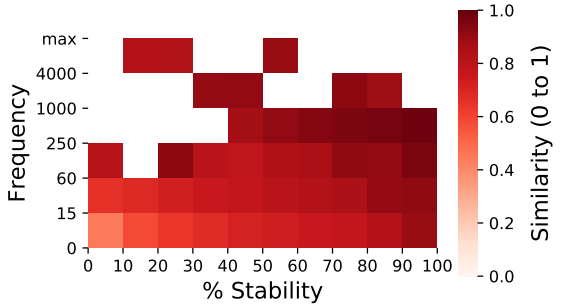
Part-of-speech (POS) tagging is a substantially more complicated task than word similarity. We use a bidirectional LSTM implemented using DyNet (Neubig et al., 2017). We train nine sets of



(a) POS error probability with fixed vectors.



(b) POS error probability when vectors may shift in training.



(c) Cosine similarity between original word vectors and shifted word vectors.

Figure 8: Results for POS tagging. (a) and (b) show average POS tagging error divided by the number of tokens (darker is more errors) while either keeping word vectors fixed or not during training. (c) shows word vector shift, measured as cosine similarity between initial and final vectors. In all graphs, words are bucketed by frequency and stability.

128-dimensional word embeddings with *word2vec* using different random seeds. The LSTM has a single layer and 50-dimensional hidden vectors. Outputs are passed through a *tanh* layer before classification. To train, we use SGD with a learning rate of 0.1, an input noise rate of 0.1, and recurrent dropout of 0.4.

This simple model is not state-of-the-art, scoring 95.5% on the development set, but the word vectors are a central part of the model, providing a clear signal of their impact. For each word, we group tokens based on stability and frequency. Figure 8 shows the results.[5] Fixing the word vectors provides a clearer pattern in the results, but also leads to much worse performance: 85.0% on the development set. Based on these results, it seems that training appears to compensate for stability. This hypothesis is supported by Figure 8c, which shows the similarity between the original word vectors and the shifted word vectors produced by the training. In general, lower stability words are shifted more during training.

Understanding how the LSTM is changing the input embeddings is useful information for tasks with limited data, and it could allow us to improve embeddings and LSTM training for these low-resource tasks.

7 Conclusion and Recommendations

Word embeddings are surprisingly variable, even for relatively high frequency words. Using a regression model, we show that domain and part-of-speech are key factors of instability. Downstream experiments show that stability impacts tasks using embedding-based features, though allowing embeddings to shift during training can reduce this effect. In order to use the most stable embedding spaces for future tasks, we recommend either using GloVe or learning a good curriculum for *word2vec* training data. We also recommend using in-domain embeddings whenever possible.

The code used in the experiments described in this paper is publicly available from `http://lit.eecs.umich.edu/downloads.html`.

Acknowledgments

We would like to thank Ben King and David Jurgens for helpful discussions about this paper, as well as our anonymous reviewers for useful feedback. This material is based in part upon work supported by the National Science Foundation (NSF #1344257) and the Michigan Institute for Data Science (MIDAS). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF or MIDAS.

[5] The unusual dark spot that occurs at medium-high stability and low frequency is caused primarily by words that have a substantially different POS distribution in the test data than in the training data.

References

Maria Antoniak and David Mimno. 2018. Evaluating the stability of embedding-based word similarities. *Transactions of the Association for Computational Linguistics (TACL)* 6:107–119. https://transacl.org/ojs/index.php/tacl/article/view/1202.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the International Conference on Machine Learning (ICML)*. pages 41–48. https://dl.acm.org/citation.cfm?id=1553380.

John A Bullinaria and Joseph P Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods* 39(3):510–526. https://link.springer.com/article/10.3758/BF03193020.

Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL: HLT)*. pages 1606–1615. http://www.aclweb.org/anthology/N15-1184.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library. https://onlinelibrary.wiley.com/doi/full/10.1002/9781405198431.wbeal1285.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the International Conference on World Wide Web (WWW)*. pages 406–414. https://dl.acm.org/citation.cfm?id=372094.

W Nelson Francis and Henry Kucera. 1979. Brown corpus manual. *Brown University* 2.

Nikhil Garg, Londa Schiebinger, Dan Jurafsky, and James Zou. 2018. Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences (PNAS)* https://doi.org/10.1073/pnas.1720347115.

Aparna Garimella, Carmen Banea, and Rada Mihalcea. 2017. Demographic-aware word associations. In *Proceedings of the International Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 2285–2295. http://www.aclweb.org/anthology/D17-1242.

Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. 2012. Large-scale learning of word relatedness with constraints. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. pages 1406–1414. https://dl.acm.org/citation.cfm?id=2339751.

Johannes Hellrich and Udo Hahn. 2016. Bad company-Neighborhoods in neural embedding spaces considered harmful. In *Proceedings of the International Conference on Computational Linguistics (COLING)*. pages 2785–2796. http://www.aclweb.org/anthology/C16-1262.

Arthur E Hoerl and Robert W Kennard. 1970. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics* 12(1):55–67. https://amstat.tandfonline.com/doi/abs/10.1080/00401706.1970.10488634.

Tom Kenter and Maarten De Rijke. 2015. Short text similarity with word embeddings. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM)*. pages 1411–1420. https://dl.acm.org/citation.cfm?id=2806475.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Machine Translation Summit X*. volume 5, pages 79–86.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics (TACL)* 3:211–225. https://www.transacl.org/ojs/index.php/tacl/article/view/570.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics* 19(2):313–330. https://dl.acm.org/citation.cfm?id=972475.

Diana McCarthy and Roberto Navigli. 2007. Semeval-2007 task 10: English lexical substitution task. In *Proceedings of the International Workshop on Semantic Evaluations (SemEval)*. pages 48–53. https://dl.acm.org/citation.cfm?id=1621483.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*. volume 2, page 3. https://www.isca-speech.org/archive/interspeech_2010/i10_1045.html.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013a. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168* https://arxiv.org/abs/1309.4168.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NIPS)*. pages 3111–3119. http://papers.nips.cc/paper/5021-distributed-representations-of-words-andphrases.

George A Miller. 1995. WordNet: A lexical database for English. *Communications of the ACM* 38(11):39–41. https://dl.acm.org/citation.cfm?id=219748.

David Mimno and Laure Thompson. 2017. The strange geometry of skip-gram with negative sampling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 2863–2868. http://www.aclweb.org/anthology/D17-1308.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. DyNet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980* https://arxiv.org/abs/1701.03980.

Denis Newman-Griffis and Eric Fosler-Lussier. 2017. Second-order word embeddings from nearest neighbor topological features. *arXiv preprint arXiv:1705.08488* https://arxiv.org/abs/1705.08488.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. *Proceedings of The International Conference on Language Resources and Evaluation (LREC)* pages 2089–2096. http://www.lrec-conf.org/proceedings/lrec2012/pdf/274_Paper.pdf.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: Computing word relatedness using temporal semantic analysis. In *Proceedings of the International Conference on World Wide Web (WWW)*. pages 337–346. https://dl.acm.org/citation.cfm?id=1963455.

Evan Sandhaus. 2008. The New York Times annotated corpus. *Linguistic Data Consortium, Philadelphia* 6(12):e26752. https://catalog.ldc.upenn.edu/ldc2008t19.

Luchen Tan, Haotian Zhang, Charles LA Clarke, and Mark D Smucker. 2015. Lexical comparison between Wikipedia and Twitter corpora by using word embeddings. In *Assocation for Computational Linguistics (ACL)*. pages 657–661. http://www.aclweb.org/anthology/P15-2108.

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Brian MacWhinney, and Chris Dyer. 2016. Learning the curriculum with Bayesian optimization for task-specific word representation learning. *Association for Computational Linguistics (ACL)* pages 130–139. http://anthology.aclweb.org/P/P16/P16-1013.pdf.

Robert L Weide. 1998. The CMU pronouncing dictionary http://www.speech.cs.cmu.edu/cgibin/cmudict.

Mining Evidences for Concept Stock Recommendation

Qi Liu and Yue Zhang
Singapore University of Technology and Design,
8 Somapah Road, Singapore 487372
{qi_liu, yue_zhang}@sutd.edu.sg

Abstract

We investigate the task of mining relevant stocks given a topic of concern on emerging capital markets, for which there is lack of structural understanding. Deep learning is leveraged to mine evidences from large scale textual data, which contain valuable market information. In particular, distributed word similarities trained over large scale raw texts are taken as a basis of relevance measuring, and deep reinforcement learning is leveraged to learn a strategy of topic expansion, given a small amount of manually labeled data from financial analysts. Results on two Chinese stock market datasets show that our method outperforms a strong baseline using information retrieval techniques.

1 Introduction

Stock prices are affected by events. For example, recent announcement of a state plan to build a new economic region, Xiong'an near Beijing, by the Chinese government has led to the rise of hundreds of stocks, which can directly or indirectly benefit from the plan. As a second example, the winning of a lawsuit against IP (Intellectual Property) breach can strengthen investors' confidence on technological and entertainment companies. We refer to the topics or themes of such events (e.g. Xiong'an and IP) as *concepts* and their relevant stocks as *concept stocks*. Given a news event, it can be highly useful for investors to find a list of relevant concept stocks for making investment decisions.

For popular concepts, lists of relevant concept stocks can be found from analyst reports from financial websites. On the other hand, concepts are dynamic and flexible. In addition, insights can be relatively scarce for emerging capital markets, such as the Chinese market, which had been closed to foreign investments before 2015. It is therefore a challenging research question how to automatically find out potentially relevant stocks given a topic of interest, from a large market of multi-thousand equities.

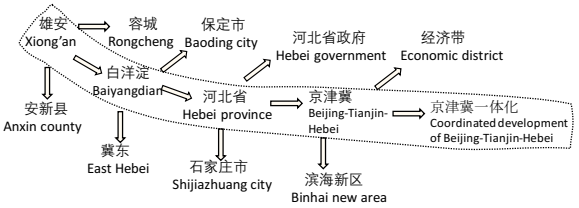


Figure 1: Concept relatedness. $A \Rightarrow B$ indicates that B is one of the 3 most related concepts to A.

Intuitively, evidences between concepts and stocks exist in text documents over the Internet. For example, news articles report events and companies involved. In addition, company filings such as annual/quarter reports contain factual knowledge about stocks, which can also be useful background information. For example, knowing that a company invests heavily on research is useful for correlating the company with IP-protection laws. Such evidence-mining process can involve multiple steps. As shown in Figure 1, starting from concept, Xiong'an, one might learn that the new economical region is located in the Baiyangdian area, which is further located in Hebei province. By further reading, one can infer that the new economic region is related with the coordinated development plan for the Beijing-Tianjin-Hebei region, and therefore benefit a wider range of stocks.

Based on the intuition above, we build a neural model for mining evidences for concept stock recommendation. The basis of our model is distributed similarities between concepts and stocks, obtained from embeddings trained over large-scale raw documents. Embedding similarities encode correlations from direct narrative evidence within context windows. To further include a multi-step

Proceedings of NAACL-HLT 2018, pages 2103–2112
New Orleans, Louisiana, June 1 - 6, 2018.

evidence mining, we build an iterative model for *concept expansion*, augmenting a given concept by iteratively adding more relevant concepts from background documents. As demonstrated in Figure 1, this process can be ambiguous, since there can be multiple directions for further reading given a set of concepts. We leverage a small amount of manually labeled data, downloaded from financial analysis websites, for guiding evidence mining.

In particular, we take a reinforcement learning method, which regards the evidence mining process as a decision process. The starting point is a given input concept, such as Xiong'an or Electronic Vehicle. At each step, a decision is made to stop further reading, or to continue adding related concepts to the set of concepts being considered. Existing concepts can also be removed from further consideration. Documents that discuss each concept are used to support the decision. After the process stops, relevant stocks to the set of concepts are recommended. The decision process is guided using a neural network model structure, trained with a loss function over the quality of the finally recommended stocks.

Results on two Chinese datasets show that our method outperforms a strong ranking-based baseline, which utilizes only direct evidences. Our method can be easily adapted for other markets given the availability of a small amount of training data. Our code is released[1].

2 Related Work

Our work is related to information retrieval and query expansion, where a concept can be regarded as a query and relevant stocks can be regarded as retrieved results. We rely on external evidence for correlating concepts and stocks.

Ranking is an important problem in information retrieval. We focus on ranking using neural models here. One line of work (Shen et al., 2014a,b) models queries and documents using convolutional neural network and ranks the documents pair-wise or list-wise. Another related method (Cao et al., 2015) adopts recursive neural networks to rank sentences for multi-document summarization. These methods requires massive annotated data, which is expensive to obtain for concept stock recommendation.

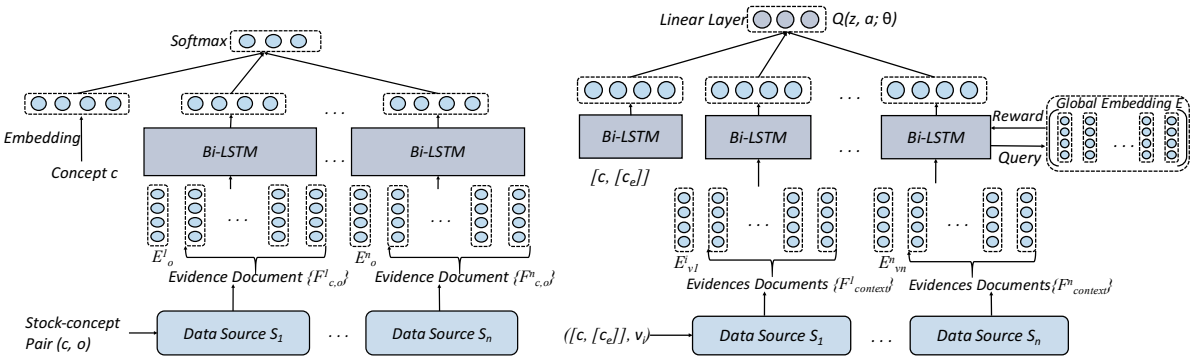
Query Expansion: One line of work (Cao et al., 2008; Preston and Colman, 2000) utilizes a feedback-based relevance model to expand queries. Another line applies language modeling to estimate conditional probabilities of concepts given a query, and expands the query with the most probable concepts (Bai et al., 2005; Carpineto and Romano, 2012). Recently, word embeddings are adopted for query expansion (Kuzi et al., 2016; Diaz et al., 2016). Our framework belongs to this line of work with a difference that we use reinforcement learning to dynamically expand queries instead of following handcrafted rules such as using k-nearest neighbors.

Reinforcement Learning: Our work aligns with existing work using reinforcement learning to collect evidences. Narasimhan et al. (2016) utilize external evidence to improve information extraction. While the work requires handcrafted features, our model uses dense embedding features. Athukorala et al. (2016) devise an interactive search engine balancing exploration and exploitation. Their work relies on user interaction to make decisions. In contrast, our work does not rely on active feedback, which can be expensive to obtain under our settings. Rodrigo and Cho (2017) introduce a query reformulation system based on reinforcement learning that rewrites a long and complex query to maximize the number of relevant documents returned. Differently, we do not assume complex queries and focus on recommending relevant stocks in our system. Zhong et al. (2017) solves a different problem, i.e. translating natural language questions to corresponding SQL queries.

3 Problem Definition

Our task is to find stocks relevant to a concept according to a variety of data sources, such as news, tweets and company files. Formally, given a concept c, a set of m stocks $\{o_i\}_{i=1}^{m}$ and n data sources $\{S_i\}_{i=1}^{n}$, where each S_i is a set of documents $\{D_j^i\}_{j=1}^{|S_i|}$ and each D_j^i is a sequence of words $w_1, w_2 ... w_{|D_j^i|}$, we assume the relevant stocks of the concept are revealed in the data sources (e.g. we discover PetroChina as a concept stock of 'petroleum' from the document 'PetroChina acquires Keppel's entire stake in Singapore Petroleum') and the task is to automatically discover these relations and select a subset of stocks as c's concept stocks based on the data sources $\{S_i\}_{i=1}^{n}$.

[1] https://github.com/leuchine/concept-stock-recommendation



(a) Ranking Baseline (b) Reinforcement Learning for Query Expansion

Figure 2: Concept Stock Recommendation Models

4 Representation

Motivated by the success of embedding-based models (Mikolov et al., 2013; Pennington et al., 2014) in capturing semantic regularities, we use embeddings to represent concepts, stocks and documents.

In particular, we adopt Chinese word segmentation (Yang et al., 2017) to obtain words from documents. Doc2Vec (Le and Mikolov, 2014) is then used on the documents of each data source S_i to obtain a local word embedding matrix E^i and a local document embedding matrix F^i, where each column of E^i (F^i) corresponds to a word (document) vector representation of S_i. In particular, we use embeddings, E^i_c and E^i_o as the local concept representation of c and the local stock representation of o in data source S_i, respectively. Furthermore, we obtain a global word embedding matrix E by averaging the local embedding matrices, $E^1 ... E^n$, where E_c and E_o are regarded as the global concept representation of c and the global stock representation of o, respectively.

We propose a ranking baseline and a reinforcement learning model for query expansion based on these representations.

5 Ranking Baseline

Inspired by Shen et al. (2014a; 2014b), our ranking baseline discriminatively projects the representations of concepts and evidences of stocks into a semantic space for measuring their relevances.

Mining Evidences: Formally, given a concept c and a stock o, we consult the data sources, retrieving the set of documents $\{D^i_{c,o}\}$ most relevant to (c, o) from each data source S_i as evidences.

To obtain evidences, we use c's local embedding E^i_c and o's local embedding E^i_o for representing the stock-concept pair (c, o). Cosine similarities are calculated between $E^i_c + E^i_o$ and each column of F^i for measuring the semantic relatedness of each document to (c, o). Suppose that the columns are normalized, the scores are calculated as:

$$score(\{D^i_j\}_{j=1}^{|S_i|}) = (F^i)^T(E^i_c + E^i_o) \quad (1)$$

q (q is set as 5 empirically) documents $\{D^i_{c,o}\}$ with the maximum scores are selected as evidences from each S_i. When $|F^i|$ is large, we use approximate k-nearest-neighbor algorithms, namely Locality Sensitive Hashing (Datar et al., 2004), to improve efficiency.

Learning to Rank o Given c: The overall framework for measuring relevances is shown in Figure 2 (a). Given a concept c and stock o, for each data source S_i, the local stock representation E^i_o and the local document representations of the q most relevant documents, denoted as $\{F^i_{c,o}\}$, are sequentially fed into Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) to acquire a semantic representation of the evidences.

A bidirectional extension (Graves and Schmidhuber, 2005) is applied to capture semantics both left-to-right and right-to-left. As a result, two sequences of hidden states are obtained, i.e. $\overrightarrow{h}_1, \overrightarrow{h}_2 ... \overrightarrow{h}_{q+1}$ and $\overleftarrow{h}_1, \overleftarrow{h}_2 ... \overleftarrow{h}_{q+1}$. We concatenate $\overrightarrow{h}_t$ and $\overleftarrow{h}_t$ at each time step to obtain the final hidden states $h_1, h_2 ... h_{q+1}$.

Average pooling (Boureau et al., 2010) is applied on the hidden states $h_1, h_2 ... h_{q+1}$ to obtain

evidence representation $I^i_{c,o}$,

$$I^i_{c,o} = \frac{\sum_{t=1}^{q+1} h_t}{q+1} \quad (2)$$

We concatenate all $I^i_{c,o}$ with the global concept representation of c (i.e. the average of local representations, $E^1_c ... E^n_c$) and feed the result to a softmax layer to obtain the probability of a stock o being c's concept stock, denoted as $p(o|c)$. Given a concept c, all stocks are ranked by the probabilities.

Given a set of gold-standard concept stock data, supervised learning is conducted to learn $p(o|c)$. The loss function is defined as:

$$E_{(c,o,y)}[-log\, p(o|c)^y - log\, (1-p(o|c))^{(1-y)}] \quad (3)$$

Here y is 1 when o is a concept stock of c, and 0 otherwise. Equation 3 maximizes $p(o|c)$ $(1 - p(o|c))$ when $y = 1$ $(y = 0)$. AdaGrad (Duchi et al., 2011) is applied to update parameters.

6 Recommendation via Query Expansion

The ranking baseline can require large amounts of annotated data to deliver satisfying performance (Shen et al., 2014a,b), which can be costly. In addition, the algorithm has to deal with highly imbalanced datasets, since there are thousands of stocks in a stock market, but only a few are related to a concept c, which greatly harms the performance of discriminatively trained algorithms (Wu and Chang, 2003).

We take a different approach, utilizing the same data sources and representations as the ranking baseline. To better leverage a small amount of supervision data, we apply reinforcement learning to expand the query concept c, consulting supporting evidences from the data sources $\{S_i\}_{i=1}^m$. We leverage embedding similarities as a basis for concept-stock relatedness. The advantage is that embeddings can be trained over large scale raw texts unsupervisedly, without the need for manually labeled stock lists.

6.1 Direct Semantic Relatedness

Embeddings represent similarities between concepts and stocks if they co-exist literally in a context window during embedding training. As a result, irrelevant (relevant) stocks are less (more) similar to the concept c, since they infrequently (frequently) co-occur, which alleviates the problem brought by imbalanced datasets in that irrelevant stocks can be spotted at ease.

Global representations of c and o is utilized to obtain a direct relevance score $\hat{f}(c, o)$:

$$\hat{f}(c, o) = E_c \cdot E_o, \quad (4)$$

where $\cdot$ denotes the dot product operation. The stocks are ranked by $\hat{f}(c, o)$ and concept stocks are these with the maximum cosine similarities.

6.2 Indirect Semantic Relatedness by Query Expansion

While $\hat{f}(c, o)$ measures direct relevance between c and o in embedding contexts, we want to find those o' that are indirectly relevant to c by reasoning as shown in Figure 1. Query expansion (Kuzi et al., 2016; Diaz et al., 2016) is used to this end. One naive baseline is expanding the concept c with its k-nearest-neighbor concepts, denoted as $[c_e]$, from global matrix E measured by cosine similarity. Relevance between the expanded concepts $[c, [c_e]]$ and o is calculated as:

$$\begin{aligned}\hat{f}([c, [c_e]], o) &= E_{[c,[c_e]]} \cdot E_o \\ &= (E_c + \sum_{c_e \in [c_e]} E_{c_e}) \cdot E_o \end{aligned} \quad (5)$$

We define $E_{[c,[c_e]]}$ as the addition of E_c and each E_{c_e}. The baseline is relatively inflexible since a fixed number of k expansion concepts are selected for all c. In contrast, the reasoning procedures shown in Figure 1 can take an arbitrary number of steps. Besides, the naive baseline does not incorporate supervision, thus being unable to decide whether the selected concepts are beneficial for concept stock recommendation.

We use reinforcement learning to tackle this issue, directly learning how to expand queries from a few labeled cases. Given c, our method works iteratively, expanding the concept until it expects further expansions are not desired. For each candidate concept to expand c, a decision is made by the model on whether it will improve, worsen or have no effect on recommendation accuracies.

Based on these, we model query expansion with a Markov Decision Process (MDP) to discriminatively select expansion concepts for c to maximize recommendation accuracies, while requiring much less training data compared to the ranking baseline.

The overall framework is shown in Figure 2 (b). Formally, a MDP is a list $[Z, A, T, R]$, where $Z = \{z\}$ is a set of states, $A = \{a\}$ is a set of actions, $T(z, a)$ is a transition function, which determines the next state $z' = T(z, a)$ after performing action a on z, and R is a reward function. We describe each in detail below:

States: Each state z is a list of lists:

$$z = [[c, [c_e]], [v^1, \{F^1_{context}\}]...[v^n, \{F^n_{context}\}]], \quad (6)$$

where $[c, [c_e]]$ consists of the input concept c and its expansion concept list $[c_e]$ so far. In the start state, $[c_e]$ is empty, and thus $[c, [c_e]] = [c, [\,]]$. $[v^i, \{F^i_{context}\}]$ consists of a new candidate concept v^i and its supporting evidences $\{F^i_{context}\}$. Since globally trained embeddings underperform locally trained embeddings (Diaz et al., 2016) for query expansion, we use local embeddings E^i to suggest candidate concepts v^i, instead of using global embedding E. v^i is obtained by finding the most similar concept of $[c, [c_e]]$ from local embeddings E^i. $\{F^i_{context}\}$ is the document representations of the q most relevant documents to $([c, [c_e]], v^i)$ as evidences. Formally, $\{F^i_{context}\}$ is the document representations of the q documents with the maximum scores, $score'(\{D^i_j\}^{|S_i|}_{j=1}) = (F^i)^T(E^i_{[c,[c_e]]} + E^i_{v^i})$. As a result, at each state, we have n candidate concepts $v^1...v^n$. The neural agent chooses at most one concept to be added to $[c_e]$ based on the evidences.

Action: The agent can take four types of actions. (1) Add one of n candidate concepts to $[c_e]$ (2) Reject all n candidate concepts. (3) Remove the last added concept from $[c_e]$ (4) Stop the process.

State Transition: After taking an action a on a state z, a new state z' is yielded by the transition function $T(z, a)$. If one of the candidate concepts v^i is chosen, v^i is added to $[c_e]$. In addition, the new state z' is obtained by updating $[v^1, \{F^1_{context}\}]...[v^n, \{F^n_{context}\}]$ by finding the most similar concepts of the new $[c, [c_e]']$ and their supporting evidences among the local embeddings. If action (2) is chosen, $[c, [c_e]]$ remains, while the $v^1...v^n$ is replaced with the second most similar concepts of $[c, [c_e]]$ among the local embeddings, and the process repeats until action (1), (3) or (4) is chosen. If (3) is chosen, the last added concept is removed from $[c_e]$, and $[v^1, \{F^1_{context}\}]...[v^n, \{F^n_{context}\}]$ are updated according to the new $[c, [c_e]']$. If (4) is chosen, the query expansion process finishes. The final $[c, [c_e]]$ is the result of query expansion.

Neural Agent: Given a state z, the neural agent chooses one action to take among the four types of actions. To this end, $[c, [c_e]]$ and each $[v^i, \{F^i_{context}\}]$ are fed into separate Bi-LSTM to obtain a concept representation and candidate concept representations, respectively. We further concatenate these representations and use a linear layer to obtain Q-values $Q(z, a; \theta)$ for each action a (Sutton and Barto, 1998). Note that we do not use softmax to normalize the Q-values since Q-value is the expectation of discounted sums of rewards by definition instead of probabilities. The action with the maximum Q-value is chosen.

Reward: A reward r is associated at each step specified by the reward function R, which evaluates the goodness of action a on state z. We use the difference of mean average precision (MAP) (Christopher et al., 2008) before and after an action a as the reward function:

$$R(z, a, z') = MAP(z') - MAP(z), \quad (7)$$

where MAP is defined as:

$$MAP(z) = \frac{1}{|\omega(c)|} \sum_{o \in \omega(c)} Precision@rank(o; z, E) \quad (8)$$

and

$$Precision@K = \frac{\sum_{o' \in \omega(c)} \mathbb{1}(rank(o'; z, E) \leq K)}{K} \quad (9)$$

$\omega(c)$ is the set of concept stocks of the concept c in training data. $rank(o; z, E)$ is the rank of the stock o, which is calculated by utilizing $[c, [c_e]]$ of z and global embedding E to rank all stocks using Equation 5. $\mathbb{1}$ is the indicator function. Therefore, MAP measures the goodness of the ranking, which is large if the stocks in $\omega(c)$ are ranked higher compared to the others. Reward r is positive if $[c, [c_e]']$ of z' can rank stocks better compared to $[c, [c_e]]$ of z and negative otherwise.

We choose MAP based on two reasons: (1) MAP provides a measure of quality, which has been shown to have good discrimination and stability. Besides, MAP is roughly the average area under the precision-recall curve for a set of queries (Christopher et al., 2008). Thus, optimizing MAP can indirectly improve both precision and recall. (2) MAP provides smoother scores than other metrics such as Precision@K and Recall@K.

In summary, at each step, the MDP framework chooses an action a based on a state z, obtaining a

Algorithm 1 Training Phase of MDP for Query Expansion

1: Initialize experience memory M
2: Initialize action network with random weights θ
3: Initialize target network with weights $\theta_{target} \leftarrow \theta$
4: **for** episode from 1 to N **do**
5: **for** each concept c **do**
6: Obtain start state $z \leftarrow$*get_state*$([c, [\,]], E_1 ... E_n)$
7: **while** $true$ **do**
8: **if** $random() < \epsilon$ **then**
9: Select a random action a
10: **else**
11: Send state z to neural agent
12: Obtain action a from action network
13: **end if**
14: Obtain new state $z' \leftarrow T(z, a)$
15: Calculate reward, $r \leftarrow R(z, a, z')$
16: Store sample (z, a, z', r) to M
17: Update state $z \leftarrow z'$
18: Sample mini-batch (z_t, a_t, z'_t, r_t) from M
19: Calculate sample estimate using Equation 11
20: Perform a batch gradient descent step on the action network, updating parameters θ using Equation 12
21: Update $\theta_{target} \leftarrow \theta$ at every C steps.
22: **if** $a == action\ (4)$ **then**
23: **break**
24: **end if**
25: **end while**
26: Send the final $[c, [c_e]]$ to E and rank the stocks
27: **end for**
28: **end for**

new state z' and a reward r, which forms a sample, (z, a, z', r). The process repeats until action (4) is chosen.

6.3 Learning

We adopt Q-learning (Sutton and Barto, 1998) to optimize the neural agent, which uses a function $Q(z, a)$ to represent Q-values and the recursive Bellman equation to perform Q-value iteration, when observing a new sample (z, a, z', r). Since the state space Z can be extremely large in practice, we represent the Q-value function $Q(z, a)$ with a neural agent shown in Figure 2 (b) named the action network parametrized by θ (Mnih et al., 2015). The deep Q-learning method has the ability to capture nonlinear features and achieve better performance compared with traditional methods (Narasimhan et al., 2015). Formally,

$$Q(z, a) = Q(z, a; \theta) \tag{10}$$

To improve learning stability, sample reward estimates are obtained from a separate target network with the same architecture as the action network (Mnih et al., 2015), parametrized by θ_{target}. Formally, the sample reward estimate of (z, a, z', r) is:

$$y' = \begin{cases} r & if\ a = action(4) \\ r + \gamma \max_{a_{new} \in A} Q(z', a_{new}; \theta_{target}) & otherwise \end{cases} \tag{11}$$

Note that if the action (4) is taken, $y' = r$ since the process stops at state z and no further action will be taken so that the sum of rewards is r.

To learn the model parameters θ, the action network outputs $Q(z, a; \theta)$ should be close to sample estimates obtained from target network. Thus, we introduce an experience memory M to save history samples and select a mini-batch of samples according to a uniform distribution. We use the mean square error as the loss function:

$$E_{(z,a,z',r) \sim U(M)}[(Q(z, a; \theta) - y')^2] \tag{12}$$

The training phase is shown in Algorithm 1. In lines 8-12, we use ϵ-greedy exploration, which encourages the agent to explore unknown state space (Sutton and Barto, 1998).

7 Experiments

7.1 Datasets

We construct two datasets from the Chinese websites, Jinrongjie[2] and Tonghuashun[3], respectively, which are two mass medias for China stock markets. These two websites periodically publish their concept stock lists, which are manually collected and analyzed by their financial professionals. We observe high quote change correlations of the stocks of each concept c and their lists are commonly used by investors to select stocks, which confirms the credibility of these datasets. The Jinrongjie dataset consists of 206 concepts and each concept has an average of 25.4 manually suggested concept stocks. For the Tonghuashun dataset, there are 900 concepts and 15.6 manually suggested concept stocks on average.

There are two main stock exchanges in China, the Shanghai Stock Exchange[4] and the Shenzhen Stock Exchange[5]. We crawled stock lists from their official websites, with 3326 stocks in total.

We utilize four public data sources, S_1 to S_4, the statistics of which are shown in Table 1.

[2]金融界 http://stock.jrj.com.cn/concept/
[3]同花顺 http://stock.10jqka.com.cn/gngyw_list/
[4]上海证券交易所 http://www.sse.com.cn/assortment/stock/list/share/
[5]深圳证券交易所 http://www.szse.cn/

Source	# Docs	Avg # Words
News	255,318	2753
Report	12, 431	19,145
Wikipedia	2,143	3745
Search Engine	6,130	1846

Table 1: Data Source statistics

S_1: *News* is crawled from Sina Finance News[6], which originates from 2009 to 2017.

S_2: *Reports* consists of annual and quarterly company reports crawled from Sina Finance[7].

S_3: *Wikipedia* includes relevant wikipedia pages of the concepts and stocks, if any, which can provide some background knowledge.

S_4: *Search Engine* includes open-domain information for the concepts and stocks obtained using Bing API [8]. We adopt search engine results for representing heterogeneous web texts. The top-ranked webpages are crawled.

Given the Jinrongjie and Tonghuashun datasets, we randomly select 70%, 10% and 20% of the concepts as training, development and testing sets, respectively.

7.2 Baselines and Parameter Settings

We compare our method with four baselines:

Search is a naive information retrieval baseline, which sends the concept c and each stock o to an inverted index and obtains a list of top-k ranked documents ($k = 5$ in experiments) by a fixed ranking metric, Ocapi BM25 (Robertson et al., 2009). The stocks are ranked by the average of top-k documents' BM25 scores.

Rank is our ranking baseline. Five top-ranked documents from each source are fed into the model. All 3326 stocks are ranked for each concept.

Semantics ranks the stocks using Equation 4, which is the naive semantic relatedness $\hat{f}(c, o)$.

Semantics+ extends **Semantics** by including 8 most similar words to expand original concepts.

Semantics++ extends **Semantics** by including the most similar words with similarities larger than 0.65 to expand original concepts. On average, 6.3 concepts are included.

[6] http://finance.sina.com.cn/

[7] http://finance.sina.com.cn/focus/ssgsnb2016/

[8] https://azure.microsoft.com/en-us/services/cognitive-services/bing-web-search-api/

Jinrongjie				
Method	$P@5$	$P@10$	$R@30$	MAP
Search	0.402	0.315	0.338	0.296
Semantics	0.45	0.367	0.380	0.332
Semantics+	0.471	0.370	0.391	0.352
Semantics++	0.478	0.375	0.396	0.359
Rank	0.467	0.376	0.402	0.365
RL	**0.524**$*$	**0.427**$*$	**0.428**$*$	**0.398**$*$
Tonghuashun				
Method	$P@5$	$P@10$	$R@30$	MAP
Search	0.387	0.302	0.315	0.278
Semantics	0.437	0.347	0.360	0.327
Semantics+	0.448	0.356	0.374	0.345
Semantics++	0.453	0.362	0.380	0.351
Rank	0.458	0.373	0.381	0.356
RL	**0.507**$*$	**0.402**	**0.422**$*$	**0.378**$*$

Table 2: Concept stock recommendation results on Jinrongjie and Tonghuashun. $*$ denotes statistical significance using Wilcoxon signed rank test ($p < 0.05$)

For our model, denoted as **RL**, we set the window size as 80, the embedding size as 300 and the vocabulary size as $100,000$ in view of the large variety of phrases after word segmentation to train Doc2Vec. Also, the experience memory size is set to $50,000$ and older training samples are abandoned. The ϵ value is set as 1 at the start and gradually decreases to 0.1 after 3000 annealing steps. We perform a training phase after every 3 decision steps. The mini-batch size is set to 50. Dropout is applied to avoid overfitting and the dropout rate is 0.5. We set the learning rate for AdaGrad as 0.01. Gradient clipping (Pascanu et al., 2013) is adopted to prevent gradient exploding and vanishing during training process.

7.3 Recommendation Accuracies

We use four metrics, mean average precision (MAP), precision at 5 and 10 ($P@5$, $P@10$) and recall at 30 ($R@30$) to evaluate the algorithms. The results are shown in Table 2.

From Table 2, the first observation is that **RL** outperforms the baselines on both datasets, which demonstrates the effectiveness of combining semantic relatedness with query expansion based on reinforcement learning. The baseline **Rank** achieves the second best results. The large gap between **RL** and **Rank** indicates that **RL** is much easier to train compared to **Rank** on small data.

Second, we observe that **Semantics+** improves over **Semantics**, which shows that query expansion has the potentials to alleviate concept ambiguities and benefit concept stock recommendation. **Semantics++** can outperform **Semantics+** by considering semantic similarities. Also, compared to

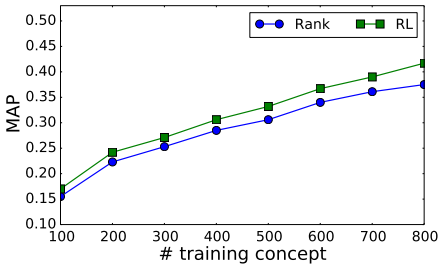


Figure 3: Learning Curve

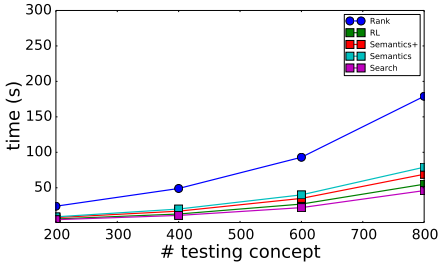


Figure 4: Efficiency

RL, we conclude that query expansion based on reinforcement learning could better utilize training data and significantly outperform naive query expansion methods.

The last observation is that **Search** performs the worst among the methods. This sheds light on the limitations of traditional search models and confirms the effectiveness of semantic modeling by word embedding and neural models.

7.4 Influence of Size of Training Data

We increase the amount of training concepts and study whether **RL** is easier to train than **Rank**. The results on Tonghuashun is shown in Figure 3 (similar patterns are demonstrated using Jinrongjie). With more training concepts, the MAPs of both methods increase. However, **RL** consistently outperforms **Rank** and the margin becomes larger. Thusly, we conclude that **RL** requires less data than **Rank** to achieve similar performance.

7.5 Efficiency Comparison

Figure 4 shows the efficiency of all algorithms on testing data. The three unsupervised algorithm **Search**, **Semantics** and **Semantics+** are more efficient compared to the supervised algorithm, **Rank** and **RL**. **RL** is more efficient compared to **Rank**, since **Rank** has to rank every stock to obtain concept stocks.

中字头 (Sino)		
Semantics+	*Rank*	*RL*
中远海控ZYHK	中国国航ZGGH	中国交建ZGJJ
中国交建ZGJJ	中国中铁ZGZT	中国中冶ZZZY
~~石油济柴SYJC~~	中国石油ZGSY	中国建筑ZGJZ
中国一重ZGYZ	~~中储股份ZCGF~~	中国中铁ZGZT
~~招商轮船ZSLC~~	中国中冶ZGZY	中国铁建ZGTJ
特斯拉 (Tesla)		
Semantics+	*Rank*	*RL*
~~万向钱潮WXQC~~	协鑫集成XXJC	~~万向钱潮WXQC~~
~~协鑫集成XXJC~~	~~比亚迪BYD~~	国机汽车GJQC
天汽模TQM	亚太股份YTGF	天汽模TQM
亚太股份YTGF	天汽模TQM	亚太股份YTGF
~~爱康科技AKKJ~~	~~长城汽车CCQC~~	~~上海临港SHLG~~
智能物流 (Intelligent Logistics)		
Semantics+	*Rank*	*RL*
东杰智能DJZN	~~华胜天成HSTC~~	飞力达FLD
~~亿阳信通YYXT~~	飞力达FLD	中储股份ZZGF
飞力达FLD	~~美菱电器MLDQ~~	东杰智能DJZN
~~美克家居MKJJ~~	东杰智能DJZN	圆通速递YTSD
~~天成自控TCZK~~	圆通速递YTSD	华鹏飞HPF

Table 3: Example concept stocks, where ~~stock~~ indicates a incorrectly recognized concept stock.

7.6 Case Study

Data Sources Effectiveness: To study the effectiveness of data sources, we count how many concepts are chosen from each data source during query expansion. For the Tonghuashun test data (similar tendencies are observed for Jinrongjie), 761, 689, 199, 344 concepts are selected for S_1-S_4, respectively. Accumulated rewards of these concepts for S_1-S_4 are 76.13, 61.32, 7.49 and 14.10, respectively. We conclude that *News* and *Reports* are relatively more effective for improving recommendation accuracies.

Recommended Stocks: To obtain a better understanding of our method, we examine the symbols of the top-5 selected stocks of concepts and some examples are shown in Table 3.

We notice that **RL** can effectively extend concepts with relevant concepts. For example, the algorithm extends 中字头 (Sino) with 中国 (China) and 国企 (State-owned enterprises), 特斯拉 (Tesla) with 入华 (into China), 电动车 (Electric cars) and 马斯克 (Musk) and 智能物流 (Intelligent Logistics) with 物流 (Logistics), CSN (China Smart Logistic Network), 仓储 (Warehousing) and 配送 (Delivery), which results in more accurate concept stocks.

For 特斯拉 (Tesla), **RL** made two mistakes due to rumor and ambiguity. For example, 上海临港SHLG is chosen because of rumors that Tesla will establish a new factory there. 万向钱潮WXQC is mistakenly chosen because 万向钱潮WXQC is called China's Tesla in some news due to its investments in electric cars. In contract, **Semantics+** and **Rank** are limited by lack of su-

pervision and highly unbalanced datasets, respectively. For example, **Rank** mistakenly chooses 美菱电器MLDQ in that it confuses 智能家电 (Smart Appliances) with 智能物流 (Intelligent Logistics). We conclude that **RL** is capable of expanding concepts with relevant concepts that helps find more revelant stocks.

Conclusion

We have investigated a reinforcement learning method to automatically mine evidences from large-scale text data for measuring the correlation between a concept and a list of stocks. Compared to standard information retrieval methods, our method leverages a small amount of training data for obtaining a flexible strategy of query expansion, thus being able to disambiguate contexts in exploration. Results on two Chinese datasets show that our method is highly competitive for our task, thus providing a tool for investors to gain understandings of emerging markets.

Acknowledgments

We thank the anonymous reviewers for their insightful comments. We would like to thank Yumin Zhou for her insightful discussion and assisting coding. Yue Zhang is the corresponding author.

References

Kumaripaba Athukorala, Alan Medlar, Antti Oulasvirta, Giulio Jacucci, and Dorota Glowacka. 2016. Beyond relevance: Adapting exploration/exploitation in information retrieval. In *Proceedings of the 21st International Conference on Intelligent User Interfaces*. ACM, pages 359–369.

Jing Bai, Dawei Song, Peter Bruza, Jian-Yun Nie, and Guihong Cao. 2005. Query expansion using term relationships in language models for information retrieval. In *CIKM*. ACM, pages 688–695.

Y-Lan Boureau, Jean Ponce, and Yann LeCun. 2010. A theoretical analysis of feature pooling in visual recognition. In *ICML*. pages 111–118.

Guihong Cao, Jian-Yun Nie, Jianfeng Gao, and Stephen Robertson. 2008. Selecting good expansion terms for pseudo-relevance feedback. In *SIGIR*. ACM, pages 243–250.

Ziqiang Cao, Furu Wei, Li Dong, Sujian Li, and Ming Zhou. 2015. Ranking with recursive neural networks and its application to multi-document summarization. In *AAAI*. pages 2153–2159.

Claudio Carpineto and Giovanni Romano. 2012. A survey of automatic query expansion in information retrieval. *ACM Computing Surveys* 44(1):1.

D Manning Christopher, Raghavan Prabhakar, and SCHÜTZE Hinrich. 2008. Introduction to information retrieval. *An Introduction To Information Retrieval* 151:177.

Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*. ACM, pages 253–262.

Fernando Diaz, Bhaskar Mitra, and Nick Craswell. 2016. Query expansion with locally-trained word embeddings. *arXiv preprint arXiv:1605.07891* .

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

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks* pages 602–610.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. MIT Press, pages 1735–1780.

Saar Kuzi, Anna Shtok, and Oren Kurland. 2016. Query expansion using word embeddings. In *CIKM*. ACM, pages 1929–1932.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*. pages 1188–1196.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*. pages 3111–3119.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. 2015. Human-level control through deep reinforcement learning. *Nature* pages 529–533.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. *arXiv preprint arXiv:1506.08941* .

Rodrigo Nogueira and Kyunghyun Cho. 2017. Task-oriented query reformulation with reinforcement learning. *arXiv preprint arXiv:1704.04572* .

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. *ICML* 28:1310–1318.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pages 1532–1543.

Carolyn C Preston and Andrew M Colman. 2000. Optimal number of response categories in rating scales: reliability, validity, discriminating power, and respondent preferences. *Acta psychologica* pages 1–15.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval* pages 333–389.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014a. A latent semantic model with convolutional-pooling structure for information retrieval. In *CIKM*.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014b. Learning semantic representations using convolutional neural networks for web search. In *WWW*. ACM, pages 373–374.

Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*. MIT press Cambridge.

Gang Wu and Edward Y Chang. 2003. Class-boundary alignment for imbalanced dataset learning. In *ICML 2003 workshop on learning from imbalanced data sets II, Washington, DC*. pages 49–56.

Jie Yang, Yue Zhang, and Fei Dong. 2017. Neural word segmentation with rich pretraining. In *ACL*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103* .

Binarized LSTM Language Model

Xuan Liu*, **Di Cao***, **Kai Yu**
Key Laboratory of Shanghai Education Commission for
Intelligent Interaction and Cognitive Engineering,
SpeechLab, Department of Computer Science and Engineering,
Brain Science and Technology Research Center,
Shanghai Jiao Tong University, Shanghai, China
{liuxuan0526, caodi0207, ky219.cam}@gmail.com

Abstract

The long short-term memory (LSTM) language model (LM) has been widely investigated for automatic speech recognition (ASR) and natural language processing (NLP). Although excellent performance is obtained for large vocabulary tasks, tremendous memory consumption prohibits the use of LSTM LMs in low-resource devices. The memory consumption mainly comes from the word embedding layer. In this paper, a novel binarized LSTM LM is proposed to address the problem. Words are encoded into binary vectors and other LSTM parameters are further binarized to achieve high memory compression. This is the first effort to investigate binary LSTMs for large vocabulary language modeling. Experiments on both English and Chinese LM and ASR tasks showed that binarization can achieve a compression ratio of 11.3 without any loss of LM and ASR performance and a compression ratio of 31.6 with acceptable minor performance degradation.

1 Introduction

Language models (LMs) play an important role in natural language processing (NLP) tasks. N-gram language models used to be the most popular language models. Considering the previous N-1 words, N-gram language models predict the next word. However, this leads to the loss of long-term dependencies. The sample space size increases exponentially as N grows, which induces data sparseness (Cao and Yu, 2017).

Neural network (NN) based models were first introduced into language modeling in 2003 (Bengio et al., 2003). Given contexts with a fixed size, the model can calculate the probability distribution of the next word. However, the problem of long-term dependencies still remained, because the context window is fixed. Currently, recurrent neural network (RNN) based models are widely used on natural language processing (NLP) tasks for excellent performance (Mikolov et al., 2010). Recurrent structures in neural networks can solve the problem of long-term dependencies to a great extent. Some gate based structures, such as long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and gated recurrent unit (GRU) (Chung et al., 2014) improve the recurrent structures and achieve state-of-the-art performance on most NLP tasks.

However, neural network models occupy tremendous memory space so that it is almost impossible to put the models into low-resource devices. In practice, the vocabulary is usually very large. So the memory consumption mainly comes from the embedding layers. And, the word embedding parameters are floating point values, which adds to the memory consumption.

The first contribution in this paper is that a novel language model, the binarized embedding language model (BELM) is proposed to reduce the memory consumption. Words are represented in the form of binarized vectors. Thus, the consumption of memory space is significantly reduced. Another contribution in the paper is that we binarize the LSTM language model combined with the binarized embeddings to further compress the parameter space. All the parameters in the LSTM language model are binarized.

Experiments are conducted in language modeling and automatic speech recognition (ASR) rescoring tasks. Our model performs well without any loss of performance at a compression ratio of 11.3 and still has acceptable results with only a minor loss of performance even at a compression ratio of 31.6. Investigations are also made to evaluate whether the binarized embeddings lose information. Experiments are conducted on word

*Both authors contributed equally to this work.

Proceedings of NAACL-HLT 2018, pages 2113–2121
New Orleans, Louisiana, June 1 - 6, 2018.

similarity tasks. The results show the binarized embeddings generated by our models still perform well on the two datasets.

The rest of the paper is organized as follows, section 2 is the related work. Section 3 explains the proposed language model and section 4 shows the experimental setup and results. Finally, conclusions will be given in section 5 and we describe future work in section 6.

2 Related Work

Nowadays, with the development of deep learning, neural networks have yielded good results in many areas. However, neural networks may require tremendous memory space, making it difficult to run such models on low-resource devices. Thus, it is necessary to compress neural networks.

In recent years, many methods of compressing neural networks have been proposed. Pruning (Han et al., 2015) reduces the number of parameters of the neural network by removing all connections with the weights below a threshold. Quantization (Han et al., 2015) clusters weights to several clusters. A few bits are used to represent the neurons and to index a few float values.

Binarization is also a method to compress neural networks. BNNs(Courbariaux et al., 2016) are binarized deep neural networks. The weights and activations are constrained to 1 or -1. BNNs can drastically reduce memory size and replace most arithmetic operations with bit-wise operations. Different from pruning and quantization, binarization does not necessarily require pre-training and can achieve a great compression ratio. Many binarization methods have been proposed (Courbariaux et al., 2015, 2016; Rastegari et al., 2016; Xiang et al., 2017). However, only a few (Hou et al., 2016; Edel and Köppe, 2016) are related to recurrent neural network. (Hou et al., 2016) implements a character level binarized language model with a vocabulary size of 87. However, they did not do a comprehensive study on binarized large vocabulary LSTM language models.

3 Binarized Language Model

3.1 LSTM Language Model

The RNN language model is proposed to deal with sequential data. Due to the vanishing and exploding gradients problem, it is difficult for a RNN language model to learn long-term dependencies. The LSTM, which strengthens the recurrent neural model with a gating mechanism, tackles this problem and is widely used in natural language processing tasks.

The goal of a language model is to compute the probability of a sentence $(x_1, \ldots, x_N)$. A typical method is to decompose this probability word by word.

$$P(x_1, ..., x_N) = \prod_{t=1}^{N} P(x_t | x_1, ..., x_{t-1}) \quad (1)$$

(Hochreiter and Schmidhuber, 1997) proposed a Long Short-Term Memory Network, which can be used for sequence processing tasks. Consider an one-layer LSTM network, where N is the length of the sentence, and x_t is the input at the t-th moment. y_t is the output at the t-th moment, which is equal to x_{t+1} in a language model. Denote $\boldsymbol{h_t}$ and $\boldsymbol{c_t}$ as the hidden vector and the cell vector at the t-th moment, which is used for representing the history of $(x_1, ..., x_{t-1})$. $\boldsymbol{h_0}$ and $\boldsymbol{c_0}$ are initialized with zero. Given x_t, $\boldsymbol{h_{t-1}}$ and $\boldsymbol{c_{t-1}}$, the model calculates the probability of outputting y_t.

The first step of an LSTM language model is to extract the representation $\boldsymbol{e_t}$ of the input x_t from the embeddings $\mathbf{W_e}$. Since x_t is a one-hot vector, this operation can be implemented by indexing rather than multiplication.

$$\boldsymbol{e_t} = \mathbf{W_e} \boldsymbol{x_t} \quad (2)$$

After that, $\boldsymbol{e_t}$, along with $\boldsymbol{h_{t-1}}$ and $\boldsymbol{c_{t-1}}$ are fed into the LSTM cell. The hidden vector $\boldsymbol{h_t}$ and the cell vector $\boldsymbol{c_t}$ can be computed according to:

$$\begin{aligned}
\boldsymbol{f_t} &= \text{sigmoid}\left(\mathbf{W_f}\left\{\boldsymbol{h_{t-1}}, \boldsymbol{e_t}\right\} + \boldsymbol{b_f}\right) \\
\boldsymbol{i_t} &= \text{sigmoid}\left(\mathbf{W_i}\left\{\boldsymbol{h_{t-1}}, \boldsymbol{e_t}\right\} + \boldsymbol{b_i}\right) \\
\boldsymbol{o_t} &= \text{sigmoid}\left(\mathbf{W_o}\left\{\boldsymbol{h_{t-1}}, \boldsymbol{e_t}\right\} + \boldsymbol{b_o}\right) \\
\hat{\boldsymbol{c}}_{\boldsymbol{t}} &= \tanh\left(\mathbf{W_{\hat{c}}}\left\{\boldsymbol{h_{t-1}}, \boldsymbol{e_t}\right\} + \boldsymbol{b_{\hat{c}}}\right) \\
\boldsymbol{c_t} &= \boldsymbol{f_t} \cdot \boldsymbol{c_{t-1}} + \boldsymbol{i_t} \cdot \hat{\boldsymbol{c}}_{\boldsymbol{t}} \\
\boldsymbol{h_t} &= \boldsymbol{o_t} \cdot \tanh\left(\boldsymbol{c_t}\right)
\end{aligned} \quad (3)$$

The word probability distribution at the t-th moment can be calculated by:

$$P(y_t | x_1, ..., x_t) = \boldsymbol{p_t} = \text{softmax}(\mathbf{W_y} \boldsymbol{h_t}) \quad (4)$$

The probability of taking y_t as the output at the t-th moment is:

$$p_{y_t} = \boldsymbol{p_t} \times \boldsymbol{y_t} \quad (5)$$

3.2 Binarized Embedding Language Model

The binarized embedding language model (BELM) is a novel LSTM language model with binarized input embeddings and output embeddings. For a one-layer LSTM language model with a vocabulary size of V, embedding and hidden layer size of H. The size in bytes of the input embeddings, the output embeddings, and the LSTM cells are $4VH$, $4VH$ and $32H^2 + 16H$. When V is much larger than H, which is often the case for language models, the parameters of the input embeddings and the output embeddings occupy most of the space. If the embeddings of the input layer and the output layer are binarized, the input layer and the output layer will only take $1/32$ of the original memory consumption, which can greatly reduce the memory consumption of running neural language model.

It is important to find good binary embeddings. Directly binarizing well-trained word embeddings cannot yield good binarized representations. Instead, we train good binary embeddings from scratch. The training approach is similar to the methods proposed in (Courbariaux et al., 2016; Rastegari et al., 2016). At run-time, the input embedding and the output embedding are binarized matrices. However, at train-time, float versions of the embeddings, which are used for calculating the binarized version of embeddings, are still maintained. In the propagation step, a deterministic function sign is used to binarize the float versions of the embeddings. In the back-propagation step, the float versions of the embeddings are updated according to the gradient of the binarized embedding.

$$w^b = \text{sign}\,(w) = \begin{cases} +1 & if\ w > 0, \\ -1 & otherwise. \end{cases} \tag{6}$$

The derivative of the sign function is zero almost everywhere, and it is impossible to back-propagate through this function. As introduced in (Hubara et al., 2016), a straight-through estimator is used to get the gradient. Assume the gradient of the binarized weight $\frac{\partial C}{\partial \mathbf{W^b}}$ has been obtained, the gradient of the float version of the weight is:

$$\frac{\partial C}{\partial \mathbf{W}} = \frac{\partial C}{\partial \mathbf{W^b}} \tag{7}$$

A typical weight initialization method initializes each neuron's weights randomly from the Gaussian distribution $N(0, \sqrt{1/H})$. This initialization approach can maximize the gradients and mitigate the vanishing gradients problem. From this perspective, 1 or -1 is too large. So, in practice, we binarize the embeddings to a smaller scale. Although the weight is binarized to a floating point number, the matrix can also be saved one bit per neuron, as long as the fixed float value is memorized separately.

$$\text{binarize}\,(w) = \begin{cases} +\sqrt{1/H} & if\ w > 0, \\ -\sqrt{1/H} & otherwise. \end{cases} \tag{8}$$

Since directly binarizing the input embeddings $\mathbf{W_e}$ and the output embeddings $\mathbf{W_y}$ will limit the scale of the embeddings, additional linear layers (without activation) are added behind the input embedding layer and in front of the output embedding layer to enhance the model. Denote $\mathbf{W_e^b}$ and $\mathbf{W_y^b}$ as the binarized weights corresponding to $\mathbf{W_e}$ and $\mathbf{W_y}$. Denote $\mathbf{W_{T_e}}$ and $\boldsymbol{b_{T_e}}$, $\mathbf{W_{T_y}}$ and $\boldsymbol{b_{T_y}}$ as the weights and the biases of the first and the second linear layer. The input of the LSTM $\boldsymbol{e_t}$ and the word probability $\boldsymbol{p_t}$ of the binarized embedding language model are calculated according to:

$$\begin{aligned} \boldsymbol{e_t} &= \mathbf{W_{T_e}}\left(\mathbf{W_e^b}\boldsymbol{x_t}\right) + \boldsymbol{b_{T_e}} \\ \boldsymbol{p_t} &= \text{softmax}\left(\mathbf{W_y^b}\left(\mathbf{W_{T_y}}\boldsymbol{h_t} + \boldsymbol{b_{T_y}}\right)\right) \end{aligned} \tag{9}$$

The additional linear layer before the output embedding layer is very important for the binarized embedding language model, especially for low dimensional models. Removing this layer will result in an obvious decrease in performance.

3.3 Binarized LSTM Language Model

Subsection 3.2 explains how to binarize the embedding layer, but the LSTM network can also be binarized. In a binarized LSTM language model, all the matrices in the parameters are binarized, which can save much more memory space. Implementing the binarized linear layer is important for designing a binarized LSTM language model (BLLM). In a binarized linear layer, there are three parameters, $\mathbf{W}$, $\boldsymbol{\gamma}$ and $\boldsymbol{b}$. $\mathbf{W}$ is a matrix, $\boldsymbol{\gamma}$ and $\boldsymbol{b}$ are vectors. The matrix $\mathbf{W}$, which takes up most of the space in a linear layer, is binarized. $\boldsymbol{\gamma}$ and $\boldsymbol{b}$ remain floating point values. b is the bias of the linear layer, and $\boldsymbol{\gamma}$ is introduced to fix the scale problem of the binary matrix.

The forward- and back-propagation algorithms are shown in Algorithm 1 and Algorithm 2. The structure of this linear layer is very similar to the structure of batch normalization (Ioffe and Szegedy, 2015), except the output of each dimension over the mini-batches is not normalized. Batch normalization is hard to apply to a recurrent neural network, due to the dependency over entire sequences. However, the structure of the batch normalization is quite useful. Since binarizing W would fix the scale of the weight, additional freedom is needed to overcome this issue. The shift operation can rescale the output to a reasonable range.

Algorithm 1 The propagation of linear layer

Input: input $\boldsymbol{x}$, weights $\mathbf{W}$, $\boldsymbol{\gamma}$ and $\boldsymbol{b}$

Output: output $\boldsymbol{y}$

1: $\mathbf{W^b} = \text{binarize}\left(\mathbf{W}\right)$
2: $\boldsymbol{s} = \mathbf{W^b}\boldsymbol{x}$
3: $\boldsymbol{y} = \boldsymbol{s} \cdot \exp\left(\boldsymbol{\gamma}\right) + \boldsymbol{b}$

Algorithm 2 The back-propagation of linear layer

Input: input $\boldsymbol{x}$, weights $\mathbf{W}$, $\boldsymbol{\gamma}$ and $\boldsymbol{b}$, binarized weight $\mathbf{W^b}$, temporary value $\boldsymbol{s}$ (calculated in the propagation period), the gradient of the output $\frac{\partial C}{\partial \boldsymbol{y}}$, learning rate η, binary weight range α

Output: the gradient of the input $\frac{\partial C}{\partial \boldsymbol{x}}$, the gradient of the weight $\frac{\partial C}{\partial \mathbf{W}}$, $\frac{\partial C}{\partial \boldsymbol{\gamma}}$, $\frac{\partial C}{\partial \boldsymbol{b}}$, update the weights

1: $\frac{\partial C}{\partial \boldsymbol{b}} = \frac{\partial C}{\partial \boldsymbol{y}}$
2: $\frac{\partial C}{\partial \boldsymbol{\gamma}} = \frac{\partial C}{\partial \boldsymbol{y}} \cdot \boldsymbol{s} \cdot \exp\left(\boldsymbol{\gamma}\right)$
3: $\frac{\partial C}{\partial \boldsymbol{s}} = \frac{\partial C}{\partial \boldsymbol{y}} \cdot \exp\left(\boldsymbol{\gamma}\right)$, $\frac{\partial C}{\partial \mathbf{W^b}} = \frac{\partial C}{\partial \boldsymbol{s}}\boldsymbol{x}$, $\frac{\partial C}{\partial \mathbf{W}} = \frac{\partial C}{\partial \mathbf{W^b}}$
4: $\frac{\partial C}{\partial \boldsymbol{x}} = \frac{\partial C}{\partial \boldsymbol{s}}\mathbf{W^b}$
5: update $\mathbf{W}$, $\boldsymbol{\gamma}$, $\boldsymbol{b}$ according to $\frac{\partial C}{\partial \mathbf{W}}$, $\frac{\partial C}{\partial \boldsymbol{\gamma}}$, $\frac{\partial C}{\partial \boldsymbol{b}}$ with learning rate η.
6: $\text{clamp}(\mathbf{W}, -\alpha, \alpha)$
7: **return** $\frac{\partial C}{\partial x}$

The structure of the input embeddings and the output embeddings of the binarized LSTM language model is similar to the binarized embedding language model. The embeddings are binarized and additional linear layers are added after the input embedding layer and in front of the output embedding layer. However, the additional linear layers are also binarized according to Algorithm 1 and Algorithm 2.

3.4 Memory Reduction

Denote the size of the vocabulary as V, and the size of the embedding and hidden layer as H. The memory consumptions of a one-layer LSTM language model, BELM and BLLM are listed in Table 1.

Model	Memory (bytes)
LSTM	$8VH + 32H^2 + 16H$
BELM	$0.25VH + 40H^2 + 24H$
BLLM	$0.25VH + 1.25H^2 + 48H$

Table 1: Memory Requirements

For a language model, the vocabulary size is usually much larger than the hidden layer size. The main memory consumption comes from the embedding layers, which require $8VH$ bytes for an LSTM language model. Binarized embeddings can reduce this term to $0.25VH$ bytes. Further compression of the LSTM can drop the coefficient of H^2 from 32 to 1.25.

4 Experiments

4.1 Experimental Setup

Our model is evaluated on the English Penn TreeBank (PTB) (Marcus et al., 1993), Chinese short message (SMS) and SWB-Fisher (SWB). The Penn TreeBank corpus is a famous English dataset, with a vocabulary size of 10K and 4.8% words out of vocabulary (OOV), which is widely used to evaluate the performance of a language model. The training set contains approximately 42K sentences with 887K words. The Chinese SMS corpus is collected from short messages. The corpus has a vocabulary size of about 40K. The training set contains 380K sentences with 1931K words. The SWB-Fisher corpus is an English corpus containing approximately 2.5M sentences with 24.9M words. The corpus has a vocabulary size of about 30K. hub5e is the dataset for the SWB ASR task.

We also evaluate the word embeddings produced by our models on two word similarity datasets. The models are trained on the Text8 corpus to extract the word embeddings. The Text8 corpus is published by Google and collected from Wikipedia. Text8 contains about 17M words with a vocabulary size of about 47k. The WordSimilarity-353(WS-353) Test Collection contains two sets of English word pairs along with

human-assigned similarity judgments. The collection can be used to train and test computer algorithms implementing semantic similarity measures. A combined set (combined) is provided that contains a list of all 353 words, along with their mean similarity scores. (Finkelstein et al., 2001) The MEN dataset consists of 3,000 word pairs, randomly selected from words that occur at least 700 times in the freely available ukWaC and Wackypedia corpora combined (size: 1.9B and 820M tokens, respectively) and at least 50 times (as tags) in the open-sourced subset of the ESP game dataset. In order to avoid picking unrelated pairs only, the pairs are sampled so that they represent a balanced range of relatedness levels according to a text-based semantic score (Bruni et al., 2014).

First, we conduct experiments on the PTB, SWB and Text8 corpora respectively to evaluate language modeling performance. We use perplexity (PPL) as the metric to evaluate models of different sizes. Then, the models are evaluated on ASR rescoring tasks. Rescoring the 100-best sentences generated by the weighted finite state transducer (WFST), the model is evaluated by word error rate (WER). Finally, we conduct experiments on word similarity tasks to evaluate whether the word embeddings produced by our models lose any information.

4.2 Experiments in Language Modeling

For traditional RNN based language models, the memory consumption mainly comes from the embedding layers (both input and output layers). However, when the hidden layer size grows, the memory consumption of the RNN module also becomes larger. So the total memory usage relates to both the vocabulary size and hidden layer size, as mentioned in section 3.4.

Experiments are conducted in language modeling to evaluate the model on the PTB, SWB, and SMS corpora respectively. In language modeling tasks, we regularize the networks using dropout(Zaremba et al., 2014). We use stochastic gradient descent (SGD) for optimization. The batch size is set to 64. For the PTB corpus, the dropout rate is tuned for different training settings. For the SWB corpus, we do not use dropout technique. For the SMS corpus, the dropout rate is set to 0.25. We train models of different sizes on the three corpora and record the memory usage of the trained models. The initial learning rate is set to 1.0 for all settings. Since PTB is a relatively small dataset and the convergence rates of the BELM and the BLLM are slower than LSTM language model, we reduce the learning rate by half every three epochs if the perplexity on the validation set is not reduced. For the other experiments, the learning rate is always reduced by half every epoch if the perplexity on the validation set is not reduced. As introduced in section 3, the bias of the output embedding layer is omitted. Adding bias term in the output embedding layer leads to small performance degradation in the BELM and the BLLM model, although it leads to a small improvement in the LSTM model. This phenomenon may be related to optimization problems.

	Hidden size	**LSTM**	**BELM**	**BLLM**
Memory	500	48.0M	11.3M	1.6M
PPL		91.8	**88.0**	95.2
Memory	1000	112.0M	42.5M	3.8M
PPL		89.4	**85.7**	94.9

Table 2: Performances on the English PTB corpus

	Hidden size	**LSTM**	**BELM**	**BLLM**
Memory	500	129.1M	13.8M	4.1M
PPL		**57.6**	58.4	60.4
Memory	1000	274.2M	47.6M	8.9M
PPL		56.1	**55.6**	56.2

Table 3: Performance on the English SWB corpus

	Hidden size	**LSTM**	**BELM**	**BLLM**
Memory	500	170.8M	15.1M	5.4M
PPL		90.0	**89.8**	96.8
Memory	1000	357.6M	50.2M	11.5M
PPL		89.5	**87.8**	94.3

Table 4: Performance on the Chinese SMS corpus

Because the total memory usage relates to both the vocabulary size and hidden layer size, the memory reduction on various corpora is quite different. For our BELM model, the floating point embedding parameters are replaced by single bits, which could significantly reduce the memory usage. On the PTB corpus, the BELM models even

outperform the baseline LSTM LM. The small model (500 LSTM units) has a relative PPL improvement of 4.1% and achieves a compression ratio of 4.3 and the large model (1000 LSTM units) also has a relative PPL improvement of 4.1% and achieves a compression ratio of 2.6. On the SWB corpus, the BELM models still perform well compared with the baseline model and achieve compression ratios of 9.4 and 5.8 respectively for the small and large models. On the SMS corpus, the BELMs model also gains relative PPL improvements of 0.2% and 1.9%, and achieves compression ratios of 11.3 and 7.1 respectively. In summary, the BELM model performs as well as the baseline model both on English and Chinese corpora, and reduces the memory consumption to a large extent.

The BLLM model, however, does not outperform the baseline model, but still has acceptable results with a minor loss of performance. Since both the LSTM model and the embeddings are binarized, the total compression ratio is quite significant. The average compression ratio is about 32.0, so the memory consumption of the language model is significantly reduced.

We also study the performance of pruning the LSTM language model. We prune each parameter matrix and the embedding layers with various pruning rates respectively, and fine-tune the model with various dropout rates. In our experiments, pruning 75% parameter nodes hardly affects the performance. However, if we try pruning more parameter nodes, the perplexity increases rapidly. For example, for the English PTB dataset, when we prune 95% parameter nodes of the embedding layers of an LSTM language model (500 LSTM units), the perpexity will increase from 91.8 to 112.3. When we prune 95% parameter nodes of an LSTM language model (500 LSTM units), the perplexity will increase from 91.8 to 132.3. Therefore, the effect of pruning is not as good as binarization for the language modeling task.

Binarization can be considered as a special case of quantization, which quantizes the parameters to pairs of opposite numbers. So, compared to normal quantization, binarization can achieve a better compression ratio. In addition, for binarization, we do not need to determine the position of each unique values in advance. Therefore, binarization is more flexible than quantization.

We then study the effect of extra binary linear layers in the BLLM. The additional binary linear layer after the input embedding layer and the additional binary linear layer in front of the output embedding layer are removed respectively in this experiment. We use well-trained embeddings to initialize the corresponding embedding layers and do the binarization using the method proposed in (Rastegari et al., 2016) when the additional binary linear layer is removed. The perplexities are listed in Table 5. No-i means no additional binary linear layer after the input embedding layer. No-o means no additional binary linear layer in front of the output embedding layer. No-io means no additional binary linear layers. The experiment is conducted on the PTB corpus.

	Hidden size	**BLLM**	**BLLM no-i**	**BLLM no-o**	**BLLM no-io**
PPL	500	95.2	95.2	101.7	100.3
PPL	1000	94.9	94.5	96.7	96.3

Table 5: Performances on the English PTB corpus

If the additional binary linear layer after the input embedding layer is removed, the performance does not drop, and even becomes better when the hidden layer size is 1000. Although the additional binary layer after the input embedding layer is removed, the float version of the input embeddings of BLLM no-i is initialized with well-trained embeddings, while the BLLM is not initialized with the well-trained embeddings. We think initialization is the reason why the BLLM no-i performs comparatively to the BLLM. We also observe a PPL increase of 1-2 points for BLLM no-i if the input embeddings are not pre-trained (not listed in the table). This phenomenon prompts us to pre-train embeddings, which we leave to future work. Once the additional binary linear layer in front of the output embedding layer is removed, the performance degradation is serious. This shows that the output embeddings of the language model should not be directly binarized; the additional binary linear layer should be inserted to enhance the model's capacity, especially for low dimensional models.

4.3 Experiments on ASR Rescoring Tasks

Experiments are conducted on the ASR rescoring task to evaluate the model on the hub5e and SMS corpora. Hub5e is a test dataset of the SWB corpus which we use for ASR rescoring tasks. For the hub5e dateset, A VDCNN (Qian et al., 2016)

(very deep CNN) model on the 300-hour task is applied as the acoustic model. For the Chinese SMS dataset, the acoustic model is a CD-DNN-HMM model. The weighted finite state transducer (WFST) is produced with a 4-gram language model. Then our language models are utilized to rescore the 100-best candidates. The models are evaluated by the metric of word error rate (WER).

Model	Hidden size	hub5e	SMS
LSTM	500	8.7	10.5
BELM		**8.5**	**10.3**
BLLM		8.7	10.8
LSTM	1000	8.5	10.4
BELM		8.5	**10.2**
BLLM		**8.4**	10.3

Table 6: Performances on ASR rescoring tasks

Table 6 shows the results on ASR rescoring tasks. The BELM model and BLLM model perform well both on the English and Chinese datasets. The BELM model achieves an absolute 0.2% WER improvement compared with the baseline model in three of the experiments. The BLLM model also has good results, even though it performs not so well in language modeling. The results show that our language models work well on ASR rescoring tasks even with much less memory consumption.

4.4 Investigation of Binarized Embeddings

The experiments above show the good performances of our models. We also want to investigate whether the binarized embeddings lose any information. So, the embeddings are evaluated on two word similarity tasks. Experiments are conducted on the WS-353 and MEN tasks. We have trained the baseline LSTM model, the BELM model and BLLM model of a medium size on the Text8 corpus. We binarize the embeddings of the trained baseline LSTM model to investigate whether there is any loss of information by the simple binarization method (labeled LSTM-bin in the table below). For each dimension, we calculate the mean and set the value to 1 if it is bigger than the mean, otherwise, we set it to -1.

The embedding size and the hidden layer size are set to 500. We use stochastic gradient descent (SGD) to optimize our models. We use cosine distance to evaluate the similarity of the word pairs. Spearman's rank correlation coefficient is calculated to evaluate the correlation between the two scores given by our models and domain experts.

Model	PPL
LSTM	166.0
BELM	164.7
BLLM	172.3

Table 7: Language modeling performance on the Text8 corpus

Model	WS-353	MEN
LSTM	53.1	46.3
LSTM-bin	25.5	19.4
BELM	49.1	47.0
BLLM	**56.0**	**52.2**

Table 8: Performances on the word similarity tasks

Table 7 shows our models perform well in language modeling on the Text8 corpus. Table 8 summarizes the performance of the word embeddings in the similarity tasks. The embeddings generated by the simple binarization method perform obviously worse than the other embeddings, which indicates that much information is lost. The BELM model outperforms the baseline model on the MEN task, although it doesnt perform as well as the baseline model on the WS-353 task. However, the MEN dataset contains many more word pairs, which makes the results on this dataset more convincing. The BLLM model significantly outperforms the baseline model on the two tasks. The results indicate that the binarized embeddings of the BLLM do not lose any semantic information although the parameters are represented only by -1 and 1.

We suspect that binarization plays a role in regularization and produces more robust vectors. We also give an example visualization of some word vectors. The dimension of the embeddings of the BLLM is reduced by TSNE (Maaten and Hinton, 2008). The words which are the closest to **father** (according to the cosine distance of word vectors) are shown in Figure 1.

In this figure, **mother** and **parents** are the closest words to **father**, which is quite understandable. The words **husband**, **wife**, **grandfather** and **grandmother** also gather together and most words in the figure are related to **father**, indicat-

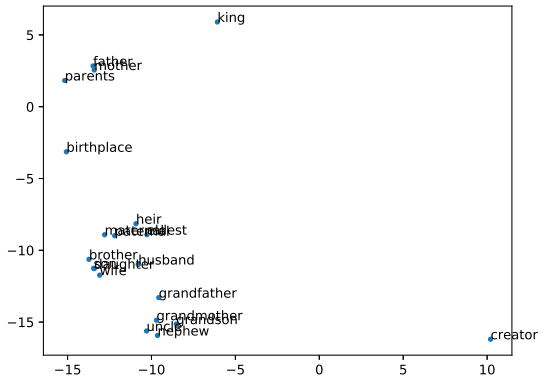


Figure 1: Visualization of the Binarized Embeddings

ing the embeddings indeed carry semantic information.

5 Conclusion

In this paper, a novel language model, the binarized embedding language model (BELM) is proposed to solve the problem that NN based language models occupy tremendous space. For traditional RNN based language models, the memory consumption mainly comes from the embedding layers (both input and output layers). However, when the hidden layer size grows, the memory consumption of the RNN module also becomes larger. So, the total memory usage relates to both the vocabulary size and hidden layer size. In the BELM model, words are represented in the form of binarized vectors, which only contain parameters of -1 or 1. For further compression, we binarize the long short-term memory language model combined with the binarized embeddings. Thus, the total memory usage can be significantly reduced. Experiments are conducted on language modeling and ASR rescoring tasks on various corpora. The results show that the BELM model performs well without any loss of performances at compression ratios of 2.6 to 11.3, depending on the hidden and vocabulary size. The BLLM model compresses the model parameters almost thirty-two times with a slight loss of performance. We also evaluate the embeddings on word similarity tasks. The results show the binarized embeddings even perform much better than the baseline embeddings.

6 Future Work

In the future, we will study how to improve the performance of the BLLM model. And, we will research methods to accelerate the training and reduce the memory consumption during training.

Acknowledgments

The corresponding author is Kai Yu. This work has been supported by the National Key Research and Development Program of China under Grant No.2017YFB1002102, and the China NSFC projects (No. 61573241). Experiments have been carried out on the PI supercomputer at Shanghai Jiao Tong University.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research* 3(Feb):1137–1155.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *J. Artif. Intell. Res.(JAIR)* 49(2014):1–47.

Di Cao and Kai Yu. 2017. Deep attentive structured language model based on lstm. In *International Conference on Intelligent Science and Big Data Engineering*. Springer, pages 169–180.

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

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. 2015. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems*. pages 3123–3131.

Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830* .

Marcus Edel and Enrico Köppe. 2016. Binarized-blstm-rnn based human activity recognition. In *Indoor Positioning and Indoor Navigation (IPIN), 2016 International Conference on*. IEEE, pages 1–7.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*. ACM, pages 406–414.

Song Han, Huizi Mao, and William J Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149* .

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

Lu Hou, Quanming Yao, and James T Kwok. 2016. Loss-aware binarization of deep networks. *arXiv preprint arXiv:1611.01600* .

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized neural networks. In *Advances in neural information processing systems*. pages 4107–4115.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*. pages 448–456.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research* 9(Nov):2579–2605.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. *Building a large annotated corpus of English: the penn treebank*. MIT Press.

Tomas Mikolov, Martin Karafit, Lukas Burget, Jan Cernock, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September*. pages 1045–1048.

Yanmin Qian, Mengxiao Bi, Tian Tan, Kai Yu, Yanmin Qian, Mengxiao Bi, Tian Tan, and Kai Yu. 2016. Very deep convolutional neural networks for noise robust speech recognition. *IEEE/ACM Transactions on Audio Speech & Language Processing* 24(12):2263–2276.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*. Springer, pages 525–542.

Xu Xiang, Yanmin Qian, and Kai Yu. 2017. Binary deep neural networks for speech recognition. *Proc. Interspeech 2017* pages 533–537.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329* .

Conversational Memory Network for Emotion Recognition in Dyadic Dialogue Videos

Devamanyu Hazarika
School of Computing,
National University of Singapore
hazarika@comp.nus.edu.sg

Soujanya Poria
Artificial Intelligence Initiative,
A*STAR, Singapore
sporia@ihpc.a-star.edu.sg

Amir Zadeh
Language Technologies
Institute, CMU, USA
abagherz@cs.cmu.edu

Erik Cambria
School of Computer Science and
Engineering, NTU, Singapore
cambria@ntu.edu.sg

Louis-Philippe Morency
Language Technologies
Institute, CMU, USA
morency@cs.cmu.edu

Roger Zimmermann
School of Computing,
National University of Singapore
rogerz@comp.nus.edu.sg

Abstract

Emotion recognition in conversations is crucial for the development of empathetic machines. Present methods mostly ignore the role of inter-speaker dependency relations while classifying emotions in conversations. In this paper, we address recognizing utterance-level emotions in dyadic conversational videos. We propose a deep neural framework, termed conversational memory network, which leverages contextual information from the conversation history. The framework takes a multimodal approach comprising audio, visual and textual features with gated recurrent units to model past utterances of each speaker into memories. Such memories are then merged using attention-based hops to capture inter-speaker dependencies. Experiments show an accuracy improvement of $3-4\%$ over the state of the art.

1 Introduction

Development of machines with emotional intelligence has been a long-standing goal of AI. With the increasing infusion of interactive systems in our lives, the need for empathetic machines with emotional understanding is paramount. Previous research in affective computing has looked at dialogues as an essential basis to learn emotional dynamics (Sidnell and Stivers, 2012; Poria et al., 2017a; Zhou et al., 2017).

Since the advent of Web 2.0, dialogue videos have proliferated across the internet through platforms like movies, webinars, and video chats. Emotion detection from such resources can benefit numerous fields like counseling (De Choudhury et al., 2013), public opinion mining (Cambria et al., 2017), financial forecasting (Xing et al., 2018), and intelligent systems such as smart homes and chatbots (Young et al., 2018).

In this paper, we analyze emotion detection in videos of dyadic conversations. A dyadic conversation is a form of a dialogue between two entities. We propose a conversational memory network (CMN), which uses a multimodal approach for emotion detection in utterances (a unit of speech bound by breathes or pauses) of such conversational videos.

Emotional dynamics in a conversation is known to be driven by two prime factors: self and inter-speaker emotional influence (Morris and Keltner, 2000; Liu and Maitlis, 2014). Self-influence relates to the concept of *emotional inertia*, i.e., the degree to which a person's feelings carry over from one moment to another (Koval and Kuppens, 2012). Inter-speaker emotional influence is another trait where the other person acts as an influencer in the speaker's emotional state. Conversely, speakers also tend to mirror emotions of their counterparts (Navarretta et al., 2016). Figure 1 provides an example from the dataset showing the presence of these two traits in a dialogue.

Existing works in the literature do not capitalize on these two factors. Context-free systems infer emotions based only on the current utterance in the conversation (Bertero et al., 2016). Whereas, state-of-the-art context-based networks like Poria et al., 2017b, use long short-term memory (LSTM) networks to model speaker-based context that suffers from incapability of long-range summarization and unweighted influence from context, leading to model bias.

Our proposed CMN incorporates these factors by using emotional context information present in the conversation history. It improves speaker-based emotion modeling by using memory networks which are efficient in capturing long-term

Proceedings of NAACL-HLT 2018, pages 2122–2132
New Orleans, Louisiana, June 1 - 6, 2018.

dependencies and summarizing task-specific details using attention models (Weston et al., 2014; Graves et al., 2014; Young et al., 2017).

Specifically, the memory cells of CMN are continuous vectors that store the context information found in the utterance histories. CMN also models interplay of these memories to capture inter-speaker dependencies.

CMN first extracts multimodal features (audio, visual, and text) for all utterances in a video. In order to detect the emotion of a particular utterance, say u_i, it gathers its histories by collecting previous utterances within a context window. Separate histories are created for both speakers. These histories are then modeled into memory cells using gated recurrent units (GRUs).

After that, CMN reads both the speaker's memories and employs attention mechanism on them, in order to find the most useful historical utterances to classify u_i. The memories are then merged with u_i using an addition operation weighted by the attention scores. This is done to model inter-speaker influences and dynamics. The whole cycle is repeated for multiple hops and finally, this merged representation of utterance u_i is used to classify its emotion category.

The contributions of this paper can be summarized as follows:

1. We propose an architecture, termed CMN, for emotion detection in a dyadic conversation that considers utterance histories of both the speaker to model emotional dynamics. The architecture is extensible to multi-speaker conversations in formats such as textual dialogues or conversational videos.

2. When applied to videos, we adopt a multimodal approach to extract diverse features from utterances. It also makes our model robust to missing information.

3. CMN provides a significant increase in accuracy of 3 – 4% over previous state-of-the-art networks. One variant called CMN$_{self}$ which does not consider the inter-speaker relation in emotion detection also outperforms the state of the art by a significant margin.

The remainder of the paper is organized as follows: Section 2 provides a brief literature review; Section 3 formalizes the problem statement; Section 4 describes the proposed method in detail; experimental results are covered in Section 5; finally, Section 6 provides concluding remarks.

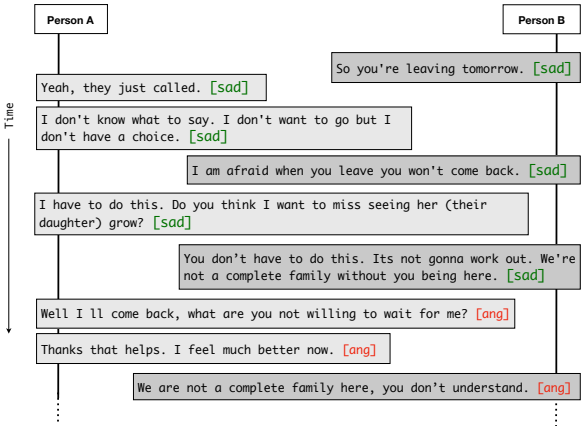


Figure 1: An abridged dialogue from the dataset. Person A (wife) is leaving B (husband) for a work assignment. Initially both A and B are emotionally driven by their own *emotional inertia*. In the end, *emotional influence* can be seen when B, despite being sad, reacts angrily to A's angry statement.

2 Related Works

Over the years, emotion recognition as an area of research has seen contributions from researchers across varied fields like signal processing, machine learning, cognitive and social psychology, natural language processing, etc. (Picard, 2010). Ekman, 1993, provided initial findings that related facial expressions as universal indicators of emotions. Datcu and Rothkrantz, 2008, 2011, showed the importance of acoustic cues in affect modeling.

A large section of researchers approaches emotion recognition from a multimodal learning perspective. Hence, many works used visual and audio features together for detecting affect (Busso et al., 2004; Castellano et al., 2008; Ranganathan et al., 2016). An in-depth review of the literature in these systems is provided by D'mello and Kory, 2015. Our work, which performs context-sensitive recognition (Wöllmer et al., 2010) uses three modalities: audio, visual and text. Recently, this combination of modalities has provided the best performance in affect recognition systems (Poria et al., 2017b; Wang et al., 2017; Tzirakis et al., 2017), thus motivating the use of a multimodal approach.

Previous works have focused on conversations as a resourceful event for emotion analysis. Rususvuori, 2013, provides an in-depth analysis on how emotions affect social interactions and conversations. In fact, significant works have attributed emotional dynamics as an interactive phe-

nomenon, rather than being within-person and one-directional (Richards et al., 2003; Hareli and Rafaeli, 2008). Such emotional dynamics are modeled by observing transition properties. Yang et al., 2011, study patterns for emotion transitions and show the evidence of emotional inertia. Xiaolan et al., 2013, use finite state machines to model transitions using stimuli and personality characteristics. Our work also tries to model emotional transitions using multimodal features. Unlike these works, however, we use memory networks to achieve the same.

The use of memory networks have been instrumental in the progress of multiple research problems, e.g., question-answering (Weston et al., 2014; Sukhbaatar et al., 2015; Kumar et al., 2016), machine translation (Bahdanau et al., 2014), speech recognition (Graves et al., 2014), and commonsense reasoning (Cambria et al., 2018). The repeated read and write to their memory cells is often coupled with attention modules, thus allowing it to filter only relevant memories.

Our model is loosely inspired from Sukhbaatar et al., 2015. Unlike their model, which directly encodes sentences into memories, we perform temporal sequence processing on our utterance histories using GRUs. We also extend their architecture to handle two speakers while keeping the possibility to add more. Finally, our model is different in the fact that we use multimodal features for input and processing.

3 Task Definition

Our goal is to infer the emotion of utterances present in a dyadic conversation. Let us define a dyadic conversation to be an asynchronous exchange of utterances between two persons P_a and P_b. Both the speakers speak a sequence of utterances U_a and U_b, respectively. Here, $U_\lambda = (s^1_\lambda, s^2_\lambda, ..., s^{l_\lambda}_\lambda)$ is ordered temporally, where s^i_λ is the i^{th} utterance by P_λ and l_λ is the total number of utterances spoken by person $P_\lambda,\ \lambda \in \{a, b\}$. Overall, the utterances by both speakers can be linearly ordered based on temporal occurrence as $(u_1, u_2, ...u_{l_a+l_b})$, where, $u_j \in U_a$ or U_b.

Our model takes as input an utterance u_i whose emotion category (Section 5.1) needs to be classified. To get its history, preceding K utterances of each person are separately collected as $hist_a$ and $hist_b$. Here, K serves as the length of the context window for history of u_i. Thus, for $\lambda \in \{a, b\}$:

$$hist_\lambda = \{u_j \mid u_j \in U_\lambda,\ j < i\}\ ,\ \mid hist_\lambda \mid \leq K \quad (1)$$

$hist_\lambda$ is also ordered temporally. At the beginning of the conversation, histories would have lesser than K utterances, i.e., $\mid hist_\lambda \mid < K$.

In the remaining sections, for brevity, we explain the processes using a subscript λ which can instantiate to either a or b, i.e., $\lambda \in \{a, b\}$.

4 Approach

We start by detailing the multimodal feature extraction scheme for all utterances followed by the mechanism to model emotional context using memory networks.

4.1 Multimodal Feature Extraction

The first phase of CMN is to extract multimodal features of all utterances in the conversations. The dyadic conversations are present in the form of videos. Each utterance of a particular conversation is thus a small segment of the full video. For each utterance, we extract features for the modes: audio, visual and text. The process of feature extraction for each mode is described below.

4.1.1 Textual Features Extraction

We extract features from the transcript of an utterance video using convolutional neural networks (CNNs). CNNs are effective in learning high level abstract representations of sentences from constituting words or n-grams (Kalchbrenner et al., 2014). To get our sentence representation, we use a simple CNN with one convolutional layer followed by max-pooling (Kim, 2014; Poria et al., 2016).

Specifically, the convolution layer consists filters of sizes $3, 4$ and 5 with 50 feature maps each. Max-pooling is employed on these feature maps with a pooling window of size 2. Finally, a fully connected layer is used with 100 neurons. The activations of this layer form our sentence representation t_u.

4.1.2 Audio Feature Extraction

To extract audio features we use openSMILE (Eyben et al., 2010). It is an open-source software which provides high dimensional audio vectors. These vectors comprise of features like loudness, Mel-spectra, MFCC, pitch, etc. Audio features play a significant role in providing information on the emotional state of a speaker (Song et al., 2004).

In fact, the literature shows that there exists a high correlation between many statistical measures of speech with speakers' emotion. For example, high pitch and fast speaking rate often denote anger while sadness associates low standard deviation of pitch and slow speech rate (Dellaert et al., 1996; Amir, 1998). In this work, we use the *IS13_ComParE*[1] config file which extracts a total of 6373 features for each utterance video. Z-standardization is performed for voice normalization and dimension of the audio vector is reduced to 100 using a fully-connected neural layer. This provides the final audio feature vector a_u.

4.1.3 Visual Feature Extraction

Facial expressions and visual surrounding provide rich emotional indicators. We use a 3D-CNN to capture these details from the utterance video. Apart from the benefits of extracting relevant features from each image frame, 3D-CNN also extracts spatiotemporal features across frames (Tran et al., 2015). This leads to the identification of emotional expressions like a smile or frown.

The working of a 3D-CNN is identical to its 2D counterpart with an input being a video v of dimension: $(3, f, h, w)$. Here, 3 represents the RGB channels and f, h, w are the number of frames, height, and width of each frame, respectively. For the convolution operation, a 3D filter f_l of dimension $(f_m, 3, f_d, f_h, f_w)$ is used where, $f_{[m/d/h/f]}$ represents number of feature maps, depth, height and width of the filter, respectively. Max-pooling is applied to the output of this convolution across a 3D sliding window of dimension (m_p, m_p, m_p).

In our model, we use 128 feature maps for 3D filters of size 5. For pooling, we set m_p to be 3 whose output is fed to a fully connected layer with 100 neurons. All the values are decided using hyperparameter tuning (see Section 5). For the input utterance, the activations of this layer form the video representation v_u.

Fusion: We perform feature level fusion to map the individual modalities to a joint space. This is done through a simple feature concatenation. Thus, the extracted features t_u, a_u and v_u are joined to form the utterance representation $u = [t_u; a_u; v_u]$ of dimension $d_{in} = 300$. This multimodal representation is generated for all utterances in a conversation.

[1] http://audeering.com/technology/opensmile

Literature consists of numerous fusion techniques for multimodal data (Atrey et al., 2010; Zadeh et al., 2017; Poria et al., 2017c). Exploring these on CMN, however, is beyond the scope of this paper and left as a future work.

4.2 Conversational Memory Network

For classifying the emotion of an utterance u_i, its corresponding histories ($hist_a$ and $hist_b$) are taken. Each history $hist_\lambda$ contains the preceding K utterances by person P_λ (see Section 3). Here, both u_i and utterances in the histories are represented using their multimodal feature vectors of dimension $\mathcal{R}^{d_{in}}$ (Figure 2).

The histories are first modeled into memory cells using GRUs. This provides the memories with context information summarized by the GRU. We call this step as memory representation. Following cognitive evidence of self-emotional dynamics, we model separate memory cells for each person. Thus, identical but separate computations are performed on both histories. From these memories, content relevant to utterance u_i is then filtered out using attention mechanism over multiple input/output hops. At each hop, both memories are accumulated and merged with u_i to model inter-speaker emotional dynamics. First, we describe our model as a single layer memory network which runs one hop operation on the memories.

4.2.1 Single Layer

Here, we explain the representation scheme of the memories for both histories and the input/output operations on them along with attention mechanism. The memory representation for each history is generated using a GRU for modeling emotion transitions. First, we define the GRU cell.

Gated Recurrent Unit: GRUs are a gating mechanism in recurrent neural networks introduced by (Cho et al., 2014). Similar to an LSTM (Hochreiter and Schmidhuber, 1997), GRU provides a simpler computation with similar performance. At any timestep t, it utilizes two gates r_t (*reset gate*) and z_t (*update gate*) to control the combination criteria with current input utterance u_t and previous hidden state s_{t-1}.

The new state s_t is computed as:

$$z_t = \sigma(V^z.u_t + W^z.s_{t-1} + b^z) \quad (2)$$

$$r_t = \sigma(V^r.u_t + W^r.s_{t-1} + b^r) \quad (3)$$

$$h_t = tanh(V^h.u_t + W^h.(s_{t-1} \otimes r_t) + b^h) \quad (4)$$

$$s_t = (1 - z_t) \otimes h_t + z_t \otimes s_{t-1} \quad (5)$$

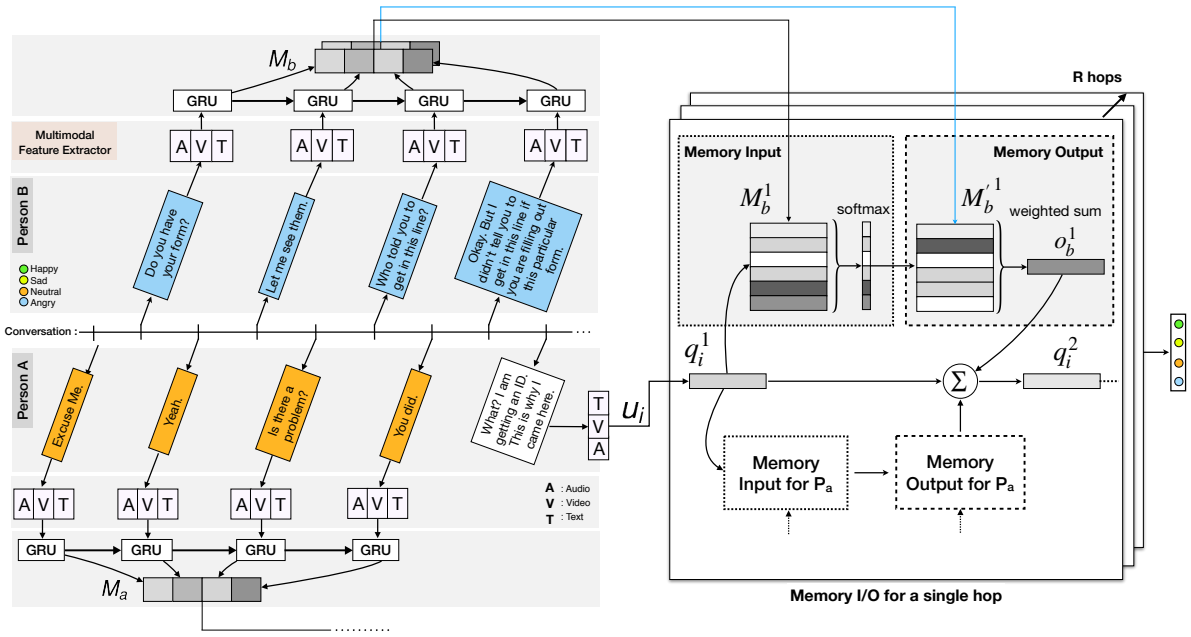


Figure 2: Overall architecture of proposed model: CMN. First, multimodal representations are extracted for each utterance and then the previous $K = 4$ utterances by both persons are used to model the histories using GRUs. For each person, $R+1$ different GRUs are used to represent $M_\lambda^{(r)}$ for all R hops. Then, attention based filtering using multiple memory hops is performed. Finally, Person A's utterance u_i is classified to predict its emotion category.

Here, V, W and b are parameter matrices and vector and $\otimes$ represents element-wise multiplication. The above equations can be summarized as: $s_t = GRU_\lambda(s_{t-1}, u_t)$.

Memory Representation: For each $\lambda \in \{a, b\}$, a memory representation $M_\lambda = [m_\lambda^1, ..., m_\lambda^K]$ for $hist_\lambda$ is generated using a GRU. To grasp the temporal context, the K utterances in $hist_\lambda$ are framed as a sequence (starting from the oldest one) and fed to the GRU$_\lambda$. At each timestep $t \in [1, K]$, the GRU$_\lambda$'s internal state s_t (equation 5) forms the t^{th} memory cell m_λ^t of memory representation M_λ.

Memory Input: This step takes the memory representation M_λ and performs an attention mechanism on it, resulting in an attention vector $p_\lambda \in \mathcal{R}^K$. First, the current utterance u_i is embedded into a vector q_i of dimension $\mathcal{R}^d$ using a projection matrix $B \in \mathcal{R}^{d \times d_{in}}$. To find the relevance of each memory m_λ^t's context with q_i, a match between both is computed.
We do this by taking an inner product as follows:

$$q_i = B.u_i \tag{6}$$

$$p_\lambda^t = softmax(q_i^T.m_\lambda^t) \tag{7}$$

Here, $softmax(x_i) = e^{x_i} / \sum_j e^{x_j}$ and attention vector $p_\lambda = \{p_\lambda^t\}$ is a probability distribution over the input memories $M_\lambda = \{m_\lambda^t\}$ for $t \in [1, K]$.

Memory Output: First a new set of memories are created using another GRU$_\lambda^{'}$ to get new memory representation $M_\lambda^{'} = \{(m_\lambda^t)^{'}\}$. An output representation $o_\lambda \in \mathcal{R}^d$ is then generated using the weighted sum of attention vector p_λ and new memory $M_\lambda^{'}$ as follows:

$$o_\lambda = \sum_t p_\lambda^t.(m_\lambda^t)^{'} = M_\lambda^{'}.p_\lambda \tag{8}$$

Thus, the output representation o_λ contains weighted contextual summary accumulated from the memory.

Final Prediction: To generate the predictions for the current utterance u_i, we combine the output representations of both persons: o_a and o_b with u_i's representation q_i and perform an affine transformation using matrix W_o. Softmax is applied to this final vector to get the emotion predictions,

$$\hat{y} = softmax(W_o.(q_i + o_a + o_b)) \tag{9}$$

Categorical cross-entropy is used as the loss:

$$Loss = \frac{-1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} y_{i,j} \; log_2(\hat{y}_{i,j}) \tag{10}$$

Here, N denotes total utterances across all videos and C is the number of emotion categories. y_i is the one-hot vector ground truth of i^{th} utterance from the training set and $\hat{y}_{i,j}$ is its predicted probability of belonging to class j.

4.2.2 Multiple Layers

Many recent works on memory networks adopt a multiple hop scheme in their network. This repeated input and output cycle on the memories along with a soft attention module, leads to a refined representation of the memories (Sukhbaatar et al., 2015; Kumar et al., 2016). Motivated by these works, we extend our model to perform R hops on the memories. This is done by stacking the single hop layers (Section 4.2.1) as follows:

- At a particular hop r, the output memory of the previous hop $M_\lambda^{'(r-1)}$ is used as the input memory of the current hop $M_\lambda^{(r)}$. Output memory of current r^{th} hop is generated using a new $GRU_\lambda^{(r)}$. This constraint of sharing parameters adjacently between layers is added for reduction in total parameters and ease of training.
- At every hop, the query utterance u_i's representation q_i is updated as:

$$q_i^{(r+1)} = q_i^{(r)} + o_a^{(r)} + o_b^{(r)} \quad (11)$$

  $o_\lambda^{(r)}$ is calculated as per equation 8 using $M_\lambda^{'(r)}$.
- After R hops, the final prediction is done using equation 9 as: $\hat{y} = softmax(W_o.(q_i^{(R+1)}))$. Algorithm 1 summarizes the overall CMN network.

Algorithm 1 Conversational Memory Network

procedure CMN($u_i, hist_a, hist_b, K, R$) ▷ predict the emotion of u_i
  $q_i^{(1)} \leftarrow B.u_i$
  $M_\lambda^{'(0)} \leftarrow GRU_\lambda^{(0)}(hist_\lambda)$
  for `r in [1,R]` **do** ▷ Multi-hop memory I/O
    $M_\lambda^{(r)} \leftarrow M_\lambda^{'(r-1)}$
    $M_\lambda^{'(r)} \leftarrow GRU_\lambda^{(r)}(hist_\lambda)$
    $p_\lambda \leftarrow softmax(\ (q_i^r)^T.M_\lambda^{(r)}\ )$ ▷ Memory in
    $o_\lambda^{(r)} \leftarrow M_\lambda^{'(r)}.p_\lambda$ ▷ Memory out
    $q_i^{(r+1)} \leftarrow q_i^{(r)} + o_a^{(r)} + o_b^{(r)}$ ▷ Query update
  return $\hat{y} \leftarrow softmax(W_o.(q_i^{(R+1)}))$ ▷ Prediction

5 Experiments

5.1 Dataset

We perform experiments on the IEMOCAP dataset[2] (Busso et al., 2008). It is a multimodal database of 10 speakers (5 male and 5 female) involved in two-way dyadic conversations. A pair of speakers is given multiple conversation scenarios which are grouped in a single session. All the conversations are segmented into utterances. Each utterance is annotated using the following emotion categories: anger, happiness, sadness, neutral, excitement, frustration, fear, surprise, and other. However, in our experiments, we consider the first four categories. This is done to compare our method with state-of-the-art frameworks (Poria et al., 2017b; Rozgic et al., 2012). The dataset provides rich video and audio samples for all the utterances along with transcriptions.

Apart from these emotional states, we also investigate the valence and arousal degrees of each utterance. IEMOCAP provides labels for both these attributes on a 5-point Likert scale. Following Aldeneh et al., 2017, we convert the attributes into 3 categories, namely, *low* ($\leq$ 2), *medium* ($>$ 2 and $<$ 4) and *high* ($\geq$ 4). The dataset configuration for the experiments is obtained from Poria et al. (2017b). The first 8 speakers (Session 1 - 4) compose the training fold while the last session is used as the testing fold. Overall, the training and testing set comprises of 4290 utterances (120 conversational videos) and 1208 utterances (31 conversational videos), respectively. There is no speaker overlap in the training and testing set to make the model person-independent.

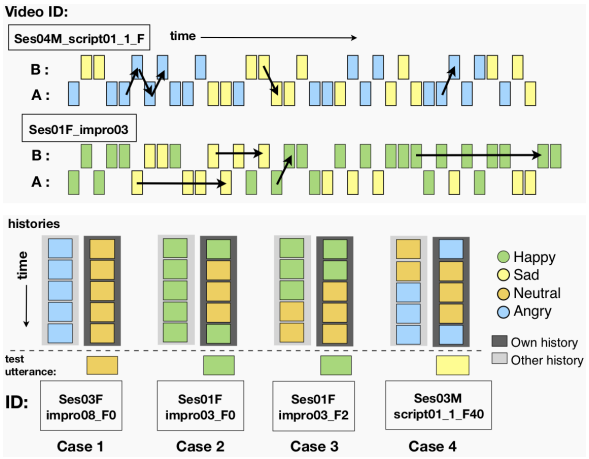


Figure 3: Each block represents an utterance and the blocks are ordered as per temporal occurrence. Color scheme identifies their corresponding emotions. The arrows denote emotional influence directions.

5.2 Emotional Influence Patterns

In this section, we perform dataset exploration to check the existence of emotional influences. Figure 3a) presents the emotion sequence of two videos

[2] http://sail.usc.edu/iemocap/

sampled from the dataset. Both videos show the presence of self and inter-speaker emotional influences. Visual exploration of videos from the dataset reveal significant existence of such instances in the conversations. To provide quantitative evidence of the emotional influence patterns, we curate a non exhaustive list of possible cases of influence. For all utterances in the dataset, we sample their histories by setting $K = 5$, i.e., five previous utterances (as per availability) from both speakers.

Cases 1 and 2 (Figure 3) represent scenarios when the emotion of current utterance is influenced by self or the other person respectively. In case 3, the utterance has relevant content in the histories that do not precede immediately. An effective attention mechanism provides the capability to capture this pattern. Finally case 4 presents the situation when the utterance is independent of the history. Such situations are indicative from the content of the utterance which often deviates from the previous topic of discussion or introduces a new information. Table 1 presents a statistical summary of these cases present in the dataset. From the table it can be seen that a large section of the dataset demonstrate these influence patterns. This provides motivation to explicitly model these patterns. We thus hypothesize that models that are able to capture these cases would have superior emotion inference capabilities.

This passive exploration is a label-based analysis which is performed as a sanity check. Needless to say, existence of some false positive patterns at the label level is imminent. On the other hand, our model CMN is content-based which enables it to mine intricate patterns from the utterance histories.

5.2.1 Training Details

We use 10% of the training set as a held-out validation set for hyperparameter tuning. To optimize the parameters, we use Stochastic Gradient Descent (SGD) optimizer, starting with an initial learning rate (lr) of 0.01. An annealing approach halves the lr every 20 epochs and termination is decided using an early-stop measure with a patience of 12 by monitoring the validation loss. Gradient clipping is used for regularization with a norm set to 40. Hyperparameters are decided using a Random Search (Bergstra and Bengio, 2012). Based on validation performance, context window length K is set to be 40 and the number of hops R is fixed at 3 hops. If K previous utterances are unavailable, then null utterances are added at the beginning of the history sequence. The dimension size of the memory cells d is set as 50.

	Case 1	*Case 2*	*Case 3*	*Case 4*
Percentage	63.77	40.44	30.97	16.24

Table 1: Percentage of occurrence of different cases in the dataset as mentioned in Section 5.2. All cases are analyzed with $K = 5$. Utterances whose history has atleast 3 similar emotion labels in either own history or the history of the other person, is counted in case 1 or 2, respectively. Case 3 is considered when the utterance's emotion is found in atleast 3 utterances which occur before the second past-utterance of each history. Case 4 is considered when no history has the emotion label of the current utterance.

5.2.2 Baselines

We compare CMN with the following baselines:

SVM-ensemble: A strong context-free benchmark model which uses similar multimodal approach on an ensemble of trees. Each node represents binary support vector machines (SVM) (Rozgic et al., 2012).

bc-LSTM: A bi-directional LSTM equipped with hierarchical fusion, proposed by Poria et al., 2017b. It is the present state-of-the-art method. The model uses context features from unimodal LSTMs and its concatenation is fed to a final LSTM for classification. For fair comparison in an end-to-end learning paradigm, we remove the penultimate SVM of this model. The model doesn't accommodate inter-speaker dependencies.

Memn2n: The original memory network as proposed by Sukhbaatar et al., 2015. Contrasting to CMN, the model generates the memory representations for each historical utterance using an embedding matrix B as used in equation 7, without sequential modeling. Thus for utterance u_i, both memories are created as M_λ using $\{m^t_\lambda = B.u_t \mid u_t \in hist_\lambda \text{ and } t \in [1, K]\}$ for $\lambda \in \{a, b\}$.

CMN$_{Self}$: In this baseline, we use only self history for classifying emotion of utterance u_i. Thus, if u_i is spoken by person P_a, then only $hist_a$ is considered. Clearly, this variant is also incapable of modeling inter-speaker dependencies.

CMN$_{NA}$: Single layer variant of the CMN with no attention module. Thus, its output o_λ (equation 8) is generated using a uniform probability distribution p_λ, i.e., $\{p^t_\lambda = \frac{1}{K}\}^K_{t=1}$.

5.3 Results

Table 2 presents the performances of CMN and its variants along with the state-of-the-art mod-

Models	hops	history	Emotion Categories					Valence		Arousal	
			Happiness	*Sadness*	*Neutral*	*Anger*	*WAA*	*WAA*	*UAR*	*WAA*	*UAR*
SVM-ensemble[1]	-	single	72.40	61.90	58.10	73.10	69.50	-	-	-	-
bc-LSTM[2]	-	single	74.21	76.50	66.31	75.68	74.31	64.3	62.3	70.1	45.0
Memn2n	1	dual	72.36	76.16	66.93	80.23	74.17	-	-	-	-
	3	dual	75.03	76.36	66.45	81.59	75.08	65.3	64.0	71.5	45.6
CMN_{Self}	3	single	77.14†	76.99	66.99	87.26†	76.54†	65.5	64.0	72.1	47.1
CMN_{NA}	1	dual	74.33	76.93	66.49	86.29†	75.77	65.6	64.2	71.6	46.3
CMN	3	dual	**81.75**†	77.73	67.32	**89.88**†	**77.62**†	**66.1**	**64.3**	**72.2**	**47.6**

[1](Rozgic et al., 2012), [2](Poria et al., 2017b). †: significantly better than bc-LSTM[1]

Table 2: Comparison of CMN and its variants with state-of-the-art models (Section 5.2.2). All results use multimodal features. We report scores using weighted accuracy (WAA) and unweighted recall (UAR). UAR is a popular metric that is used when dealing with imbalanced classes (Rosenberg, 2012). Results are an average of 10 runs with varied weight initializations. We assert significance when $p < 0.05$ under McNemar's test.

els. CMN succeeds over both neural (Poria et al., 2017b) and SVM-based (Rozgic et al., 2012) methods by 3.3% and 8.12%, respectively. Improvement in performance is seen for all emotions over the ensemble-SVM based method. A similar trend is seen with bc-LSTM (Poria et al., 2017b), where our model does explicitly well for the active emotions *happiness* and *anger*. This trend suggests that CMN is capable of capturing inter-speaker emotional influences which are often seen in the presence of such active emotions.

The importance of sequential processing of the histories using a recurrent neural network (in our case, a GRU) is evidenced by the poorer performance of Memn2n with respect to CMN. This suggests that gathering contexts temporally through sequential processing is indeed a superior method over non-temporal memory representations. CMN_{self} which uses only single history channel also provides lesser performance when compared to CMN. This signifies the role of inter-speaker influences that often moderate the emotions of the current utterance. Overall, predictions on valence and arousal levels also show similar results which reinforce our hypothesis of CMN's ability to model emotional dynamics.

Hyperparameters: Figure 4 provides a summary of the performance trend of our model for different values of the hyperparameters K (context window length) and Q (number of hops). In the first graph, as K increases, more past-utterances are provided to the model as memories. The performance maintains a positive correlation with K. This trend supplements our intuition that the historical context acts as an essential resource to model emotional dynamics. Given enough history, the performance saturates. The second graph shows that multiple hops on the histories indeed lead to an improvement in performance. The attention-based filtering in each hop provides a refined context representation of the histories. Models with hops in the range of 3 – 10 outperform the single layer variant. However, each added hop contributes a new set of parameters for memory representation, leading to an increase in total parameters of the model and making it susceptible to overfitting. This effect is evidenced in the figure where higher hops lead to a dip in performance.

Multimodality: Table 3 summarizes the performance of unimodal and multimodal variants of the baselines along with CMN. As seen in the table, text modality performes best out of the three. This is in contrast to Rozgic et al. 2012 where audio provides the best performance. A possible reason for

Models	*unimodal audio*	*unimodal visual*	*unimodal text*	*trimodal*
SVM-ensemble	60.8	51.5	48.5	69.5‡
bc-LSTM	62.2	56.1	72.5	74.3‡
Memn2n	63.0	61.8	72.6	75.0‡
CMN_{self}	63.1	62.5	73.0	76.5‡
CMN_{NA}	62.4	60.9	74.1	75.7
CMN	**65.3**	**64.2**	**74.2**	**77.6**‡

‡: significantly better than unimodals ($p < 0.05$)

Table 3: Comparison of CMN to all the baselines in different modalities. Weighted accuracy is used as the metric.

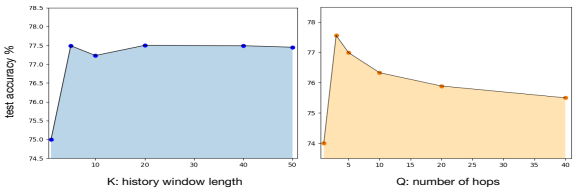
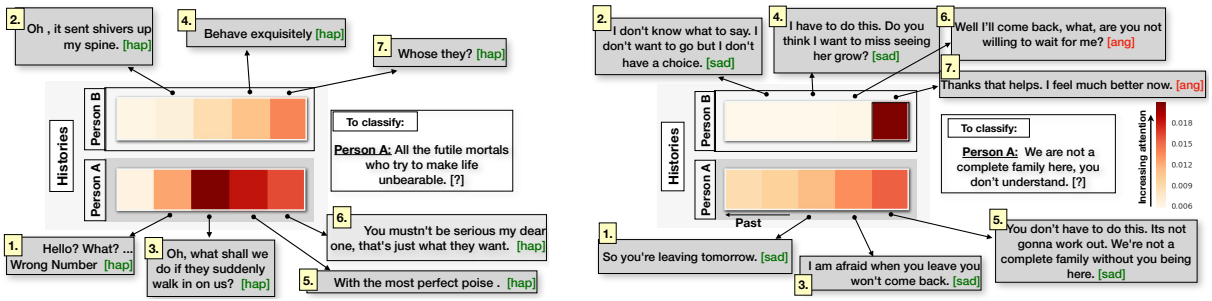


Figure 4: Performance trends of our model with different values of K (history length) and Q (number of hops). While K is varied, Q is set to be 3. Similarly, $K = 20$ when Q varies.



(a) Correct label: *happiness* (b) Correct label: *anger*

Figure 5: Average attention vectors across 3 hops for both memories for a given test utterance.

this shift is the improved representational scheme of the textual modality. Text tends to have lesser noisy signals as opposed to audio-visual sources, thus providing better features in the joint representation. Overall, multimodal systems outperform the unimodal variants justifying the design of CMN as a multimodal system.

Table 3 also showcases the superiority of CMN and its variants over bc-LSTM. The proposed model achieves better performance over the state of the art in all the unimodal and multimodal segments. This asserts the importance of the memory-network framework and its ability to effectively store context information.

Role of Attention: Attention module plays a vital role in memory refinement. This is also observed in Table 2, where CMN$_{NA}$ provides inferior performance over CMN. With the uniform weight, all the memory cells in both memories M_a and M_b equally contribute to the output representation. This incorporates irrelevant information from the perspective of emotional context.

Case Study: We perform qualitative visualization of the attention module by applying it on the testing set. Figure 5a represents a conversation where both the speakers are in an excited and jolly mood. Person A, in particular, drives the dialogue with less influence from Person B. To classify the test utterance of A, the attention module of CMN successfully focuses on the utterances $1, 3, 5$ which had triggered the speaker's positive mood in the video. This shows CMN's capacity to model speaker-based emotions. Also, at the textual level, utterances 3 and 6 do not seem to depict a happy mood. However, audio and visual sources provide contrasting evidence which helps CMN to correctly model them as utterances spoken with happiness. This shows the advantage of a multimodal system.

In Figure 5b we reiterate through the dialogue presented in Figure 1. As shown, Person A converses in a sad mood (utterances $1, 3, 5$ in Fig 5b), bounded by the grief of his wife's departure. But when he expresses his inhibitions, his wife B reacts in an angry and sarcastic manner (utterance 7). This ignites an emotional shift for A who then replies angrily. In this example, CMN is able to focus on utterance 7 spoken by B to anticipate A's test utterance to be an angry statement, thus showing its ability to model inter-speaker influences. However, there are cases where our model fails, e.g., in the absence of historical utterances as this forces attention to focus on null memories.

6 Conclusion

In this paper, we presented a deep neural framework that identifies emotions for utterances in dyadic conversational videos. Our results suggest that leveraging context information from utterance histories and representing them as memories indeed helps to better recognize emotions. Performing speaker-specific modeling and considering inter-speaker influences also helps in capturing emotional dynamics.

This work also showed the importance of attention mechanism in filtering relevant contextual information from utterance histories and, hence, paved the path to the development of more efficient and human-like dialogue systems.

Acknowledgement

This research was supported in part by the National Natural Science Foundation of China under Grant no. 61472266 and by the National University of Singapore (Suzhou) Research Institute, 377 Lin Quan Street, Suzhou Industrial Park, Jiang Su, People's Republic of China, 215123.

References

Zakaria Aldeneh, Soheil Khorram, Dimitrios Dimitriadis, and Emily Mower Provost. 2017. Pooling acoustic and lexical features for the prediction of valence.

Noam Amir. 1998. Towards an automatic classification of emotions in speech. *ICSLP*, pages 699–702.

Pradeep K Atrey, M Anwar Hossain, Abdulmotaleb El Saddik, and Mohan S Kankanhalli. 2010. Multimodal fusion for multimedia analysis: a survey. *Multimedia systems*, 16(6):345–379.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305.

Dario Bertero, Farhad Bin Siddique, Chien-Sheng Wu, Yan Wan, Ricky Ho Yin Chan, and Pascale Fung. 2016. Real-time speech emotion and sentiment recognition for interactive dialogue systems. In *EMNLP*, pages 1042–1047.

Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower, Samuel Kim, Jeannette N Chang, Sungbok Lee, and Shrikanth S Narayanan. 2008. Iemocap: Interactive emotional dyadic motion capture database. *Language resources and evaluation*, 42(4):335.

Carlos Busso, Zhigang Deng, Serdar Yildirim, Murtaza Bulut, Chul Min Lee, Abe Kazemzadeh, Sungbok Lee, Ulrich Neumann, and Shrikanth Narayanan. 2004. Analysis of emotion recognition using facial expressions, speech and multimodal information. In *ICMI*, pages 205–211.

Erik Cambria, Soujanya Poria, Alexander Gelbukh, and Mike Thelwall. 2017. Sentiment analysis is a big suitcase. *IEEE Intelligent Systems*, 32(6):74–80.

Erik Cambria, Soujanya Poria, Devamanyu Hazarika, and Kenneth Kwok. 2018. SenticNet 5: Discovering conceptual primitives for sentiment analysis by means of context embeddings. In *AAAI*.

Ginevra Castellano, Loic Kessous, and George Caridakis. 2008. Emotion recognition through multiple modalities: face, body gesture, speech. *Affect and emotion in human-computer interaction*, pages 92–103.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Dragos Datcu and L Rothkrantz. 2008. Semantic audio-visual data fusion for automatic emotion recognition. *Euromedia'2008*.

Dragoş Datcu and Léon JM Rothkrantz. 2011. Emotion recognition using bimodal data fusion. In *International Conference on Computer Systems and Technologies*, pages 122–128. ACM.

Munmun De Choudhury, Michael Gamon, Scott Counts, and Eric Horvitz. 2013. Predicting depression via social media. *ICWSM*, 13:1–10.

Frank Dellaert, Thomas Polzin, and Alex Waibel. 1996. Recognizing emotion in speech. In *ICSLP*, volume 3, pages 1970–1973.

Sidney K D'mello and Jacqueline Kory. 2015. A review and meta-analysis of multimodal affect detection systems. *ACM Computing Surveys (CSUR)*, 47(3):43.

Paul Ekman. 1993. Facial expression and emotion. *American psychologist*, 48(4):384.

Florian Eyben, Martin Wöllmer, and Björn Schuller. 2010. Opensmile: the munich versatile and fast open-source audio feature extractor. In *International Conference on Multimedia*, pages 1459–1462. ACM.

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

Shlomo Hareli and Anat Rafaeli. 2008. Emotion cycles: On the social influence of emotion in organizations. *Research in organizational behavior*, 28:35–59.

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

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *ACL 2014*, volume 1, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP 2014*, pages 1746–1751.

Peter Koval and Peter Kuppens. 2012. Changing emotion dynamics: individual differences in the effect of anticipatory social stress on emotional inertia. *Emotion*, 12(2):256.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *ICML*, pages 1378–1387.

Feng Liu and Sally Maitlis. 2014. Emotional dynamics and strategizing processes: A study of strategic conversations in top team meetings. *Journal of Management Studies*, 51(2):202–234.

Michael W Morris and Dacher Keltner. 2000. How emotions work: The social functions of emotional expression in negotiations. *Research in organizational behavior*, 22:1–50.

Costanza Navarretta, K Choukri, T Declerck, S Goggi, M Grobelnik, and B Maegaard. 2016. Mirroring facial expressions and emotions in dyadic conversations. In *LREC*.

Rosalind W Picard. 2010. Affective computing: from laughter to ieee. *IEEE Transactions on Affective Computing*, 1(1):11–17.

Soujanya Poria, Erik Cambria, Rajiv Bajpai, and Amir Hussain. 2017a. A review of affective computing: From unimodal analysis to multimodal fusion. *Information Fusion*, 37:98–125.

Soujanya Poria, Erik Cambria, Devamanyu Hazarika, Navonil Majumder, Amir Zadeh, and Louis-Philippe Morency. 2017b. Context-dependent sentiment analysis in user-generated videos. In *ACL 2017*, volume 1, pages 873–883.

Soujanya Poria, Erik Cambria, Devamanyu Hazarika, Navonil Mazumder, Amir Zadeh, and Louis-Philippe Morency. 2017c. Multi-level multiple attentions for contextual multimodal sentiment analysis. In *ICDM 2017*, pages 1033–1038. IEEE.

Soujanya Poria, Erik Cambria, Devamanyu Hazarika, and Prateek Vij. 2016. A deeper look into sarcastic tweets using deep convolutional neural networks. In *COLING 2016*, pages 1601–1612.

Hiranmayi Ranganathan, Shayok Chakraborty, and Sethuraman Panchanathan. 2016. Multimodal emotion recognition using deep learning architectures. In *WACV*, pages 1–9. IEEE.

Jane M Richards, Emily A Butler, and James J Gross. 2003. Emotion regulation in romantic relationships: The cognitive consequences of concealing feelings. *Journal of Social and Personal Relationships*, 20(5):599–620.

Andrew Rosenberg. 2012. Classifying skewed data: Importance weighting to optimize average recall. In *INTERSPEECH 2012*.

Viktor Rozgic, Sankaranarayanan Ananthakrishnan, Shirin Saleem, Rohit Kumar, and Rohit Prasad. 2012. Ensemble of svm trees for multimodal emotion recognition. In *APSIPA ASC*, pages 1–4.

Johanna Ruusuvuori. 2013. Emotion, affect and conversation. *The handbook of conversation analysis*, pages 330–349.

Jack Sidnell and Tanya Stivers. 2012. *The handbook of conversation analysis*, volume 121. John Wiley & Sons.

Mingli Song, Jiajun Bu, Chun Chen, and Nan Li. 2004. Audio-visual based emotion recognition-a new approach. In *CVPR*, volume 2.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *NIPS 2015*, pages 2440–2448.

Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. 2015. Learning spatiotemporal features with 3d convolutional networks. In *ICCV 2015*, pages 4489–4497.

Panagiotis Tzirakis, George Trigeorgis, Mihalis A Nicolaou, Björn W Schuller, and Stefanos Zafeiriou. 2017. End-to-end multimodal emotion recognition using deep neural networks. *IEEE JSTSP*, 11(8):1301–1309.

Haohan Wang, Aaksha Meghawat, Louis-Philippe Morency, and Eric P Xing. 2017. Select-additive learning: Improving generalization in multimodal sentiment analysis. In *ICME*, pages 949–954.

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

Martin Wöllmer, Angeliki Metallinou, Florian Eyben, Björn Schuller, and Shrikanth S Narayanan. 2010. Context-sensitive multimodal emotion recognition from speech and facial expression using bidirectional lstm modeling. In *INTERSPEECH 2010*.

Peng Xiaolan, Xie Lun, Liu Xin, and Wang Zhiliang. 2013. Emotional state transition model based on stimulus and personality characteristics. *China Communications*, 10(6):146–155.

Frank Xing, Erik Cambria, and Roy Welsch. 2018. Natural language based financial forecasting: A survey. *Artificial Intelligence Review*.

Liang Yang, Hong-fei LIN, and Wei GUO. 2011. Text-based emotion transformation analysis. *Computer Engineering & Science*, 9:026.

Tom Young, Erik Cambria, Iti Chaturvedi, Hao Zhou, Subham Biswas, and Minlie Huang. 2018. Augmenting end-to-end dialog systems with commonsense knowledge. In *AAAI*.

Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. 2017. Recent trends in deep learning based natural language processing. *arXiv preprint arXiv:1708.02709*.

Amir Zadeh, Minghai Chen, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2017. Tensor fusion network for multimodal sentiment analysis. In *EMNLP 2017*, pages 1103–1114.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2017. Emotional chatting machine: Emotional conversation generation with internal and external memory. *arXiv:1704.01074*.

How Time Matters: Learning Time-Decay Attention for Contextual Spoken Language Understanding in Dialogues

Shang-Yu Su[†] **Pei-Chieh Yuan**[†] **Yun-Nung Chen**[⋆]
[†]Department of Electrical Engineering
[⋆]Department of Computer Science and Information Engineering
National Taiwan University
{r05921117,b03901134}@ntu.edu.tw y.v.chen@ieee.org

Abstract

Spoken language understanding (SLU) is an essential component in conversational systems. Most SLU components treat each utterance independently, and then the following components aggregate the multi-turn information in the separate phases. In order to avoid error propagation and effectively utilize contexts, prior work leveraged history for contextual SLU. However, most previous models only paid attention to the related content in history utterances, ignoring their temporal information. In the dialogues, it is intuitive that the most recent utterances are more important than the least recent ones, in other words, time-aware attention should be in a decaying manner. Therefore, this paper designs and investigates various types of time-decay attention on the sentence-level and speaker-level, and further proposes a flexible universal time-decay attention mechanism. The experiments on the benchmark Dialogue State Tracking Challenge (DSTC4) dataset show that the proposed time-decay attention mechanisms significantly improve the state-of-the-art model for contextual understanding performance[1].

1 Introduction

Spoken dialogue systems that can help users to solve complex tasks such as booking a movie ticket have become an emerging research topic in artificial intelligence and natural language processing areas. With a well-designed dialogue system as an intelligent personal assistant, people can accomplish certain tasks more easily via natural language interactions. Today, there are several virtual intelligent assistants, such as Apple's Siri, Google's Home, Microsoft's Cortana, and Amazon's Echo. Recent advance of deep learning has inspired many applications of neural models to dialogue systems (Wen et al., 2017; Bordes et al., 2017; Dhingra et al., 2017; Li et al., 2017).

A key component of a dialogue system is a spoken language understanding (SLU) module—it parses user utterances into semantic frames that capture the core meaning (Tur and De Mori, 2011). A typical pipeline of SLU is to first decide the domain given the input utterance, and based on the domain, to predict the intent and to fill associated slots corresponding to a domain-specific semantic template, where each utterance is treated independently (Hakkani-Tür et al., 2016; Chen et al., 2016b,a; Wang et al., 2016). To overcome the error propagation and further improve understanding performance, the contextual information has been shown useful (Bhargava et al., 2013; Xu and Sarikaya, 2014; Chen et al., 2015; Sun et al., 2016). Prior work incorporated the dialogue history into the recurrent neural networks (RNN) for improving domain classification, intent prediction, and slot filling (Xu and Sarikaya, 2014; Shi et al., 2015; Weston et al., 2015; Chen et al., 2016c). Recently, Chi et al. (2017) and Zhang et al. (2018) demonstrated that modeling speaker role information can learn the notable variance in speaking habits during conversations in order to benefit understanding.

In addition, neural models incorporating attention mechanisms have had great successes in machine translation (Bahdanau et al., 2014), image captioning (Xu et al., 2015), and various tasks. Attentional models have been successful because they separate two different concerns: 1) deciding which input contexts are most relevant to the output and 2) actually predicting an output given the most relevant inputs. For example, the highlighted current utterance from the tourist, "*uh on august*", in the conversation of Figure 1 is to respond the question about WHEN, and the content-

[1]The source code is at: https://github.com/MiuLab/Time-Decay-SLU.

Proceedings of NAACL-HLT 2018, pages 2133–2142
New Orleans, Louisiana, June 1 - 6, 2018.

Guide: and you were saying that you wanted to come to singapore	FOL-CONFIRM; FOL-INFO
Guide: uh maybe can i have a little bit more details like uh when will you be coming	QST-INFO; QST-WHEN
Guide: and like who will you be coming with	QST-WHO
Tourist: uh yes	FOL-CONFIRM
Tourist: um i'm actually planning to visit	RES-WHEN
Tourist: uh on august	**RES-WHEN**

Figure 1: The human-human conversational utterances and their associated semantic labels from DSTC4.

aware contexts that can help current understanding are the first two utterances from the guide "*and you were saying that you wanted to come to singapore*" and "*un maybe can i have a little bit more details like uh when will you be coming*". Previous work proposed an end-to-end time-aware attention network to leverage both contextual and temporal information for spoken language understanding and achieved the significant improvement, showing that the temporal attention can guide the attention effectively (Chen et al., 2017). However, the time-aware attention function is an inflexible hand-crafted setting, which is a fixed function of time for assessing the attention.

This paper focuses on investigating various flexible time-aware attention mechanism in neural models with contextual information and speaker role modeling for language understanding. The contributions are three-fold:

- This paper investigates different time-aware attention mechanisms and provides guidance for the future research about designing the time-aware attention function.
- This paper proposes an end-to-end learnable universal time-decay mechanism with great flexibility of modeling temporal information for diverse dialogue contexts.
- The proposed model achieves the state-of-the-art understanding performance in the dialogue benchmark DSTC dataset.

2 The Proposed Framework

The model architecture is illustrated in Figure 2. First, the previous utterances are fed into the contextual model to encode into the history summary, and then the summary vector and the current utterance are integrated for helping understanding. The contextual model leverages the attention mechanisms highlighted in red, which implements different attention functions for sentence and speaker role levels. The whole model is trained in an end-to-end fashion, where the history summary vector and the attention weights are automatically learned based on the downstream SLU task. The objective of the proposed model is to optimize the conditional probability of the intents given the current utterance, $p(\hat{\mathbf{y}} \mid \mathbf{x})$, by minimizing the cross-entropy loss.

2.1 Speaker Role Contextual Language Understanding

Given the current utterance $\mathbf{x} = \{w_t\}_1^T$, the goal is to predict the user intents of $\mathbf{x}$, which includes the speech acts and associated attributes. We apply a bidirectional long short-term memory (BLSTM) model (Schuster and Paliwal, 1997) to history encoding in order to learn the probability distribution of the user intents.

$$\mathbf{v}_{\text{cur}} = \text{BLSTM}(\mathbf{x}, W_{\text{his}} \cdot \mathbf{v}_{\text{his}}), \quad (1)$$

$$\mathbf{o} = \texttt{sigmoid}(W_{\text{SLU}} \cdot \mathbf{v}_{\text{cur}}), \quad (2)$$

where W_{his} is a weight matrix and $\mathbf{v}_{\text{his}}$ is the history summary vector, $\mathbf{v}_{\text{cur}}$ is the context-aware vector of the current utterance encoded by the BLSTM, and $\mathbf{o}$ is the intent distribution. Note that this is a multi-label and multi-class classification, so the `sigmoid` function is employed for modeling the distribution after a dense layer. The user intent labels are decided based on whether the value is higher than a threshold tuned by the development set.

Considering that speaker role information is shown to be useful for better understanding in complex dialogues (Chi et al., 2017), we follow the prior work for utilizing the contexts from two roles to learn history summary representations, $\mathbf{v}_{\text{his}}$ in (1), in order to leverage the role-specific contextual information. Each role-dependent recurrent unit $\text{BLSTM}_{\text{role}_i}$ receives corresponding inputs, x_{t,role_i}, which includes multiple utterances u_i ($i = [1, ..., t-1]$) preceding the current utterance u_t from the specific role, role_i, and have been processed by an encoder model.

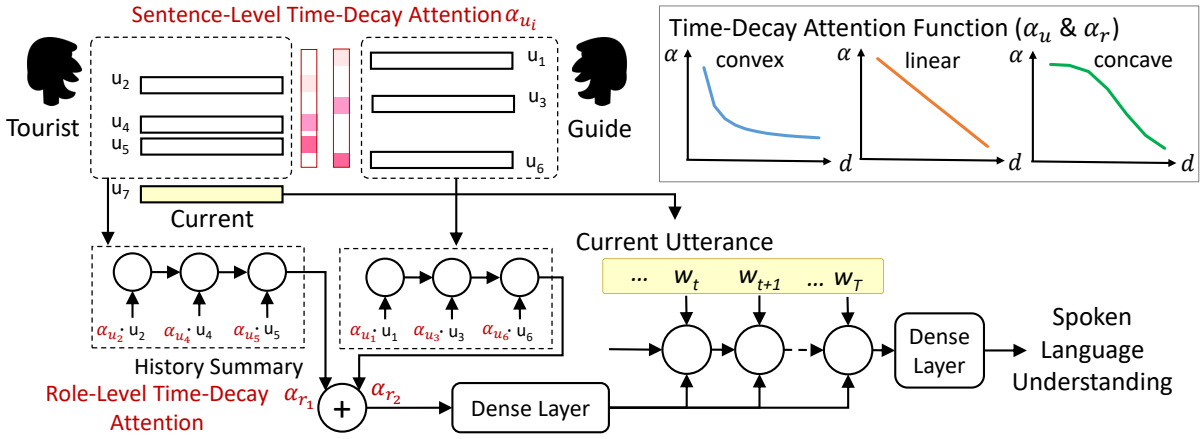


Figure 2: Illustration of the proposed time-aware attention contextual model with three types of time-decay attention functions.

$$\mathbf{v}_{\text{his}} = \sum_{\text{role}} \mathbf{v}_{\text{his,role}} = \sum_{\text{role}} \text{BLSTM}_{\text{role}}(x_{t,\text{role}}), \quad (3)$$

where $x_{t,\text{role}}$ are vectors after one-hot encoding that represent the annotated intent and the attribute features. Note that this model requires the ground truth annotations for history utterances for training and testing. Therefore, each role-based contextual module focuses on modeling role-dependent goals and speaking style, and $\mathbf{v}_{\text{cur}}$ from (1) would contain role-based contextual information.

2.2 Neural Attention Mechanism

One of the earliest work with a memory component applied to language processing is memory networks (Weston et al., 2015; Sukhbaatar et al., 2015), which encodes mentioned facts into vectors and stores them in the memory for question answering. The idea is to encode important knowledge and store it into memory for future usage with attention mechanisms. Attention mechanisms allow neural network models to selectively pay attention to specific parts. There are also various tasks showing the effectiveness of attention mechanisms (Xiong et al., 2016; Chen et al., 2016c). Recent work showed that two attention types (content-aware and time-aware) and two attention levels (sentence-level and role-level) significantly improve the understanding performance for complex dialogues. This paper focuses on expanding the time-aware attention based on the investigation of different time-decay functions, and further learning an universal time-decay function automatically. For time-aware attention mechanisms, we apply it using two levels, sentence-level and role-level structures, and Section 3 details the design and analysis of time-aware attention.

For the sentence-level attention, before feeding into the contextual module, each history vector is weighted by its time-aware attention α_{u_j} for replacing (3):

$$\mathbf{v}^U_{\text{his}} = \sum_{\text{role}} \text{BLSTM}_{\text{role}}(x_{t,\text{role}}, \{\alpha_{u_j} \mid u_j \in \text{role}\}).$$

For the role-level attention, a dialogue is disassembled from a different perspective on which speaker's information is more important (Chi et al., 2017). The role-level attention is to decide how much to address on different speaker roles' contexts ($\mathbf{v}_{\text{his,role}}$) in order to better understand the current utterance. The importance of a speaker given the contexts can be approximated to the maximum attention value among the speaker's utterances, $\alpha_{\text{role}} = \max \alpha_{u_j}$, where u_j includes all contextual utterances from the speaker. With the role-level attention, the sentence-level history from (3) can be rewritten into

$$\mathbf{v}^R_{\text{his}} = \sum_{\text{role}} \alpha_{\text{role}} \cdot \mathbf{v}_{\text{his,role}} \quad (4)$$

for combining role-dependent history vectors with their attention weights.

2.3 End-to-End Training

The objective is to optimize SLU performance, predicting multiple speech acts and attributes described in Section 2.1. In the proposed model,

all encoders, prediction models, and attention weights can be automatically learned in an end-to-end manner.

3 Time-Decay Attention Learning

The decaying function curves can be easily separated into three types: *convex*, *linear*, and *concave*, illustrated in the top-right part of Figure 2, and each type of time-decay functions expresses a time-aware perspective given dialogue contexts. Note that all attention weights will be normalized such that their summation is equal to 1.

3.1 Convex Time-Decay Attention

A *convex* curve also known as "concave upward", in a simple 2D Cartesian coordinate system (x, y), a convex curve $f(x)$ means when x goes greater, the slope $f'(x)$ is increasing. Intuitively, recent utterances contain more salient information, and the salience decreases very quickly when the distance increases; therefore we introduce the time-aware attention mechanism that computes attention weights according to the time of utterance occurrence explicitly. We first define the time difference between the current utterance and the preceding sentence u_i as $d(u_i)$, and then simply use its reciprocal to formulate a convex time-decay function:

$$\alpha_{u_i}^{\text{conv}} = \frac{1}{a \cdot d(u_i)^b}, \tag{5}$$

where a and b are scalar parameters.

The increasing slopes of the decay-curve assert that importance of utterances should be attenuated rapidly, and the importance of a earlier history sentence would be considerably compressed. Note that Chen et al. used a fixed convex time-decay function ($a = 1, b = 1$) (Chen et al., 2017).

3.2 Linear Time-Decay Attention

A *linearly* decaying time-aware attention function should also be taken into consideration. In a simple 2D Cartesian coordinate system (x, y), the slopes of a linear function remain consistent when x changes. That is, the importance of preceding utterances linearly declines as the distance between the previous utterance and the target utterance becomes larger.

$$\alpha_{u_i}^{\text{lin}} = \max(e \cdot d(u_i) + f, 0), \tag{6}$$

where e and f are the slope and the α-intercept of the linear function. Note that when the distance $d(u_i)$ is larger than $-\frac{f}{e}$, we assign the attention value as 0.

3.3 Concave Time-Decay Attention

A *concave* curve also called "concave downward", in contrast to convex curves, in a simple 2D Cartesian coordinate system (x, y), a concave curve $f(x)$ means that the slope $f'(x)$ is decreasing when x goes greater. Intuitively, the attention weight decreases relatively slow when the distance increases. To implement this idea, we design a $Butterworth\ filter$-like low-distance pass filter (Butterworth, 1930) that is similar to the concave time-decay function in the beginning of the curve.

$$\alpha_{u_i}^{\text{conc}} = \frac{1}{1 + (\frac{d(u_i)}{D_0})^n}, \tag{7}$$

where D_0 is the cut-off distance and n is the order of filter. The decreasing slopes of the decay-curve assert that the importance of utterances should weaken gradually, and the importance of a earlier history sentence would still be considerably compressed. Moreover, it is more likely to preserve the information in the multiple recent utterances instead of focusing only on the most recent one.

3.4 Universal Time-Decay Attention

As mentioned previously, there are three types of decaying curves: convex, linear, concave, each type represents a different perspective on dialogue contexts and models different contextual patterns. However, because the contextual patterns may be diverse, a single type of function could not fit the complex behavior well. Hence, we propose a flexible and universal time-decay attention function by composing three types of attentional curves:

$$\begin{aligned}\alpha_{u_i}^{\text{univ}} &= w_1 \cdot \alpha_{u_i}^{\text{conv}} + w_2 \cdot \alpha_{u_i}^{\text{lin}} + w_3 \cdot \alpha_{u_i}^{\text{conc}} \\ &= \frac{w_1}{a \cdot d(u_i)^b} + w_2(e \cdot d(u_i) + f) \\ &\quad + \frac{w_3}{1 + (\frac{d(u_i)}{D_0})^n},\end{aligned} \tag{8}$$

where w_i are the weights of time-decay attention functions. Because the framework can be trained in an end-to-end manner, all parameters (w_i, a, b, e, f, D_0, n) can be automatically learned to construct a flexible time-decay function. With the combination of different curves and the adjustable weights, the proposed universal time-decay attention function expresses the flexibility of not being

	LU Model		Sentence-Level Attention				Role-Level Attention			
			Conv.	Lin.	Conc.	Univ.	Conv.	Lin.	Conc.	Univ.
(a)	*DSTC4-Best*		61.60							
(b)	Naïve LU		70.18							
(c)	No Attention Cxt.		74.52							
(d)	Content-Aware Cxt.		73.69				74.28			
(e)	Time-Aware	Hand	75.95†	74.12	74.26	76.41†	76.73†	76.11†	76.01†	76.68†
(f)		E2E	76.04†	74.25	74.32	**76.67**†	76.69†	76.26†	76.08†	**76.75**†
(g)	Content+Time	Hand	74.71†	73.40	73.28	75.48†	76.70†	76.24†	76.03†	76.61†
(h)		E2E	74.94†	73.79	73.47	**75.83**†	76.51†	75.76†	76.22†	**76.74**†

Table 1: The understanding performance reported on F-measure in DSTC4, where the context length is 7 for each speaker (%). † indicates the significant improvement compared to all baseline methods. Hand: hand-crafted; E2E: end-to-end trainable.

strictly decaying; that is, the model can automatically learn a properly oscillating curve in order to model the diverse and complex contextual patterns using the attention mechanism.

4 Experiments

To evaluate the proposed model, we conduct the language understanding experiments on human-human conversational data.

4.1 Setup

The experiments are conducted using the DSTC4 dataset, which consist of 35 dialogue sessions on touristic information for Singapore collected from Skype calls between 3 tour guides and 35 tourists, these 35 dialogs sum up to 31,034 utterances and 273,580 words (Kim et al., 2016). All recorded dialogues with the total length of 21 hours have been manually transcribed and annotated with speech acts and semantic labels at each turn level. The speaker information (guide and tourist) is also provided. Unlike previous DSTC series collected human-computer dialogues, human-human dialogues contain rich and complex human behaviors and bring much difficulty to all the tasks. Given the complex dialogue patterns and longer contexts, DSTC4 is a suitable benchmark dataset for evaluation. We randomly selected 28 dialogues as the training set, 5 dialogues as the testing set, and 2 dialogues as the validation set.

We choose the mini-batch *Adam* as the optimizer with the batch size of 256 examples. The size of each hidden recurrent layer is 128. We use pre-trained 200-dimensional word embeddings $GloVe$ (Pennington et al., 2014). We only apply 30 training epochs without any early stop approach. We focus on predicting multiple labels including intents and attributes, so the evaluation metric is an average F1 score for balancing recall and precision in each utterance. The experiments are shown in Table 1, where we report the average results over five runs. We include the best understanding performance (row (a)) from the participants of DSTC4 in IWSDS 2016 for reference (Kim et al., 2016). The one-tailed t-test is performed to validate the significance of improvement, and the numbers with markers indicate the significant improvement with $p < 0.05$.

4.2 Effectiveness of Time-Decay Attention

To evaluate the proposed time-decay attention, we compare the performance with the naïve LU model without any contextual information (row (b)), the contextual model without any attention mechanism (row (c)), and the one using the content-aware attention mechanism (row (d)), where the attention can be learned at sentence and role levels. The row (a) is the performance reported in the DSTC challenge[2]. It is intuitive that the model without considering contexts (row (b)) performs much worse than the contextual ones for dialogue modeling. The proposed time-aware results are shown in the rows (e)-(h), where the rows (e)-(f) use only the time-aware attention while the rows (g)-(h) model both content-aware and time-aware attention mechanisms together. It is obvious that almost all time-aware results are better than three baselines.

In order to investigate the performance of various time-decay attention functions, for each curve we apply two settings: 1) **Hand**: hand-crafted

[2]This experiment is not performed on the same setup as this paper, and the shown number is estimated for reference.

hyper-parameters (rows (e) and (g)) and 2) **E2E**: end-to-end training for parameters (rows (f) and (h)). In the hand-crafted setting, the hyper-parameters $a = 1, b = 1, e = -0.125, f = 1, D_0 = 5, n = 3$ are adopted[3]. Table 1 shows that among three types of the sentence-level time-decay attention, only the convex time-decay attention significantly outperforms the baselines, indicating that an unsuitable time-decay attention function is barely useful. For both settings, the convex functions perform best among the three types of time-decay functions. Also, the end-to-end trainable setting results in better performance for most cases.

For our proposed universal time-decay attention mechanism, the same settings are conducted: 1) composing fixed versions for three types of time-decay functions weighted by learned parameters w_i and 2) fully trainable parameters for all time-decay functions. These two settings provide different levels of flexibility in fitting dialogue contextual attention, and the experimental results show that two settings both outperform all other time-decay attention functions.

For sentence-level attention, the end-to-end trainable universal time-decay attention achieves best performance (rows (f) and (h)), where the flexible time-aware attention (rows (f) and (h)) obtains 2.9% relative improvement compared to the model without the attention mechanism (row (c)) and the model using content-aware attention only (row (d)). For role-level attention, all types of time-decay functions significantly improve the results. The probably reason may be that modeling temporal importance for each sentence is more difficult and less accurate, and speaker roles in the dialogues provide informative cues for the model to connect the temporal importance from the same speakers together; therefore, the conversational patterns can be considered to additionally improve the understanding results. The further analysis is discussed in Section 4.3. Similarly, the best results are also from the end-to-end trainable universal time-decay function.

The significant improvement achieved by the universal functions indicates that our model can effectively learn a suitable attention function through this flexible setting and derive a proper curve to fit the temporal tendency to help the model preserve the essence and drop unimportant parts in the dialogue contexts. To further investigate what the universal time-decay attention learns, we inspect the learned weights w_i and find that the convex attention function almost dominates the whole function. In other words, our model automatically learns that the convex time-decay attention is more suitable for modeling contexts from the dialogue data than the other two types. Therefore, we can conclude that in complex dialogues, the recent utterances contain majority of salient information for spoken language understanding, where the attention decay trend follows a convex curve.

We analyze the content-aware attention impact by comparing the results between time-aware only (rows (e)-(f)) and content and time-aware jointly (rows (g)-(h)). The content-aware attention (row (d)) fails to focus on the important contexts for improving understanding performance in the complex dialogues and even performs slightly worse than the contextual model without attention (row (c)). Without a delicately-designed attention mechanism, it is not guaranteed that incorporating an additional content-aware attention would bring better performance and the experimental results show that a simple and coarse content-aware attention barely provides any usable information given the complex dialogues. Therefore, we focus on whether our time-aware attention mechanisms can compensate the poor attention learned from the content-aware model. In other words, we are not going to verify whether our time-aware attention mechanisms could collaborate with the content-aware attention mechanism, instead, we focus on examining how much our proposed time-aware attention could mitigate the detriment of the content-aware attention. By comparing the results between time-aware only (rows (e)-(f)) and content and time-aware jointly (rows (g)-(h)), we find that our universal time-decay attention keeps the improvement without too much performance drop by involving the learned temporal attention. Namely, our proposed attention mechanism can capture temporal information precisely, and it therefore can counteract the harmful impact of inaccurate content-aware attention.

4.3 Effectiveness of Role-Level Attention

For role-level attention, Table 1 shows that all results with various time-decay attention mecha-

[3]The chosen parameters are based on the domain knowledge about dialogue properties.

LU Model	Context Length 3	5	7
No Attention Contextual	74.75	74.69 (-)	74.52 (-)
Content-Aware Contextual	74.04	73.90 (-)	73.69 (-)
Time-Aware (Hand)	76.05	76.34 (+)	76.41 (+)
Time-Aware (E2E)	76.26	76.43 (+)	76.67 (+)
Content+Time (Hand)	75.16	75.27 (+)	75.48 (+)
Content+Time (E2E)	75.82	75.92 (+)	75.83 (-)

Table 2: The sentence-level performance reported on F1 of the proposed universal time-decay attention under different context length settings (%). The symbols '+' and '-' indicate the performance trends.

Parameter	Time-Aware (E2E Trainable) Sentence	Role
w_1	0.758	1.078
w_2	0.544	-0.378
w_3	-0.302	0.300
a	0.888	0.841
b	0.969	1.084
e	-0.320	-0.129
f	0.640	0.993
D_0	4.873	4.980
n	2.977	2.755

Table 3: The converged values of end-to-end trainable parameters from the proposed universal time-decay attention models. The values are averaged over five runs.

nisms are better than the one with only content-aware attention (row (d)). However, linear and concave time-decay functions do not provide additional improvement when we model the attention at the sentence level. The probable reason may be that it is difficult to model attention for individual sentences given the unsuitable time-decay functions. That is, if designs of attention functions are unsuitable for dialogue contexts, the encoded sentence embeddings would be weighted by improper attention values. On the other hand, for role-level attention, each speaker role is assigned an attention value to represent their importance in the conversational interactions. Previous work (Chi et al., 2017; Chen et al., 2017) also demonstrated the effectiveness of considering speaker interactions for better understanding performance. By introducing role-level attention, the sentence-level attentional weights can be smoothed to avoid inappropriate values. Surprisingly, even though learning sentence-level temporal attention is difficult, our proposed universal time-decay attention can achieve similar performance for sentence-level and role-level attention (76.67% and 76.75% from the row (f)), further demonstrating the strong adaptability of fitting diverse dialogue contexts and the capability of capturing salient information.

4.4 Robustness to Context Lengths

It is intuitive that longer context brings richer information; however, it may obstruct the attention learning and result in poor performance because more information should be modeled and accurate estimation is not trivial. Because when modeling dialogues, we have no idea about how many contexts are enough for better understanding, the robustness to varying context lengths is important for the contextual model design. Here, we compare the results using different context lengths (3, 5, 7) for detailed analysis in Table 2, where the number is for each speaker. The models without attention and the content-aware models become slightly worse with increasing context lengths. However, our proposed universal time-decay attention model mostly achieves better performance when including longer contexts, demonstrating not only the flexibility of adapting diverse contextual patterns but also the robustness to varying context lengths.

4.5 Universal Time-Decay Attention Analysis

This paper proposes a flexible time-decay attention mechanism by composing three types of time-aware attention functions in different decaying tendencies, where each decaying curves reflect a specific perspectives on distribution over salient information in dialogue contexts. The proposed universal time-decay attention shows great capability of modeling diverse dialogue patterns in the experiments and therefore proves that our proposed method is a general design of time-decay attention. In our design, we endow the attention function with flexibility by employing many trainable parameters and hence it can automatically learn a properly decaying curve for fitting the dialogue contexts better.

To further analyze the combination of different time-decay attention functions, we inspect the converged values of the trainable parameters from the proposed universal time-decay attention models in Table 3. Under the end-to-end trainable setting, the initialization of the trainable parameters are the same as the hand-crafted ones ($w_i = 1, a = 1, b = 1, e = -0.125, f = 1, D_0 = 5, n = 3$). In the experiments, the models automatically figure out that convex time-decay attention function should have a higher weight than others for both

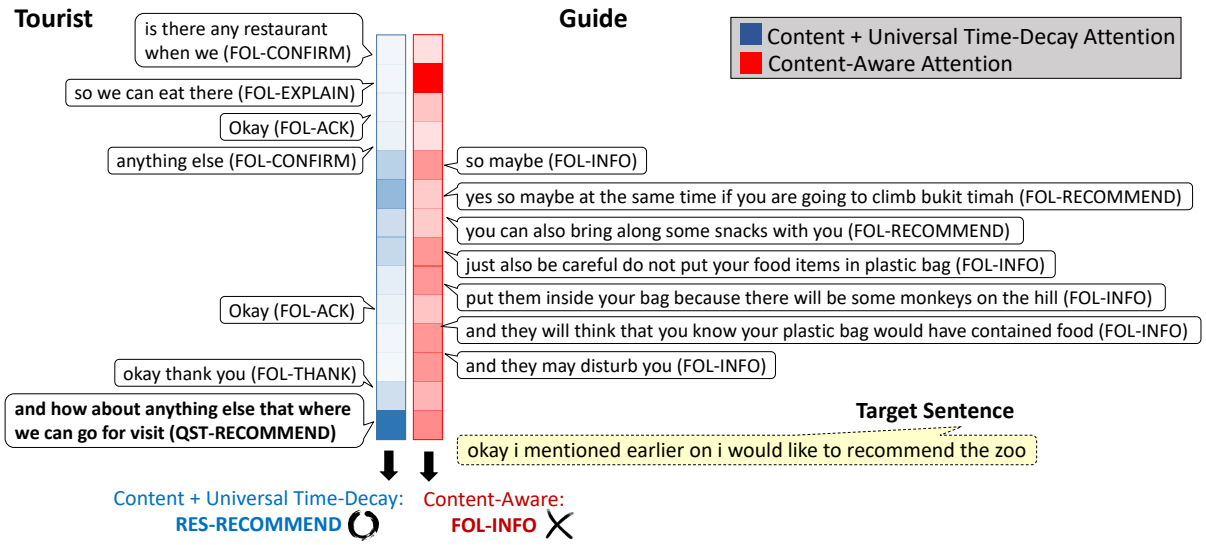


Figure 3: The visualization of the attention weights enhanced by the proposed time-decay function compared with the weights learned by the content-aware attention model.

sentence-level or role-level models ($w_1 > w_2$ and $w_1 > w_3$). Namely, in dialogue contexts, the recent utterances contain most information related to the current utterance, which is aligned with our intuition.

4.6 Qualitative Analysis

From the above experiments, the proposed time-decay attention mechanisms significantly improve the performance on both sentence and role levels. To further understand how the time-decay attention changes the content-aware attention, we dig deeper into the learned attentional values for sentences and illustrate the visualization in Figure 3. The figure shows a partial dialogue between the tourist (left) and the guide (right), where the color shades indicate the learned attention intensities of sentences. It can be found that the learned content-aware attention (red; row (c)) focuses on the incorrect sentence ("*so we can eat there*" (FOL-EXPLAIN)) and hence predicts the wrong label, FOL-INFO. The reason may be that with a coarse and simple design of content-aware attention mechanism, the attention function may not provide additional benefit for improvement. By additionally leveraging our proposed universal time-decay attention methods, the result (blue; row (g)) shows that the adjusted attention pays the highest attention on the most recent utterance and thereby predicts the correct intent, RES-RECOMMEND. It can be found that our proposed time-decay attention can effectively turn the attention to the correct contexts in order to correctly predict the dialogue act and attribute. Therefore, the proposed attention mechanisms are demonstrated to be effective for improving understanding performance in such complex human-human conversations.

5 Conclusion

This paper designs and investigates various time-decay attention functions based on an end-to-end contextual language understanding model, where different perspectives on dialogue contexts are analyzed and a flexible and universal time-decay attention mechanism is proposed. The experiments on a benchmark human-human dialogue dataset show that the understanding performance can be boosted by simply introducing the proposed time-decay attention mechanisms for guiding the model to focus on the salient contexts following a convex curve. Moreover, the proposed universal time-decay mechanisms are easily extensible to multi-party conversations and showing the potential of leveraging temporal information in NLP tasks of dialogues.

Acknowledgements

We would like to thank reviewers for their insightful comments on the paper. The authors are supported by the Ministry of Science and Technology of Taiwan, Google Research, Microsoft Research, and MediaTek Inc..

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Anshuman Bhargava, Asli Celikyilmaz, Dilek Hakkani-Tur, and Ruhi Sarikaya. 2013. Easy contextual intent prediction and slot detection. In *Proceedings of ICASSP*. pages 8337–8341.

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. In *Proceedings of ICLR*.

Stephen Butterworth. 1930. On the theory of filter amplifiers. *Wireless Engineer* 7(6):536–541.

Po-Chun Chen, Ta-Chung Chi, Shang-Yu Su, and Yun-Nung Chen. 2017. Dynamic time-aware attention to speaker roles and contexts for spoken language understanding. In *Proceedings of ASRU*. pages 554–560.

Yun-Nung Chen, Dilek Hakanni-Tür, Gokhan Tur, Asli Celikyilmaz, Jianfeng Guo, and Li Deng. 2016a. Syntax or semantics? knowledge-guided joint semantic frame parsing. In *Proceedings of 2016 IEEE Spoken Language Technology Workshop*. pages 348–355.

Yun-Nung Chen, Dilek Hakkani-Tur, Gokhan Tur, Asli Celikyilmaz, Jianfeng Gao, and Li Deng. 2016b. Knowledge as a teacher: Knowledge-guided structural attention networks. *arXiv preprint arXiv:1609.03286* .

Yun-Nung Chen, Dilek Hakkani-Tür, Gökhan Tür, Jianfeng Gao, and Li Deng. 2016c. End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding. In *Proceedings of INTERSPEECH*. pages 3245–3249.

Yun-Nung Chen, Ming Sun, Alexander I. Rudnicky, and Anatole Gershman. 2015. Leveraging behavioral patterns of mobile applications for personalized spoken language understanding. In *Proceedings of ICMI*. pages 83–86.

Ta-Chung Chi, Po-Chun Chen, Shang-Yu Su, and Yun-Nung Chen. 2017. Speaker role contextual modeling for language understanding and dialogue policy learning. In *Proceedings of IJCNLP*. pages 163–168.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of ACL*. pages 484–495.

Dilek Hakkani-Tür, Gökhan Tür, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional RNN-LSTM. In *Proceedings of INTERSPEECH*. pages 715–719.

Seokhwan Kim, Luis Fernando DHaro, Rafael E Banchs, Jason D Williams, and Matthew Henderson. 2016. The fourth dialog state tracking challenge. In *Proceedings of IWSDS*.

Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz. 2017. End-to-end task-completion neural dialogue systems. In *Proceedings of The 8th International Joint Conference on Natural Language Processing*. pages 733–743.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*. volume 14, pages 1532–1543.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45(11):2673–2681.

Yangyang Shi, Kaisheng Yao, Hu Chen, Yi-Cheng Pan, Mei-Yuh Hwang, and Baolin Peng. 2015. Contextual spoken language understanding using recurrent neural networks. In *Proceedings of ICASSP*. pages 5271–5275.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Proceedings of NIPS*. pages 2431–2439.

Ming Sun, Yun-Nung Chen, and Alexander I. Rudnicky. 2016. An intelligent assistant for high-level task understanding. In *Proceedings of IUI*. pages 169–174.

Gokhan Tur and Renato De Mori. 2011. *Spoken language understanding: Systems for extracting semantic information from speech*. John Wiley & Sons.

Zhangyang Wang, Yingzhen Yang, Shiyu Chang, Qing Ling, and Thomas S Huang. 2016. Learning a deep l encoder for hashing. In *Proceedings of IJCAI*. pages 2174–2180.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of EACL*. pages 438–449.

Jason Weston, Sumit Chopra, and Antoine Bordesa. 2015. Memory networks. In *Proceedings of ICLR*.

Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. *arXiv preprint arXiv:1603.01417* .

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of ICML*. pages 2048–2057.

Puyang Xu and Ruhi Sarikaya. 2014. Contextual domain classification in spoken language understanding systems using recurrent neural network. In *Proceedings of ICASSP*. pages 136–140.

Rui Zhang, Honglak Lee, Lazaros Polymenakos, and Dragomir Radev. 2018. Addressee and response selection in multi-party conversations with speaker interaction rnns. In *Proceedings of AAAI*.

Towards understanding text factors in oral reading

Anastassia Loukina, Van Rynald T. Liceralde, Beata Beigman Klebanov
Educational Testing Service
Princeton, NJ, USA
{aloukina,vliceralde,bbeigmanklebanov}@ets.org

Abstract

Using a case study, we show that variation in oral reading rate across passages for professional narrators is consistent across readers and much of it can be explained using features of the texts being read. While text complexity is a poor predictor of the reading rate, a substantial share of variability can be explained by timing and story-based factors with performance reaching r=0.75 for unseen passages and narrator.

1 Introduction

Listening to and performing oral reading are activities that permeate daily life, from parents reading aloud to young children, through reading instruction in elementary school, to audiobook narrations increasingly chosen by adults as the form of book-reading that fits in a busy schedule. Oral reading is also used in assessment of language skills for children and language learners, and in professions such as teaching and news broadcasting.

Reading rate is a common metric used to control or evaluate oral reading. It is usually computed as a number of words read per minute, and is used in many applications. For example, research in second language acquisition has considered both optimal reading rates for listening materials aimed at English language learners and reading rates that ensure the highest comprehensibility of accented speech (Munro and Derwing, 1998). Speech rate is a standard feature in systems for automated scoring of second language proficiency (Higgins et al., 2011) including read aloud tasks (Zechner et al., 2012; Evanini et al., 2015). Reading rate is also one of the main measures used to assess the fluency of oral reading (Hasbrouck and Tindal, 2006).

The assumption underlying these uses is that reading rate is a property of the reader (or controlled by the reader). However, variation in reading rate across different passages for the same readers has also been reported (Foulke, 1968; Tauroza and Allison, 1990; Ardoin et al., 2005; Compton et al., 2004; Beigman Klebanov et al., 2017).

Improving the understanding of the properties of oral reading, such as reading rate, is thus an important theoretical goal. We also have a specific practical reason to study text-based variation in reading rate. We are developing an intervention for improving literacy that would encourage sustained reading by having the student read aloud multiple passages from an engaging novel-length book, taking turns with others. While it is technically easy to compute reading rate by timing the readers, if a reader's rate across different texts is not stable given his current reading skill, it is not clear that tracking the rate over time would yield a valid measurement of improvement in skill. However, if such variation is systematically dependent on the text being read, rather than a random or idiosyncratic fluctuation, we might be able to adjust the measurement to account for text effect.

In order to inform both the theoretical and the applied goals, we address the following research questions in this paper:

1. Is reading rate constant for a given reader across various texts?

2. If not, do different readers show similar patterns of variation across texts, or is variation idiosyncratic?

3. If variation exists and is systematic across readers, can we identify the properties of texts that impact reading rate?

In this paper, we study reading rates in two professional narrations of the same book-length text. By using professional narrations we are able to

Proceedings of NAACL-HLT 2018, pages 2143–2154
New Orleans, Louisiana, June 1 - 6, 2018.

eliminate other factors that might cause variation in reading rate, such as reader fatigue or disfluencies. While these would play a role in a practical application, we seek first to answer the research questions in a setup that allows focusing on the relationship between reading rate and the passage being read, controlling for other factors.

2 Related work

2.1 Passage effects in reading

Passage effects in reading have been addressed most directly in the context of assessment of reading. Since the intention is to measure the student's reading ability, any difference in performance that is not due to reading ability confounds the measurement. In particular, since comprehension complexity of a passage is known to impact reading comprehension, it seems reasonable to assume that it would also impact other aspects of reading skill, including oral reading fluency. In fact, this assumption underlies text selection for tests of oral reading fluency such as DIBELS (Good and Kaminski, 2002) that rely on readability to select comparable passages (Francis et al., 2008).

Yet research also suggests that controlling for readability does not entirely solve the problem of text-based variation in reading fluency. Ardoin et al. (2005) examined readability formulas for their ability to predict fluency and generally found only low-to-moderate correlations ($r<0.5$). Researchers also observed that fluency measurements for the same students varied across texts even for passages of comparable readability (Ardoin et al., 2005; Compton et al., 2004; Petscher and Kim, 2011; Francis et al., 2008).

Moreover, Francis et al. (2008) found that while actual fluency scores vary across different readability-controlled passages, the relative ranking of students is only minimally different when estimated using different passages, suggesting that variation in fluency has some consistency across readers; results to a similar effect were reported by Beigman Klebanov et al. (2017).

Oral reading fluency is commonly measured using words correct per minute – a combination of reading accuracy and reading rate. It is thus not clear whether the observations above pertain more to the accuracy aspect of oral reading (not considered in the current paper) or to reading rate, although Beigman Klebanov et al. (2017) noted that consistent variation across students was observed both for reading rate and for reading fluency.

To summarize, it appears that while readability could explain some of the variation in oral reading performance, there are also indicators that it is not sufficient on its own to effectively control for variation in oral reading performance caused by the properties of the passage being read.

2.2 Factors that affect duration of segments and pauses

2.2.1 Sentence-level timing

Since oral reading involves saying the text aloud, the durations of individual segments, words and phrases as well as location and duration of silent pauses are subject to constraints that have been extensively studied in literature on phonetic timing; see White (2014); Hirschberg (2002) for a review.

Thus it has been long known that different segments have different intrinsic durations which account for a lot of variation in segmental durations (Peterson and Lehiste, 1960; Klatt, 1976; van Santen, 1992): for example, high vowels tend to be shorter than low vowels. At the syllable level, in many languages vowels tend to be shorter when followed by a voiceless consonant than when followed by a voiced consonant (House and Fairbanks, 1953; Crystal and House, 1988) while consonants within a consonant cluster tend to be shorter than single consonants (Klatt, 1976).

Further constraints are at play at word, phrase and sentence level. White (2014) summarizes these as "domain-head" and "domain-edge" lengthening effects. "Domain-head" lengthening refers to lengthening of salient elements such as syllables bearing lexical stress, words in prominent positions (Peterson and Lehiste, 1960; Crystal and House, 1988; van Santen, 1992). "Domain-edge" effects include lengthening of segments in word-initial position or sentence-final lengthening (Turk and Shattuck-Hufnagel, 2000, 2007).

Finally, these domain-head and domain-edge lengthening effects do not apply uniformly: some segments and some positions are more resistant to lengthening than others (Peterson and Lehiste, 1960; Klatt, 1976; van Santen, 1992; White, 2014). The magnitude of lengthening also depends on the number of elements within each domain: in monosyllabic words, the stressed syllable receives all of the prosodic lengthening, but in disyllabic and trisyllabic words, some of the

lengthening spreads to the unstressed syllables and lengthening of the stressed syllable is attenuated (Turk and Shattuck-Hufnagel, 2000; White and Turk, 2010).

In addition to segmental lengthening, phrase and sentence boundaries are often associated with some amount of pause. The location and duration of sentence-internal pauses depends both on syntactic structure and the number of syllables in each adjacent unit: sentence-internal pauses associated with punctuation or major syntactic boundaries tend to be longer than other sentence-internal pauses, with sentence-final pauses being the longest (Pfitzinger and Reichel, 2006; Burrows et al., 2005; Bailly and Gouvernayre, 2012).

2.2.2 Content and duration

Beyond domain-head and domain-edge effects, duration of segments and pauses is also affected by other aspects of text content. Frequent words tend to have shorter duration than phonologically similar less frequent words. Words that are more predictable in a given context tend to be shorter than words with higher information load and repeated words are pronounced shorter than the first mention (see Zhao and Jurafsky (2009); Bell et al. (2009) for reviews). Note that these effects persist after one controls for "domain-head" effects described in the previous section (Bell et al., 2009).

Further factors come into play in the context of story-telling where the speaker is either reading or narrating a well-rehearsed story. Montaño and Alías (2017) review approaches used to characterize story-telling speech.

Several studies observed that the duration of pauses between sentences and paragraphs in a longer story is not uniform. In their analysis of pausing in book reading, Bailly and Gouvernayre (2012) reported that pauses between paragraphs were longer than pauses between sentences. They also found that the thematic relationships between sentences affect breathing patterns although these were not immediately related to pause duration.

Reading rate has also been shown to depend on the emotional state of the speaker, whether genuine or performed as part of a dramatic reading: for example, actors tend to speak slower when expressing anger, fear or sorrow (Williams and Stevens (1972), see Scherer (2003) for a comprehensive review). Doukhan et al. (2011) analyzed pause distribution in a corpus of tales and reported "speakers' expressive reinterpretation of sentence syntactic structure" which they attributed to expressiveness of the reader.

There is also evidence that prosody may be affected by the narrative structure. Theune et al. (2006) observed in an informal analysis that Dutch actors narrating fairy-tales reduced their speech tempo when approaching the story climax. They also noticed an increase in duration in some words that indicated extreme value of a property. Doukhan et al. (2011) analyzed prosody in a corpus of French tales using Propp's morphology of Folktale (Propp, 1968). They found that narrative structures had a significant effect on various prosodic properties. For duration, epilogues were associated with lower articulation rate (syllables/min without pauses) while refrains had the lowest pausing time percentage. Finally, several studies found that impersonation by narrator of different characters leads to clear differences in pitch, intensity and spectral quality (Doukhan et al., 2011; Wang et al., 2006).

In short, previous research suggests that multiple factors may affect phone and pause duration in a reading of a story: from the phonetic properties of individual segments to where the passages falls within the narrative structure. However, most of these studies considered durations of individual segments, words, or pauses. It is not clear which of these effects will still persist when durations are averaged over a longer text as is the case for reading rate computation. In fact, studies in phonetics talk about "emergent speech rate" that can be relatively consistent over long stretches of speech (White, 2014). Furthermore, pause duration is likely to have a substantial effect on the reading rate (Kendall, 2013) yet previous research on pausing in story-telling suggests that this can be highly idiosyncratic.

3 Data

3.1 Text

We use the "Harry Potter and the Sorcerer's Stone" by J.K. Rowling (Rowling, 2015) as the case study for this paper. The book consists of 79,508 words spread across 17 chapters. We divided the text into 313 non-overlapping passages of about 250 words each (mean = 249 words; range: 190-309).[1] Boundaries of passages were set to be the starts

[1] This is roughly the intended length of a reading turn in the turn-taking reading intervention described in the Introduction.

and ends of paragraphs, where the end of a passage consists of a paragraph whose addition brings the passage closer to 250 words than without adding the paragraph. When generating passages, we took into account chapter boundaries so that no passage spanned two chapters: the word-count for passage generation was always re-set from the beginning of the chapter and any short fragments left at the end of a chapter were not included in the analysis. We randomly assigned 156 passages to the training set and 157 passages to the test set.

3.2 Audio

We used data from two narrators. The first dataset, hereafter referred to as JD, comes from a narration by the actor Jim Dale published as an audio-book (Rowling and Dale, 2016). The book is released as 17 .mp3 files with one file per chapter.

The second dataset comes from the audio-book with a female narrator, provided to us by Learning Ally.[2] We will refer to it as LA. These recordings are created by volunteers and are made available on subscription-basis to students diagnosed with disabilities that impact their ability to read print-based materials. Learning Ally recordings are subject to quality control similar to that of commercial audio-books.[3]

3.3 Calculating reading rate

We used forced alignment to automatically align the audio for JD narration for each chapter with the book text and establish the passage boundaries. We used the Kaldi toolkit (Povey et al., 2011) and publicly available acoustic models trained on the LibriSpeech corpus (Panayotov et al., 2015). The forced alignment was spot-checked manually for accuracy and found to be very accurate. The LA audio was already aligned with the book text.

The LA recordings were split across multiple audio files. To avoid any artifacts of the recording process, we only used the passages where the whole audio was in the same file. Out of the original 314 passages, 270 passages (86%) satisfied this condition, of these 134 in the training set and 136 in the test set. We used these matching training and testing passages for both narrators, in order to facilitate comparisons.

[2] https://www.learningally.org/

[3] We cannot use another well-known commercially available narration, by Stephen Fry, since he narrates the British version of the book.

For both narrators we used the time stamps for the beginning of the first word in the passage and the end of the last word in the passage to compute the total duration of the passage, which was then divided by the number of words in the passage to yield the reading rate (words per minute, WPM).

4 RQ1: Is reading rate constant?

To answer our first question, we looked at the distribution of the reading rate across the passages in the training set.

The distribution of WPM for both narrators was close to normal. JD: mean = 164.01; SD = 12.66; min = 129.2; max = 197.7. LA: mean = 125.12; SD = 11.4; min = 86.8; max = 156.9. Based on discussions in the literature regarding syllables per second being a more stable measure of reading rate than WPM (Tauroza and Allison, 1990; Griffiths, 1991; Munro and Derwing, 1998), we calculated rate in syllables per second, and observed a similar pattern of variation (JD: mean = 3.52, SD = 0.30, LA: mean = 2.72, SD = 0.27). We also found that WPM and syllables per second were highly correlated, for each of the narrators ($r \geq 0.9$). We therefore continue with WPM, as this is the commonly used measure in the reading assessment context. The distribution of WPM for each of the narrators is shown in Figure **??**.

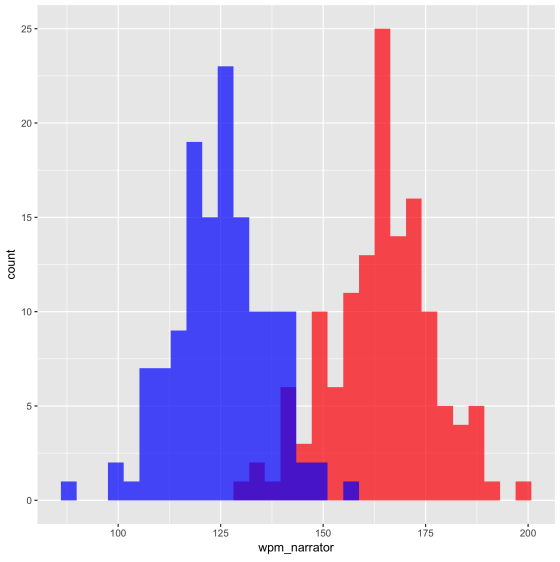


Figure 1: Distribution of WPM for LA (blue) and JD (red).

The answer to RQ1 is that while there is clearly a sense in which one narrator generally reads slower than the other, it is not the case that a narra-

tor keeps the same rate of reading across different passages.

5 RQ2: Is variation in reading rate correlated across the two narrators?

We next compared the reading rates of the two narrators on the 134 training set passages. We found them to be highly correlated: Pearson's r = .81. This suggests that a substantial share of the variation across passages is systematic rather than idiosyncratic. We therefore proceed to the next question – what factors can explain this variation?

6 RQ3: What textual factors explain variation in reading rate?

6.1 Method

We use a standard model building approach to answer RQ3. We used the train partition with JD's WPM (hereafter JD-train) to identify possible textual features as well as the best learner to combine these features. We then trained separate models on the training data for the two narrators. We evaluated the two models on: (a) different passages from the test partition as read by the same narrator; (b) same passages as used for model training but read by a different narrator; (c) different passages (test partition) read by a different narrator.

6.2 Baseline: text complexity

We used text complexity as our baseline, following the practice in the reading assessment community. While we do not expect either of the narrators to experience any reading comprehension difficulties, one might reasonably assume that a skilled narrator would slow down on fragments which are harder for the listener to comprehend.

We used TextEvaluator,[4] a state-of-the-art measure of comprehension complexity of a text (Napolitano et al., 2015; Sheehan et al., 2014, 2013; Nelson et al., 2012).[5] TextEvaluator extracts a range of linguistic features and uses them to compute a complexity score on the scale of 100–2000. TextEvaluator computes three complexity scores based on the models optimized for literary, informational and mixed texts. We used the literary metric. The average complexity score for passages in the training set was 613.1, with a large variation across passages: min=240, max=1,019, SD=154.75. In other words, the selected book offers its readers much variety in the configurations of textual features across its different passages.

6.3 Features

We used the passage text to extract 107 features that capture different factors that might affect durations in oral reading. These could be grouped into four categories.

6.3.1 Sentence-level timing factors

We hypothesized that the timing effects described in section 2.2.1 are likely to be the source of at least some variation in reading rates across the text. Due to the complexity of these effects, building an accurate model that would predict segmental durations based on the text is not a trivial task.

This problem has been extensively discussed in literature on modeling prosody for text-to-speech synthesis systems (TTS) which generally combined the insights from the phonetic studies with statistical learning in order to establish the optimal duration for each segment and pause in synthesized audio. Therefore rather than attempting to build our own model, we synthesized the audio for each passage using Apple's built-in TTS engine (OS X 10.11.6). We used the male Alex voice which in terms of overall quality and default speaking rate appeared closest to JD. According to Capes et al. (2017), linguistic features used for training this system include segment identity and segmental context, stress, part-of-speech context, prominence[6], sentence type and initial/final positional features for syllable, word, phrase and sentence;[7] in other words, features directly related to timing factors discussed in 2.2.1.

We used the generated audio to compute the WPM for each passage. The mean reading rate of TTS was close to that of JD: 157.1 vs. 164.0. There was variation across passages with WPM varying from 129.2 to 197.7 (SD = 9.13).

6.3.2 Lexical, syntactic, and discourse features

Next, we considered lexico-syntactic properties of the passages. Some of these (lexical fre-

[4] https://textevaluator.ets.org/

[5] TextEvaluator appears in the Nelson et al. (2012) benchmark as SourceRater.

[6] See for example (Hirschberg, 1993) for features used to establish prominence

[7] Note that Capes et al. (2017) describe a different engine from the one used in this study, however as noted in the paper it shares the front-end for linguistic feature extraction with other Mac OS TTS systems. The same features are also described in (Zen et al., 2009).

quency, emotion, arousal) may be associated with local changes in segment and pause durations (see 2.2.2). Many of the features are used as low-level features in readability estimation (Graesser et al., 2004; Sheehan et al., 2014), and are thus likely to capture facets of a reader's experience when reading the text.

These included: (1) Vocabulary features capturing presence as well as average score along some meaning dimension, such as concreteness, imageability, emotion, arousal, motion, academic register (Coltheart, 1981; Warriner et al., 2013; Coxhead, 2000); (2) Morphological features (e.g., count of nominalizations, count of syllables); (3) Distributional features such as average word frequency; (4) Syntactic features such as counts of different part-of-speech, as well as features based on specific constructions (relative clauses, preposed clauses etc.); (5) Discourse features that deal with paragraphing (e.g., word count of the longest paragraph, average paragraph length in sentences) and overall cohesion (e.g., average lexical overlap across adjacent sentences).

6.3.3 Story-related features

Considering previous work on prosody in storytelling, we also built features that relate to the overall story development. These included: (1) The number of occurrences of names of the main characters and other proper nouns important to the plot (*Harry*, *Hermione*, *Weasley*, *Dumbledore*, *Ollivander*, *Quidditch*), under the assumption that there might be systematic ways in which the narrators act out certain kinds of people (older vs younger, for example), as well as events that could indicate a fast-paced event, such as a commentated game of Quidditch; (2) The order in which the passage appears in the book (as a numeric continuous variable); (3) Plot arc as estimated by `syuzhet` package (Jockers, 2015). This package uses sentiment analysis to attempt to reveal the latent structure of the narrative. We used the default sentiment lexicon developed by the Nebraska Literary Lab supplied with the package.

6.3.4 Performance-related features

Finally we considered typographic features that provide clues to how the text should be performed when read aloud. These included exclamation marks (!), ellipses (...), words printed in all capitals and indications that a character stutters.

6.4 Learner

We used 3-fold cross-validation on JD-train to compare performance of 9 regressors available via `SKLL`[8] package including Random Forest, SVR and various regularized linear models. We used grid-search with 3-fold cross-validation within each fold to fine-tune the parameters for all learners. We found that Lasso regression achieved the highest average performance and therefore used it as the learner for subsequent evaluations.

6.5 Results

Table 1 shows the performance of all models on the four datasets in our study. Since we are interested in explaining variation across passages rather than predicting the actual reading rate of a given narrator for a given passage, we use Pearson's r as our evaluation metric, as it would capture the extent to which the predicted and the observed values deviate similarly from their respective means, and thus would not be affected by differences in absolute values between the two narrators.

Dataset	Baseline	M_{JD}	M_{LA}
JD-train	0.38	-	0.71
LA-train	0.40	0.80	-
JD-test	0.37	0.74	0.74
LA-test	0.45	0.75	0.80

Table 1: Performance (Pearson's r) of models trained using Lasso regression on JD-train (M_{JD}) or LA-train (M_{LA}). The models are evaluated on unseen data: different passages read by the same narrator, same passages read by a different narrator and different passages read by different narrator. The table also shows the correlations with baseline (text complexity score).

The correlations between the baseline (estimates of comprehension complexity) and WPM of the two narrators were r=0.37–0.45. We also note that the direction of the correlation was opposite to our expectation: more complex passages were in fact read faster.

Our models substantially outperformed the baseline with r increasing from 0.4 to 0.7–0.8. In other words, the final models explain much of the variability in reading rates. Furthermore, this level of performance holds for predicting variation in reading rate for a set of unseen passages

[8]We used v1.3 from https://github.com/EducationalTestingService/skll.

read by a different narrator (M_{LA} on JD-test and M_{JD} on LA-test), suggesting fairly strong generalization. Results also suggest that the prediction is somewhat easier for LA than for JD, in that evaluations on the former are in the 0.75–0.80 range, and for the latter – in the 0.71–0.74 range, no matter which narrator supplied the training data. This could be due to Jim Dale's narration being more theatrical/artistic, hence somewhat more idiosyncratic.

We used 3-fold cross-validation on JD-train and LA-train to further consider how much of the variation can be explained by different groups of features discussed in Section 6.3. The results are shown in Table 2. For both narrators, models based on all groups of features outperformed models based on individual groups of features, but all groups of features were effective in explaining at least some variance in reading rate across passages. Timing as modeled by TTS was the highest performing feature followed by lexico-syntactic features and story-based features.

Dataset	JD-train	LA-train
Baseline	0.37	0.39
All features	0.69	0.77
Sentence-level timing	0.63	0.75
Lexical/syntactic/discourse	0.55	0.65
Story-related	0.47	0.47
Performance-related	0.35	0.35

Table 2: Performance of the four groups of features described in Section 6.3 on the training passages for each of the narrators. The table shows average Pearson's r for 3-fold cross-validation on JD-train and LA-train. All models use Lasso regression.

To summarize, we found text complexity to be a poor predictor of passage-to-passage variability in reading rates of adult narrators. These findings are consistent with recent work in the oral reading fluency community which found variation in children's reading fluency across passages after controlling for grade level (see Section 2.1).

We found that textual factors that explain a substantial share of passage-to-passage variability in reading rates include sentence-level timing factors such as distribution of segments, stressed syllables, sentences, and pauses as well as features related to passage vocabulary and syntax, story and performance. Given the good generalization of our results to both a new narrator and to new passages, we believe they hold promise for explaining some of the unaccounted-for variation in reading rates observed in the oral reading fluency studies; more research is necessary to explore this direction.

7 Discussion

Out of 107 original features, 17 features had nonzero coefficients in M_{JD} and 14 in M_{LA}, with 6 features in the overlap: timing, ellipsis, number of verbs in past tense, preposition count, *Weasley*, and *Dumbledore*. Additional features selected in only one of the two models included various vocabulary features (such as age of acquisition, imageability for M_{JD}), syntax (average word count before main verb, contractions for M_{JD}), discourse (average lexical overlap in adjacent sentences), as well as story features (`syuzhet` and *Dudley* for M_{LA}, *Ollivander*, *quidditch* for M_{JD}).

Some of these features lend themselves to an easy explanation. Thus in our study, a strong predictor of narrator slowdown was occurrence of ellipsis (...), a mark of hesitation or thoughtfulness; these were not modeled as such by TTS.

Similarly, the positive weight alloted to the average lexical overlap in adjacent sentences is consistent with the expectation that repeat instances would be read faster.

Effective character features included *Ollivander* and *Dumbledore*; mentions of both of these indicate a slowdown in narration. One possible explanation is that passages with multiple mentions of these characters are likely to be those where they speak. Both of these characters are elderly; acting them out could yield a slower rate of speech.[9]

In other cases the interpretation of the feature was less straightforward. Thus the feature with the second highest coefficient after timing for both narrators was that which counted occurrences of members of the *Weasley* family. Why?

Figure 1 plots standardized reading rates of JD (blue), LA (orange), and TTS (black) as a function of the location in the book. It is clear from the plot that in addition to passage-by-passage variation there is a global pattern in narrator WPM: the narrators slow down over the first few chapters,

[9]Barbara Roseblatt, an audiobook narration coach, explicitly advises to slow down when reading the contribution of the old character in a coversation with a young one: https://www.youtube.com/watch?v=MVmywsM9-h4, 5:17. Jim Dale himself describes his image of Dumbledore as *hesitant, wheezy old man*: https://www.youtube.com/watch?v=whzhEIB9Qkg: 2:45.

then speed up, and slow down again in the last third. It is also apparent that the TTS curve is flatter, suggesting that some of the slowdown and especially the speedup are not due to sentence-level timing factors.

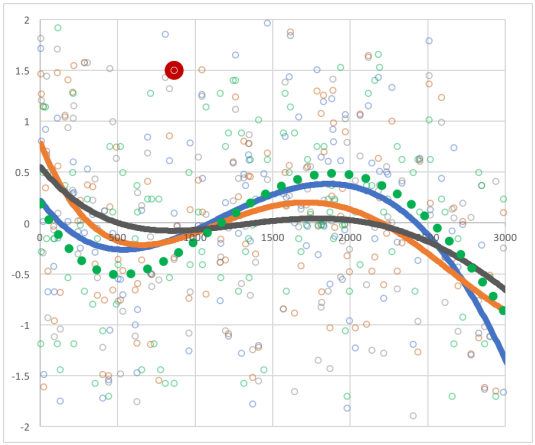


Figure 2: Standardized reading rates of JD (blue), LA (orange), TTS (black) as a function of the location of the passage in the novel. Trend lines are 4th degree polynomial approximations. The y-axis shows standardized readings rates (JD, LA, TTS). The large red dot and the green dotted line will be explained later in the text.

This book-level trend can help explain the strong performance of *Weasley*. This feature covers a number of characters that are prominent in the magic world as experienced by Harry (Ron, his brothers, sister, mother); they play no role at all in the first part of the story that is based in the Muggle world. The large red dot in Figure 1 indicates the first passage with a non-zero count for *Weasley*. This is very close to the onset of the speedup that is not captured by TTS. Apparently, the speedup coincides with an important plot transition (see Behr (2005) on plot transitions in Harry Potter), which is, in turn, indicated by a character mention pattern.

Next, we looked closely at one of the vocabulary features, specifically, imageability, calculated as the number of word tokens in a passage that belong to the MRC Imageability list (Coltheart, 1981). This feature has a partial correlation of -0.186 with JD controlling for TTS. In an attempt to identify the subset of the 1,194 words on the list that drive the correlation, we removed stopwords, all words that appeared in only one training passage, as well as short words (2-3 letters) and long words (7 letters or more). The partial correlation remained virtually the same (-0.178, $p < 0.05$).

These manipulations left us with 573 non-stop reasonably frequent 4-6 letter words. These words tend to name common everyday objects and properties (henceforth, **everyday** list), such as body parts (*knee, skin, neck, hair, nose, face, teeth*), colors (*blue, gray, green, white, black, orange, yellow*), family (*aunt, uncle, mother, father, sister, wife*), elements (*fire, water, wind, rain*) and materials (*silver, gold, stone, metal, glass, paper, silk*), eating (*cake, wine, dinner, hungry, eating*), common properties of objects (*warm, cold, broad, narrow, soft, hard, tall, short, long, clean, dirty*) and humans (*kind, evil, rude, polite, eager, proud, stupid, famous*), standard house interior (*chair, table, mirror, door, wall, room, clock*), feelings and emotions (*fear, hurt, hate, pain, anger, gloom, tired, panic, safe, boring, afraid, relief*), as well as numbers (*first, nine, half, dozen*), directional (*inside, back, front, behind, bottom*) and time expressions (*soon, hour, late, week, month, early, minute, moment*). These words "carry" the story, so to speak, in that on average about one third of all nonstop words in each passage belong to this list, albeit with substantial variation (min = 0.20; max = 0.49).

If the effect of the feature was simply due to higher incidence of the short high frequency everyday words, we would expect a *positive* correlation with the reading rate; in fact, the correlation is negative, suggesting that perhaps the feature is useful as an indirect indicator of something else, rather than for the phonological properties of the words on the list.

Variation across passages in the use of everyday words appears non-random. In particular, the first third of the book averages 41.4 matches per passages; the rest of the book averages 37.3. Given the above observations with *Weasley*, this is easy to explain in reference to the story line – the first part of the book mainly happens in Muggle ("normal") world, while the rest of the book happens in an alternative world of Hogwarts that is familiar enough (and so references to human feelings, bodies, and character still draw on the culturally familiar stock) yet different enough to drive a 10% average decline in the use of stock vocabulary, where special foods, special money units, the special game of quidditch, special subjects on the school's curriculum remain off the common list.

Since overlap with the everyday list has a negative correlation with reading rate, we flip the sign

of the standardized everyday token counts, and overlay the plot with that of reading rates; see the green dotted line in Figure 1. It is apparent that the global pattern of JD WPM is closely traced by the feature, especially in the middle area where JD is speeding up and then slowing down again.

The observed global slowdown, speedup, and slowdown appear to align with the traditional three-part narrative structure (exposition, complication, and resolution) (Chandler and Munday, 2016). One of our features (`syuzhet`) was based on the plot arc. While this feature was selected in one of the models, its partial correlation with JD after controlling for TTS was not significant. Our results suggest that important plot transitions can sometimes be captured indirectly by tracing patterns of word usage for other specific classes of words such as characters or everyday words and that for skilled readers these transitions can be associated with systematic changes in reading rate.

8 Conclusions

The main contributions of this paper are as follows. First, we demonstrate using a case study that variation in reading rate across passages for professional narrators is consistent across readers and much of it can be explained using features of the texts being read. These findings suggest that it is possible to estimate the expected variation in durations of oral reading across texts. In the assessment context, this has a potential of providing a powerful control mechanism for selecting comparable passages for parallel forms of a test of oral reading; in a context when one cannot adjust the materials (such as a reading intervention using a particular book), it might be possible to adjust the measurement of reading rate to compensate for the effects of the text on the observed performance.

Secondly, we found that timing is a very powerful feature, yet not a perfect predictor of reading rate (the two narrators are still highly correlated controlling for timing, partial r=0.64). This opens up a possibility for a sophisticated assessment of oral reading using both TTS and human benchmark to separate reading that adheres to basic timing constraints of English speech (which constitutes a demonstrably big part of fluent reading) from a more nuanced expressive reading that TTS is not currently doing, but good human readers are. Thus beyond assessment context, our findings can also inform work on text-to-speech synthesis for book-length texts.

Extending and validating the results reported here using additional types of text and separating the effect of text factors on the two components of reading rate, articulation rate and pausing, is an important next step to get a more comprehensive picture of the impact of text on oral reading.

Acknowledgments

We thank John Sabatini and Tenaha O'Reilly for many comments and suggestions that inspired this work; Binod Gyawali and Patrick Lange for preprocessing the book; Diane Napolitano for help with extracting TextEvaluator features; Learning Ally for allowing us to use their version of the narration; Nitin Madnani, Michael Flor, Keelan Evanini, and three anonymous NAACL reviewers for their comments and suggestions.

References

Scott P. Ardoin, Shannon M. Suldo, Joseph Witt, Seth Aldrich, and Erin McDonald. 2005. Accuracy of readability estimates' predictions of CBM performance. *School Psychology Quarterly* 20(1):1–22. https://doi.org/http://dx.doi.org/10.1016/j.jsp.2012.09.004.

Gérard Bailly and Cécilia Gouvernayre. 2012. Pauses and respiratory markers of the structure of book reading. In *Proceedings of Interspeech 2012, Portland, OR*. pages 67–70. https://isca-speech.org/archive/interspeech_2012/i12_2218.html.

Kate Behr. 2005. "Same-as-difference": Narrative transformations and intersecting cultures in Harry Potter. *Journal of Narrative Theory* 35(1):112–132. http://www.jstor.org/stable/30224622.

Beata Beigman Klebanov, Anastassia Loukina, John Sabatini, and Tenaha O'Reilly. 2017. Continuous fluency tracking and the challenges of varying text complexity. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*. Association for Computational Linguistics, Copenhagen, Denmark, pages 22–32. http://aclweb.org/anthology/W/W17/W17-5003.pdf.

Alan Bell, Jason M. Brenier, Michelle Gregory, Cynthia Girand, and Dan Jurafsky. 2009. Predictability effects on durations of content and function words in conversational English. *Journal of Memory and Language* 60(1):92–111. https://doi.org/10.1016/j.jml.2008.06.003.

Tina Burrows, Peter Jackson, Katherine Knill, and Dmitry Sityaev. 2005. Combining Models of Prosodic Phrasing and Pausing. In *Proceedings of Interspeech 2005, Lisbon, Portugal*. pages 1829–1832. https://isca-speech.org/archive/interspeech_2005/i05_1829.html.

Tim Capes, Paul Coles, Alistair Conkie, Ladan Golipour, Abie Hadjitarkhani, Qiong Hu, Nancy Huddleston, Melvyn Hunt, Jiangchuan Li, Matthias Neeracher, Kishore Prahallad, Tuomo Raitio, Ramya Rasipuram, Greg Townsend, Becci Williamson, David Winarsky, Zhizheng Wu, and Hepeng Zhang. 2017. Siri on-device deep learning-guided unit selection text-to-speech system. In *Proceedings of Interspeech-2017*. ISCA, ISCA, pages 4011–4015. https://doi.org/10.21437/Interspeech.2017-1798.

Daniel Chandler and Rod Munday. 2016. Narrative structure. In *A Dictionary of Media and Communications*, Oxford University Press. https://doi.org/10.1093/acref/9780191800986.001.0001.

Max Coltheart. 1981. The MRC psycholinguistic database. *The Quarterly Journal of Experimental Psychology Section A* 33(4):497–505. https://doi.org/10.1080/14640748108400805.

Donald L. Compton, Amanda C. Appleton, and Michelle K. Hosp. 2004. Exploring the Relationship Between Text-Leveling Systems and Reading Accuracy and Fluency in Second-Grade Students Who Are Average and Poor Decoders. *Learning Disabilities Research & Practice* 19(3):176–184. https://doi.org/10.1111/j.1540-5826.2004.00102.x.

Averil Coxhead. 2000. A new academic word list. *TESOL Quarterly* 34(2):213–238. https://doi.org/10.2307/3587951.

Thomas H. Crystal and Arthur S. House. 1988. Segmental durations in connected speech signals: Syllabic stress. *The Journal of the Acoustical Society of America* 83(4):1574–1585. https://doi.org/10.1121/1.395912.

David Doukhan, Albert Rilliard, Sophie Rosset, Martine Adda-Decker, and Christophe D'Alessandro. 2011. Prosodic analysis of a corpus of tales. In *Proceedings of Interspeech 2011*. pages 3129–3132. https://isca-speech.org/archive/interspeech_2011/i11_3129.html.

Keelan Evanini, Michael Heilman, Xinhao Wang, and Daniel Blanchard. 2015. Automated Scoring for the TOEFL Junior ® Comprehensive Writing and Speaking Test. *ETS Research Report Series* 2015(1):1–11. https://doi.org/10.1002/ets2.12052.

Emerson Foulke. 1968. Listening Comprehension as a Function of Word Rate. *Journal of Communication* 18(3):198–206. https://doi.org/10.1111/j.1460-2466.1968.tb00070.x.

David J. Francis, Kristi L. Santi, Christopher Barr, Jack M. Fletcher, Al Varisco, and Barbara R. Foorman. 2008. Form effects on the estimation of students' oral reading fluency using DIBELS. *Journal of school psychology* 46(3):315–42. https://doi.org/10.1016/j.jsp.2007.06.003.

Ronald H. Good and Ruth A. Kaminski. 2002. DIBELS oral reading fluency passages for first through third grades. *Technical Report* 10. Eugene, OR: University of Oregon. https://dibels.uoregon.edu/docs/techreports/shaw_csap_technical_report.pdf.

Arthur C. Graesser, Danielle S. McNamara, Max M. Louwerse, and Zhiqiang Cai. 2004. Coh-metrix: Analysis of text on cohesion and language. *Behavior Research Methods, Instruments, & Computers* 36(2):193–202. https://doi.org/10.3758/BF03195564.

Roger Griffiths. 1991. Pausological research in an L2 context: A rationale, and review of selected studies. *Applied Linguistics* 12(4):345–364. https://doi.org/10.1093/applin/12.4.345.

Jan Hasbrouck and Gerald A. Tindal. 2006. Oral Reading Fluency Norms: A Valuable Assessment Tool for Reading Teachers. *The Reading Teacher* 59(7):636–644. https://doi.org/10.1598/RT.59.7.3.

Derrick Higgins, Xiaoming Xi, Klaus Zechner, and David Williamson. 2011. A three-stage approach to the automated scoring of spontaneous spoken responses. *Computer Speech & Language* 25(2):282–306. https://doi.org/10.1016/j.csl.2010.06.001.

Julia Hirschberg. 1993. Pitch accent in context predicting intonational prominence from text. *Artificial Intelligence* 63(1):305 – 340. https://doi.org/https://doi.org/10.1016/0004-3702(93)90020-C.

Julia Hirschberg. 2002. Communication and prosody: Functional aspects of prosody. *Speech Communication* 36(1-2):31–43. https://doi.org/10.1016/S0167-6393(01)00024-3.

Arthur S. House and Grant Fairbanks. 1953. The influence of consonant environment upon the secondary acoustical characteristics of vowels. *The Journal of the Acoustical Society of America* 25(1):105–113. https://doi.org/10.1121/1.1906982.

Matthew L. Jockers. 2015. *Syuzhet: Extract Sentiment and Plot Arcs from Text*. https://CRAN.R-project.org/package=syuzhet.

T. Kendall. 2013. *Speech Rate, Pause and Sociolinguistic Variation: Studies in Corpus Sociophonetics*. Palgrave Macmillan UK.

Dennis H. Klatt. 1976. Linguistic uses of segmental duration in English: Acoustic and perceptual evidence. *The Journal of the Acoustical Society of America* 59(5):1208. https://doi.org/10.1121/1.380986.

Raúl Montaño and Francesc Alías. 2017. The role of prosody and voice quality in indirect storytelling speech: A cross-narrator perspective in four European languages. *Speech Communication* 88:1–16. https://doi.org/10.1016/j.specom.2017.01.007.

Murray J. Munro and Tracey M. Derwing. 1998. The effects of speaking rate on listener evaluations of native and foreign-accented speech. *Language Learning* 48(2):159–182. https://doi.org/10.1111/1467-9922.00038.

Diane Napolitano, Kathleen M. Sheehan, and Robert Mundkowsky. 2015. Online Readability and Text Complexity Analysis with TextEvaluator. In *Proceedings of NAACL-HLT 2015, Denver, Colorado, May 31 - June 5, 2015*. pages 96–100. http://www.aclweb.org/anthology/N15-3020.

Jessica Nelson, Charles Perfetti, David Liben, and Meredith Liben. 2012. Measures of text difficulty: Testing their predictive value for grade levels and student performance. In *Technical Report to the Gates Foundation*. http://achievethecore.org/content/upload/\nelson_perfetti_liben_measures_of_text_difficulty_\research_ela.pdf.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: An ASR corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pages 5206–5210. https://doi.org/10.1109/ICASSP.2015.7178964.

Gordon E. Peterson and Ilse Lehiste. 1960. Duration of Syllable Nuclei in English. *The Journal of the Acoustical Society of America* 32(6):693–703. https://doi.org/10.1121/1.1908183.

Yaacov Petscher and Young-Suk Kim. 2011. The utility and accuracy of oral reading fluency score types in predicting reading comprehension. *Journal of school psychology* 49(1):107–29. https://doi.org/10.1016/j.jsp.2010.09.004.

Hartmut R. Pfitzinger and Uwe D. Reichel. 2006. Text-based and Signal-based Prediction of Break Indices and Pause Durations. In *Proceedings of Speech Prosody 2006*. Dresden, Germany, page 269. https://www.isca-speech.org/archive/sp2006/sp06_269.html.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. 2011. The Kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society.

Vladimir Propp. 1968. *Morphology of the Folktale*. University of Texas Press.

J.K. Rowling. 2015. *Harry Potter and the Sorcerer's Stone*. Published as EPUB file by Pottermore Ltd.

J.K. Rowling and Jim Dale. 2016. *Harry Potter and the sorcerer's stone*. Listening Library/Penguin Random House.

Klaus R. Scherer. 2003. Vocal communication of emotion: A review of research paradigms. *Speech Communication* 40(1-2):227–256. https://doi.org/10.1016/S0167-6393(02)00084-5.

Kathleen M Sheehan, Michael Flor, and Diane Napolitano. 2013. A Two-Stage Approach for Generating Unbiased Estimates of Text Complexity. In *Proceedings of the Workshop on Natural Language Processing for Improving Textual Accessibility*. June, pages 49–58. http://www.aclweb.org/anthology/W13-1506.

Kathleen M. Sheehan, Irene Kostin, Diane Napolitano, and Michael Flor. 2014. The TextEvaluator Tool: Helping teachers and test developers select texts for use in instruction and assessment. *Elementary School Journal* 115(2):184–209. https://doi.org/10.1086/678294.

Steve Tauroza and Desmond Allison. 1990. Speech rates in British English. *Applied Linguistics* 11(1):90–105. https://doi.org/10.1093/applin/11.1.90.

Mariët Theune, Koen Meijs, Dirk Heylen, and Roeland Ordelman. 2006. Generating expressive speech for storytelling applications. *IEEE Transactions on Audio, Speech and Language Processing* 14(4):1137–1144. https://doi.org/10.1109/TASL.2006.876129.

Alice E. Turk and Stefanie Shattuck-Hufnagel. 2000. Word-boundary-related duration patterns in English. *Journal of Phonetics* 28(4):397 – 440. https://doi.org/https://doi.org/10.1006/jpho.2000.0123.

Alice E. Turk and Stefanie Shattuck-Hufnagel. 2007. Multiple targets of phrase-final lengthening in American English words. *Journal of Phonetics* 35(4):445 – 472. https://doi.org/https://doi.org/10.1016/j.wocn.2006.12.001.

Jan P.H. van Santen. 1992. Contextual effects on vowel duration. *Speech Communication* 11(6):513–546. https://doi.org/10.1016/0167-6393(92)90027-5.

Lijuan Wang, Yong Zhao, Min Chu, Yining Chen, Frank K. Soong, and Zhigang Cao. 2006. Exploring expressive speech space in an audio-book. In *In Proceedings of Speech Prosody 2006*. Dresden, Germany, page 182. https://www.isca-speech.org/archive/sp2006/sp06{_}182.html.

Amy Beth Warriner, Victor Kuperman, and Marc Brysbaert. 2013. Norms of valence, arousal, and dominance for 13,915 English lemmas. *Behavior Research Methods* 45(4):1191–1207. https://doi.org/10.3758/s13428-012-0314-x.

Laurence White. 2014. Communicative function and prosodic form in speech timing. *Speech Communication* 63-64:38–54. https://doi.org/10.1016/j.specom.2014.04.003.

Laurence White and Alice E. Turk. 2010. English words on the procrustean bed: Polysyllabic shortening reconsidered. *Journal of Phonetics* 38(3):459 – 471. https://doi.org/https://doi.org/10.1016/j.wocn.2010.05.002.

Carl E. Williams and Kenneth N. Stevens. 1972. Emotions and Speech: Some Acoustical Correlates. *The Journal of the Acoustical Society of America* 52(4B):1238–1250. https://doi.org/10.1121/1.1913238.

Klaus Zechner, Keelan Evanini, and Cara Laitusis. 2012. Using Automatic Speech Recognition to Assess the Reading Proficiency of a Diverse Sample of Middle School Students. In *Proceedings of the Third Workshop on Child, Computer Interaction (WOCCI 2012), Portland, OR, USA*. International Speech Communications Association., Portland, OR, pages 45–52. http://www.isca-speech.org/archive/wocci_2012/wc12_045.html.

Heiga Zen, Keiichi Tokuda, and Alan W. Black. 2009. Statistical parametric speech synthesis. *Speech Communication* 51(11):1039–1064. https://doi.org/10.1016/j.specom.2009.04.004.

Yuan Zhao and Dan Jurafsky. 2009. The effect of lexical frequency and Lombard reflex on tone hyperarticulation. *Journal of Phonetics* 37(2):231–247. https://doi.org/10.1016/j.wocn.2009.03.002.

Generating Bilingual Pragmatic Color References

Will Monroe
Computer Science Department
Stanford University
wmonroe4@cs.stanford.edu

Jennifer Hu
Department of Mathematics
Harvard University
jenniferhu@college.harvard.edu

Andrew Jong
Department of Computer Science
San Jose State University
andrewjong.cs@gmail.com

Christopher Potts
Department of Linguistics
Stanford University
cgpotts@stanford.edu

Abstract

Contextual influences on language often exhibit substantial cross-lingual regularities; for example, we are more verbose in situations that require finer distinctions. However, these regularities are sometimes obscured by semantic and syntactic differences. Using a newly-collected dataset of color reference games in Mandarin Chinese (which we release to the public), we confirm that a variety of constructions display the same sensitivity to contextual difficulty in Chinese and English. We then show that a neural speaker agent trained on bilingual data with a simple multitask learning approach displays more human-like patterns of context dependence and is more pragmatically informative than its monolingual Chinese counterpart. Moreover, this is not at the expense of language-specific semantic understanding: the resulting speaker model learns the different basic color term systems of English and Chinese (with noteworthy cross-lingual influences), and it can identify synonyms between the two languages using vector analogy operations on its output layer, despite having no exposure to parallel data.

1 Introduction

In grounded communication tasks, speakers face pressures in choosing referential expressions that distinguish their targets from others in the context, leading to many kinds of pragmatic meaning enrichment. For example, the harder a target is to identify, the more the speaker will feel the need to refer implicitly and explicitly to alternatives to draw subtle contrasts (Zipf, 1949; Horn, 1984; Levinson, 2000). However, the ways in which these contrasts are expressed depend heavily on language-specific syntax and semantics.

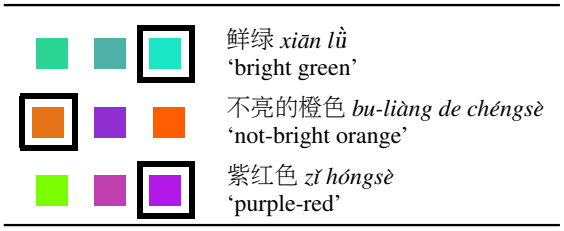


Figure 1: Reference game contexts and utterances from our Chinese corpus. The boxed color is the target. Some color terms show differences between Chinese and English, such as 绿 *lǜ* 'green' in the first example for a color that might be referred to with 'blue' or 'aqua' in English.

In this paper, we seek to develop a model of contextual language production that captures language-specific syntax and semantics while also exhibiting responsiveness to contextual differences. We focus on a color reference game (Rosenberg and Cohen, 1964; Dale and Reiter, 1995; Krahmer and van Deemter, 2012) played in both English and Mandarin Chinese. A reference game (Figure 1) involves two agents, one designated the "speaker" and the other the "listener". The speaker and listener are shown the same set of k colors $C = \{c_1, \ldots, c_k\}$ (in our experiments, $k = 3$), and one of these colors c_t is indicated secretly to the speaker as the "target". Both players share the same goal: that the listener correctly guesses the target color. The speaker may communicate with the listener in free-form natural-language dialogue to achieve this goal. Thus, a model of the speaker must process representations of the colors in the context and produce an utterance to distinguish the target color from the others. We evaluate a sequence-to-sequence speaker

Proceedings of NAACL-HLT 2018, pages 2155–2165
New Orleans, Louisiana, June 1 - 6, 2018.

agent based on that of Monroe et al. (2017), who also collected the English data we use; our Chinese data are new and were collected according to the same protocols.

While English and Chinese both use fairly similar syntax for color descriptions, our reference game is designed to elicit constructions that make reference to the context, and these constructions—particularly comparatives and negation—differ morpho-syntactically and pragmatically between the two languages. Additionally, Chinese is considered to have a smaller number of basic color terms (Berlin and Kay, 1969), which predicts markedness of more specific descriptions.

Our primary goal is to examine the effects of *bilingual* training: building one speaker trained on both English and Chinese data with a shared vocabulary, so that it can produce utterances in either language. The reference game setting offers an objective measure of success on the grounded language task, namely, the speaker's ability to guide the listener to the target. We use this to address the tricky problem of speaker evaluation. Specifically, we use the speaker model and an application of Bayes' rule to infer the most likely target color given a human utterance, and we report the accuracy of that process at identifying the target color. We refer to this metric as *pragmatic informativeness* because it requires not only accuracy but also effectiveness at meeting the players' shared goal (Grice, 1975). A more formal definition and a discussion of alternatives are given in Section 4.1.

We show that a bilingually-trained model produces distributions over Chinese utterances that have higher pragmatic informativeness than a monolingual model. An analysis of the learned word embeddings reveals that the bilingual model learns color synonyms between the two languages without being directly exposed to labeled pairs. However, using a context-independent color term elicitation task from Berlin and Kay (1969) on our models, we show that the learned lexical meanings are largely faithful to each language's basic color system, with only minor cross-lingual influences. This suggests that the improvements due to adding English data are not primarily due to better representations of the input colors or lexical semantics alone. The bilingual model does better resemble human patterns of utterance length as a function of contextual difficulty, suggesting the pragmatic level as one possible area of cross-lingual generalization.

2 Data collection

We adapted the open-source reference game framework of Hawkins (2015) to Chinese and followed the data collection protocols of Monroe et al. (2017) as closely as possible, in the hope that this can be the first step in a broader multilingual color reference project. We recruit pairs of players on Amazon Mechanical Turk in real time, randomly assigning one the role of the speaker and the other the listener. Players are self-reported Chinese speakers, but they must pass a series of Chinese comprehension questions in order to proceed, with instructions in a format preventing copy-and-paste translation. The speaker and listener are placed in a game environment in which they both see the three colors of the context and a chatbox. The speaker sends messages through the chatbox to describe the target to the listener, who then attempts to click on the target. This ends the round, and three new colors are generated for the next. Both players can send messages through the chatbox at any time. After filtering out extremely long messages (number of tokens greater than 4σ above the mean), spam games,[1] and players who self-reported confusion about the game, we have a new corpus of 5,774 Chinese messages in color reference games, which we will release publicly. Data management information is given in Appendix B.

As in Monroe et al. (2017), the contexts are divided into three groups of roughly equal size: in the *far* condition (1,421 contexts), all the colors are at least a threshold distance θ from each other; in the *split* condition (1,412 contexts), the target and one distractor are less than θ from each other, with the other distractor at least θ away from both; and in the *close* condition (1,425 contexts), all colors are within θ from each other. We set $\theta = 20$ by the CIEDE2000 color-difference formula (Sharma et al., 2005), with all colors different by at least 5.

3 Human data analysis

As we mentioned earlier, our main goal with this work is to investigate the effects of bilingual training on pragmatic language use. We first examine the similarities and differences in pragmatic be-

[1] Some players found they could advance through rounds by sending duplicate messages. Games were considered spam if the game contained 25 or more duplicates.

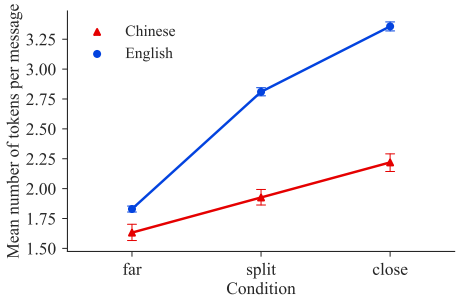


Figure 2: Comparison of mean length of messages in English and Chinese. The *split* and *close* conditions have more similar context colors (Section 2).

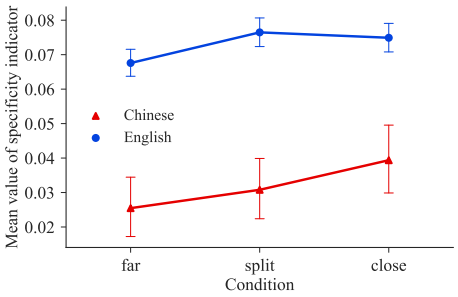


Figure 3: Comparison of WordNet specificity in Chinese and English.

haviors between the English and Chinese corpora we use. The picture that emerges accords well with our expectations about pragmatics: the broad patterns are aligned across the two languages, with the observed differences mostly tracing to the details of their lexicons and constructions.

3.1 Message length

We expect message length to correlate with the difficulty of the context: as the target becomes harder to distinguish from the distractors, the speaker will produce more complex messages, and length is a rough indicator of such complexity. To test this hypothesis, we used the Natural Language Toolkit (NLTK; Bird et al. 2009) and Jieba (Junyi, 2015) to tokenize English and Chinese messages, respectively, and counted the number of tokens in both languages as a measure of message length. The results (Figure 2) confirm that in both languages, players become more verbose in more difficult conditions.[2]

3.2 Specificity

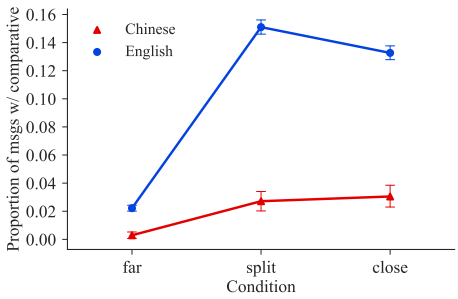
In the *split* and *far* conditions, the speaker must make fine-grained distinctions. A broad color term like *red* will not suffice if there are two reds, but more specific terms like *maroon* might identify the target. Thus, we expect specificity to increase as the difficulty of the context does. To assess this, we use WordNet (Fellbaum, 1998) to transform adjectives into derivationally-related noun forms, filter for nouns with *color* in their hypernym paths, and mark a message as "specific" if it contains at least one word with a hypernym depth greater than 7.

For Chinese, we translate to English via Google Translate, then measure the translated word using WordNet. It should be noted that this method has the drawback of obscuring differences between the two languages' color systems, as well as the potential for introducing noise due to errors in automatic translation. Though Mandarin variations of WordNet exist, we chose this translation method to standardize hypernym paths for both languages. Differences in ontology decisions between lexical resources prevent straightforward cross-lingual comparisons of hypernym depths, while automatic translation to a common language ensures the resulting hypernym paths are directly comparable.

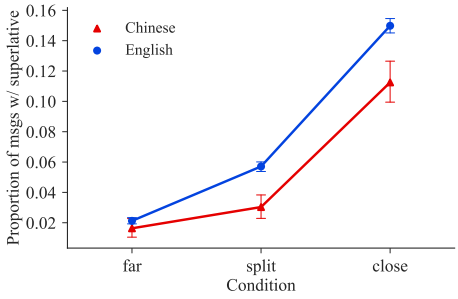
Figure 3 summarizes the results of this measurement. In general, the usage of high-specificity color words increases in more difficult conditions, as expected. However, we see that Chinese speakers use them significantly less than English speakers. Instead, Chinese speakers use nominal modifiers, such as 草 *cǎo* 'grass' and 海 *hǎi* 'ocean', which do not contain "color" in their hypernym paths and are thus not marked as high-specificity. To quantify this observation, we annotated random samples of 200 messages from each language for whether they contained nominal color descriptions, and found that 3.5% of the English messages contain such nominals versus 13.5% of the Chinese messages.

The use of nominal modifiers as opposed to adjectives ('dark orange', 'dull brown') is arguably expected given the claims of Berlin and Kay (1969) and others that Chinese has fewer basic color terms than English, thus requiring more visually evocative modifiers to clarify distinctions between similar hues. (This isn't a complete explanation, since Chinese is rich in narrow but rare

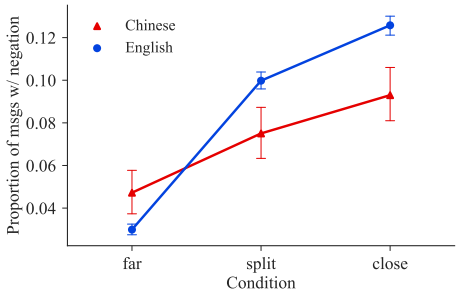
[2] We do not believe that the overall drop in message length from English to Chinese reflects a fundamental difference between the languages; this has a few possible explanations, from Chinese messages taking the form of "sentence segments" (Wang and Qin, 2010) to differences in tokenization.



(a) Usage of comparative adjectives in Chinese and English.



(b) Usage of superlative adjectives in Chinese and English.



(c) Usage of negation in Chinese and English.

Figure 4: Comparison of usage of comparatives, superlatives, and negation in English and Chinese.

non-basic color terms. For the cases where Chinese has an appropriate narrow color term, it is possible that speakers make a pragmatic decision to avoid obscure vocabulary in favor of more familiar nouns.)

3.3 Comparatives, superlatives, and negation

To detect comparative and superlative adjectives in English, we use NLTK POS-tagging, which outputs JJR and RBR for comparatives, and JJS and RBS for superlatives. In Chinese, we look for the tokens 更 *gèng* 'more' and 比 *bǐ* 'comparatively' to detect comparatives and 最 *zuì* 'most' to detect superlatives. We detect negation by tokenizing messages with NLTK and Jieba and then looking for the tokens *not* and *n't* in English and corresponding 不 *bù* and 没 *méi* in Chinese.

These statistics are shown in Figure 4. Both languages exhibit similar trends for superlative adjectives. In English, comparatives are used most frequently in the *split* condition and second most frequently in the *close* condition, while in Chinese, they occur at around the same rate in the *split* and *close* conditions. The literature is not conclusive about the source of these differences. Xia (2014) argues that complex attributives are rarely used and sound "syntactically deviant or Europeanized" (Zhu, 1982; Xie, 2001) in Chinese, citing the left-branching nature of the language as restricting attributives in length and complexity. There are also conflicting theories on the markedness of gradable adjectives in Chinese (Grano, 2012; Ito, 2008); such markedness may contribute to the frequency at which comparative forms are used.

We also see that both languages follow the same general trend of using negation more frequently as the condition becomes more difficult.

4 Models

We build and evaluate three artificial agents on this reference game task, two trained on monolingual descriptions (one for each language) and one on bilingual descriptions. We base these models on the basic speaker architecture from Monroe et al. (2017). The monolingual speakers represent the context by passing all the context colors as input to a long short-term memory (LSTM) sequence encoder, then concatenating this representation with a word vector for each previous output token as the input to an LSTM decoder that produces a color description token-by-token. This defines a distribution over descriptions u conditioned on the target and context, $S(u \mid c_t, C)$.

To accommodate bilingual training with this architecture, we expand the vocabulary to include English and Chinese words, and we add a flag ℓ to the input specifying whether the model's output should be in English ($\ell = 0$) or Chinese ($\ell = 1$):

$$S(u \mid \ell, c_t, C) = \prod_{i=1}^{|u|} s(u_i \mid u_{1..i-1}, \ell, c_t, C)$$

The flag ℓ is embedded as a single additional dimension that is concatenated alongside the context and input (previous token) vectors for the encoder. See Appendix A for additional training details.

4.1 Pragmatic informativeness

As mentioned in Section 1, we evaluate the two models on a measure of *pragmatic informative-*

ness: how well does the model represent a human speaker, such that a generative model of a listener can be built from it to interpret utterances? Formally, for a speaker $S(u \mid \ell, c_t, C)$ and an example consisting of an utterance, language identifier, and color context (u, ℓ, C), we identify the t^* that maximizes the probability of u according to S:

$$t^* = \arg\max_t S(u \mid c_t, C)$$

That is, L uses a noisy-channel model with a uniform prior over target colors and S as a generation model to infer the most likely target color given the input utterance. The pragmatic informativeness of a speaker is the proportion of target colors in a test set correctly identified by t^*.

One drawback of this metric is it does not evaluate how faithful the model is to the overall distribution of human utterances, only the relative conditional likelihoods of human utterances for different target colors. In practice, since the agents are trained to minimize log likelihood, we do not observe our agents frequently producing wildly unhumanlike utterances; however, this is a caveat to keep in mind for evaluating agents that do not naturally approximate a language model.

The understanding model implied in this metric is equivalent to a version of the Rational Speech Acts model of pragmatic language understanding (Frank and Goodman, 2012; Goodman and Frank, 2016), or the *pragmatic posterior* of the Rational Observer model (McMahan and Stone, 2015). An important difference between our speaker model and those in the work cited above is that our speaker model is a neural network that makes a combined judgment of applicability (semantic appropriateness) and availability (utterance prior), instead of modeling the two components separately. However, we stop short of directly predicting the referent of an expression discriminatively, as is done by e.g. Kennington and Schlangen (2015), so as to require a model that is usable as a speaker.

A related metric is *communicative success* as defined by Golland et al. (2010), which judges the speaker by the accuracy of a human listener when given model-produced utterances. Our pragmatic informativeness metric instead gives a model-derived listener human utterances and assesses its accuracy at identifying colors. Pragmatic informativeness has the advantage of not requiring additional expensive human labeling in response to model outputs; it can be assessed on an existing collection of human utterances, and can therefore be considered an automatic metric.

4.2 A note on perplexity

Perplexity is a common intrinsic evaluation metric for generation models.[3] However, for comparing monolingual and bilingual models, we found perplexity to be unhelpful, owing largely to its vocabulary-dependent definition. Specifically, if we fix the vocabulary in advance to include tokens from both languages, then the monolingual model performs unreasonably poorly, and bilingual training helps immensely. However, this is an unfair comparison: the monolingual model's high perplexity is dominated by low probabilities assigned to rare tokens in the opposite-language data that it did not see. Thus, perplexity ceases to be a measure of language modeling ability and assumes the role of a proxy for the out-of-vocabulary rate.

On the other hand, if we define the output vocabulary to be the set of tokens seen at least n times in training ($n = 1$ and 2 are common), then monolingual training yields better perplexity than bilingual training, but mainly because including opposite-language training data forces the bilingual model to predict more rare words that would otherwise be replaced with ⟨unk⟩.[4] This produces the counterintuitive result that perplexity initially goes *up* (gets worse) when increasing the amount of training data. (As a pathological case, with no training data, a model can get a perfect perplexity of 1 by predicting ⟨unk⟩ for every token.)

5 Experimental results and analysis

Pragmatic informativeness of the models on English and Chinese data is shown in Table 1. The main result is that training a bilingual model helps compared to a Chinese monolingual one; however, the benefit is asymmetrical, as training on monolingual English data is superior for English data to training on a mix of Chinese and English. All differences in Table 1 are significant at $p < 0.001$

[3] Two other intrinsic metrics, word error rate (WER) and BLEU (Papineni et al., 2002), were at or worse than chance despite qualitatively adequate speaker outputs, due to high diversity in valid outputs for similar contexts. This problem is common in dialogue tasks, for which BLEU is known to be an ineffective speaker evaluation metric (Liu et al., 2016).

[4] The rare words that make this difference are primarily the small number of English words that were used by the Chinese-language participants; no Chinese words were observed in the English data from Monroe et al. (2017)

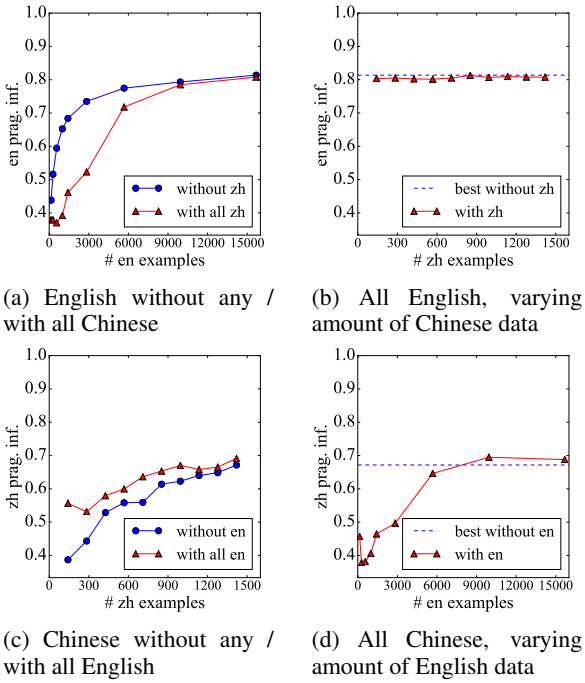
test	train	dev acc	test acc
en	en	**80.51**	**83.06**
	en+zh	79.73	81.43
zh	zh	67.16	67.75
	en+zh	**71.81**	**72.89**

Table 1: Pragmatic informativeness scores (%) for monolingual and bilingual speakers.

(approximate permutation test, 10,000 samples; Padó, 2006), except for the decrease on the English dev set, which is significant at $p < 0.05$.

An important difference between our corpora is that the English dataset is an order of magnitude larger than the Chinese. Intuitively, we expect adding more training data on the same task will improve the model, regardless of language. However, we find that the effect of dataset size is not so straightforward. In fact, the differences in training set size convey a non-linear benefit. Figure 5 shows the pragmatic informativeness of the monolingual and bilingual speakers on the development set as a function of dataset size (number of English and Chinese utterances). The blue curves (circles) in the plots on the left, Figure 5a and Figure 5c, are standard learning curves for the monolingual models, and their parallel red curves (triangles) show the pragmatic informativeness of the bilingual model with the same amount of in-language data plus all available data in the opposite language. The plots on the right, Figure 5b and Figure 5d, show the effect of gradually adding opposite-language data to the bilingual model starting with all of the in-language data.

Overall, we see that adding all English data consistently helps the Chinese monolingual model, whereas adding all Chinese data consistently hurts the English monolingual model (though with diminishing effects as the amount of English data increases). Adding small amounts of English data—especially amounts comparable to the size of the Chinese dataset—*decreases* accuracy of the Chinese model dramatically. This suggests an interaction between the total amount of data and the effect of bilingual training: a model trained on a moderately small number of in-language examples can benefit from a much larger training set in another language, but combining data in two languages is detrimental when both datasets are very small and has very little effect when the in-



(a) English without any / with all Chinese

(b) All English, varying amount of Chinese data

(c) Chinese without any / with all English

(d) All Chinese, varying amount of English data

Figure 5: Pragmatic informativeness (dev set) for different amounts and languages of training data.

language training set is large. This implies a benefit primarily in low-resource settings, which agrees with the findings of Johnson et al. (2016) using a similar architecture for machine translation.

5.1 Bilingual lexicon induction

To get a better understanding of the influence of the bilingual training on the model's lexical representations in the two languages, we extracted the weights of the final softmax layer of the bilingual speaker model and used them to induce a bilingual lexicon with a word vector analogy task. For two pairs of lexical translations, 蓝色 *lánsè* → "blue" and "red" → 红 *hóng*, we took the difference between the source language word vector and the target language word vector. To "translate" a word, we added this "translation vector" to the word vector for the source word, and found the word in the opposite language with the largest inner product to the resulting vector. The results are presented in Table 2. We identified the 10 most frequent color-related words in each language to translate. (In other words, we did not use this process to find translations of function words like "the" or the Chinese nominalization/genitive particle 的 *de*, but we show proposed translations that were not color-related, such as 灰 *huī* being translated as the English comparative ending "-er".)

zh	en	en	zh
绿色 ‘green’	**green**	green	**绿 ‘green’**
紫色 ‘purple’	**purple**	blue	**蓝 ‘blue’**
蓝色 ‘blue’	purple	purple	蓝 ‘blue’
灰色 ‘grey’	**grey**	bright	**鲜艳 ‘bright’**
亮 ‘bright’	**bright**	pink	**粉色 ‘pink’**
灰 ‘grey’	-er	grey	**灰 ‘grey’**
蓝 ‘blue’	*teal*	dark	**暗 ‘dark’**
绿 ‘green’	**green**	gray	**灰 ‘grey’**
紫 ‘purple’	**purple**	yellow	**黄色 ‘yellow’**
草 ‘grass’	*green*	light	最 ‘most’

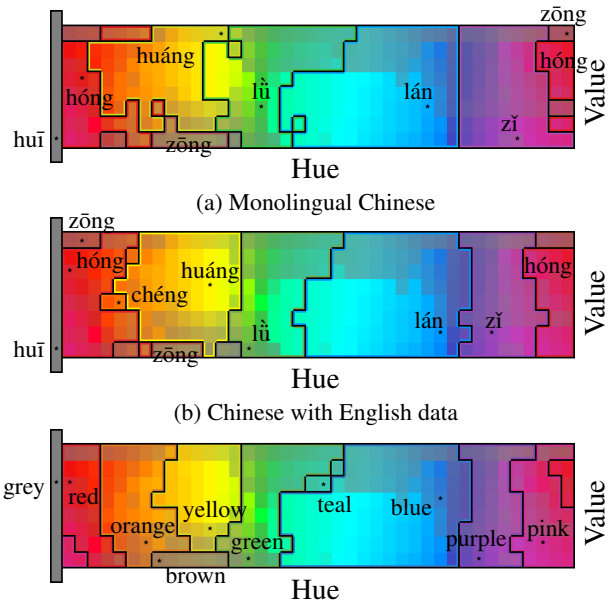
Table 2: Bilingual lexicon induction from Chinese to English (first two columns) and vice versa (last two). Correct translations in **bold**, semantically close words in *italic*.

The majority of common color words are translated correctly by this simple method, showing that the vectors in the softmax layer do express a linear correspondence between the representation of synonyms in the two languages.

5.2 Color term semantics

The above experiment suggests that the bilingual model has learned word semantics in ways that discover translation pairs. However, we wish to know whether bilingual training has resulted in changes to the model's output distribution reflecting differences in the two languages' color systems. To evaluate this, we performed an experiment similar to the basic color term elicitations in the World Color Survey (WCS; Berlin and Kay, 1969) on our models. For each of the 330 colors in the original WCS, we presented that color to our monolingual and bilingual models and recorded the most likely color description according to the conditional language model. Our models require a three-color context to produce a description; as an approximation to eliciting context-insensitive color terms, we gave the model ten contexts with randomly generated (uniform in H, S, and V) distractor colors and averaged the language model probabilities. We also identified, for each color term produced as the most likely description of one or more colors, the color that resulted in the highest probability of producing that term.

The results are in Figure 6. The charts use the layout of the WCS stimulus, in which the two axes represent dimensions of color variation similar to hue and lightness. Each region represents a set of colors that the model labeled with the same color term, and a star marks the color that resulted in the highest probability of producing that term. The Chinese terms, except for 红 *hóng*, are abbreviated by deleting the final morpheme 色 *sè* ‘color’.



(a) Monolingual Chinese

(b) Chinese with English data

(c) Monolingual English

Figure 6: Color term lexica: colors in the World Color Survey palette grouped by highest-probability description, averaged over 10 randomly-generated pairs of distractor colors. The color that results in the highest probability of each description is marked with a star. English influences on the bilingual model include the appearance of 橙色 *chéngsè* ‘orange’ and narrowing of 黄色 *huángsè* ‘yellow’ and 绿色 *lǜsè* ‘green’.

The charts agree with Berlin and Kay (1969) on most of the differences between the two languages: *orange* and *pink* have clear regions of dominance in English, whereas in the Mandarin monolingual model *pink* is subsumed by 红 *hóng* ‘red’, and *orange* is subsumed by 黄色 *huángsè* ‘yellow’. Our models produce three colors not in the six-color system[5] identified by Berlin and Kay for Mandarin: 灰色 *huīsè* ‘grey’, 紫色 *zǐsè* ‘purple’, and 棕色 *zōngsè* ‘brown’. We do not specifically claim these should be considered *basic* color terms, since Berlin and Kay give a theoretical definition of “basic color term” that is not rigorously captured by our model. In particular, they explicitly exclude 灰色 *huīsè* from the set of basic color terms, despite its frequency, because it has a mean-

[5]Notably absent are ‘black’ and ‘white’. The collection methodology of Monroe et al. (2017) restricted colors to a single lightness, so black and white are not in the data. For these charts, we replaced the World Color Survey swatches with the closest color used in our data collection.

ing that refers to an object ('ashes'). The other two may have been excluded for the same reason, or they may represent a change in the language or the influence of English on the participants' usage.[6]

A few differences between the monolingual and bilingual models can be characterized as an influence of one language's color system on the other. First, *teal* appears as a common description of a few color swatches from the English monolingual model, but the bilingual model, like the Chinese model, does not feature a common word for teal. Second, the Chinese monolingual model does not include a common word for orange, but the bilingual model identifies 橙色 *chéngsè* 'orange'. Finally, the English *green* is semantically narrower than the Chinese 绿色 *lǜsè*, and the Chinese bilingual model exhibits a corresponding narrowing of the range of 绿色 *lǜsè*.

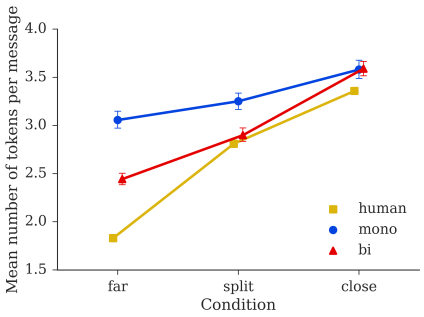
Overall, however, the monolingual models capture largely accurate maps of each language's basic color system, and the bilingual model retains the major contrasts between them, rather than "averaging" between the two. This suggests that the bilingual model learns a representation of the input colors that encodes their categorization in both languages, and that for the most part these lexical semantic representations do not influence each other.

5.3 Comparing model and human utterances

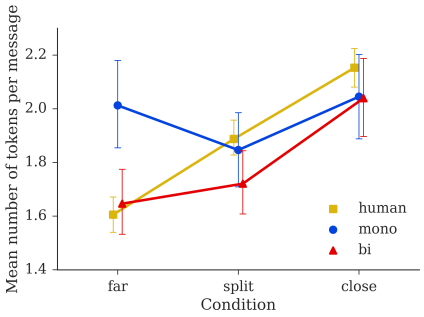
One observation indicates that the improvements in the bilingually-trained model are primarily at the pragmatic (context-dependent) level of language production. Figure 7 reveals that the bilingually-trained model better captures the main pragmatic pattern we observe in the human data, that of increasing message length in harder conditions. In both languages, the monolingual model uses longer utterances in the easy *far* condition than human speakers do, whereas the bilingual model is significantly closer on that condition to the human statistics. We see similar results in the use of negations and comparatives; the use of superlatives is not substantially different between the monolingual and bilingual models.

We note that this result does not rule out several competing hypotheses. In particular, we do not exclude improvements in compositional semantics or syntax, nor do we distinguish improvements in specific linguistic areas from broader regularization effects of having additional data in general. Preliminary experiments involving augmentation of the data by duplicating and deleting constituents show no gains, suggesting that the improvement depends on certain kinds of regularities in the English data that are not provided by artificial manipulations. However, more investigation is needed to thoroughly assess the role of general-purpose regularization in our observations.

[6]MTurk's restriction to US workers makes English influence more likely than would otherwise be expected.



(a) Human and model utterance lengths in English.



(b) Human and model utterance lengths in Chinese.

Figure 7: Comparison of mean length of messages between human and model utterances.

6 Related work

The method we use to build a bilingual model involves adding a single dimension to the previous-token vectors in the encoder representing the language (Section 4). In essence, the two languages have separate vocabulary representation at the input and output but shared hidden representations. Adding a hard constraint on the output vocabulary would make this equivalent to a simple form of multitask learning (Caruana, 1997; Collobert and Weston, 2008). However, allowing the model to use tokens from either language at any time is simpler and results in better modeling of mixed-language data, which is more common in non-English environments. In fact, our model occasionally ignores the flag and "code-switches" be-

tween the two languages within a single output, which is not possible in typical multitask architectures.

Using shared parameters for cross-lingual representation transfer has a large literature. Klementiev et al. (2012) and Hermann and Blunsom (2014) use multitask learning with multilingual document classification to build cross-lingual word vectors, and observe accurate lexical translations from linear vector analogy operations. They include predicting translations for words in parallel data as one of their tasks. Our translations from vector relationships (Section 5.1) derive their cross-lingual relationships from the non-linguistic input of our grounded task, without parallel data.

Huang et al. (2013) note gains in speech recognition from cross-lingual learning with shared parameters. In machine translation, Johnson et al. (2016) add the approach of setting the output language using a symbol in the input. Kaiser et al. (2017) extend this to image captioning, speech recognition, and parsing in one multitask system. Our work complements these efforts with an in-depth analysis of bilingual training on a grounded generation task and an exploration of the relationship between cross-lingual semantic differences and pragmatics. In general, we see grounding in non-linguistic input, including images and sensory input from real and simulated worlds, as an intriguing substitute for direct linguistic supervision in low-resource settings. We encourage evaluation of multitask and multilingual models on tasks that require reference to the context for effective language production and understanding.

7 Conclusion

In this paper, we studied the effects of training on bilingual data in a grounded language task. We show evidence that bilingual training can be helpful, but with a non-obvious effect of dataset size: accuracy as a function of opposite-language data follows a U-shaped curve. The resulting model is more human-like in measures of sensitivity to contextual difficulty (pragmatics), while exhibiting language-specific lexical learning in the form of vector relationships between lexical pairs and differences between the two languages in common color-term extensions (semantics).

It should be noted that color descriptions in English and Chinese are similar both in their syntax and in the way they divide up the semantic space. We might expect that for languages like Arabic and Spanish (with their different placement of modifiers), or Waorani and Pirahã (with their much smaller color term inventories), the introduction of English data could have detrimental effects that outweigh the language-general gains. An investigation across a broader range of languages is desirable.

Our contribution includes a new dataset of human utterances in a color reference game in Mandarin Chinese, which we release to the public[7] with our code and trained model parameters.[8]

Acknowledgments

We thank Jiwei Li for extensive editing of our Chinese translations of the Mechanical Turk task instructions, Robert X.D. Hawkins for assistance setting up the data collection platform, and members of the Stanford NLP group—particularly Reid Pryzant, Sebastian Schuster, and Reuben Cohn-Gordon—for valuable feedback on earlier drafts. This material is based in part upon work supported by the Stanford Data Science Initiative and by the NSF under Grant Nos. BCS-1456077 and SMA-1659585.

References

Brent Berlin and Paul Kay. 1969. *Basic Color Terms: their Universality and Evolution*. University of California Press, Berkeley and Los Angeles.

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

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28:41–75.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 160–167. ACM.

Robert Dale and Ehud Reiter. 1995. Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive Science*, 19(2):233–263.

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

[7] https://cocolab.stanford.edu/datasets/colors.html

[8] https://github.com/futurulus/colors-in-context

Michael C. Frank and Noah D. Goodman. 2012. Predicting pragmatic reasoning in language games. *Science*, 336(6084):998.

Dave Golland, Percy Liang, and Dan Klein. 2010. A game-theoretic approach to generating spatial descriptions. In *Proceedings of the 2010 Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

Noah D. Goodman and Michael C. Frank. 2016. Pragmatic language interpretation as probabilistic inference. *Trends in Cognitive Sciences*, 20(11):818–829.

Thomas Grano. 2012. Mandarin 'hen' and universal markedness in gradable adjectives. *Natural Language & Linguistic Theory*, 30(2):513–565.

H. Paul Grice. 1975. Logic and conversation. In Peter Cole and Jerry Morgan, editors, *Syntax and Semantics*, volume 3: Speech Acts, pages 43–58. Academic Press, New York.

Robert X. D. Hawkins. 2015. Conducting real-time multiplayer experiments on the web. *Behavior Research Methods*, 47(4):966–976.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 58–68. Association for Computational Linguistics.

Laurence R. Horn. 1984. Toward a new taxonomy for pragmatic inference: Q-based and R-based implicature. In Deborah Schiffrin, editor, *Meaning, Form, and Use in Context: Linguistic Applications*, pages 11–42. Georgetown University Press, Washington, D.C.

Jui-Ting Huang, Jinyu Li, Dong Yu, Li Deng, and Yifan Gong. 2013. Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers. In *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7304–7308. IEEE.

Satomi Ito. 2008. Typology of comparatives. In *Proceedings of the 22nd Pacific Asia Conference on Language, Information and Computation (PACLIC)*, pages 197–206.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2016. Google's multilingual neural machine translation system: enabling zero-shot translation. *arXiv preprint arXiv:1611.04558*.

Sun Junyi. 2015. Jieba Python Library.

Łukasz Kaiser, Aidan N. Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. 2017. One model to learn them all. *arXiv preprint arXiv:1706.05137*.

Casey Kennington and David Schlangen. 2015. Simple learning and compositional application of perceptually grounded word meanings for incremental reference resolution. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 292–301. Association for Computational Linguistics.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *Proceedings of COLING 2012*, pages 1459–1474. International Committee on Computational Linguistics.

Emiel Krahmer and Kees van Deemter. 2012. Computational generation of referring expressions: A survey. *Computational Linguistics*, 38(1):173–218.

Stephen C. Levinson. 2000. *Presumptive Meanings: The Theory of Generalized Conversational Implicature*. MIT Press, Cambridge, MA.

Chia-Wei Liu, Ryan Lowe, Iulian V. Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2122–2132. Association for Computational Linguistics.

Brian McMahan and Matthew Stone. 2015. A Bayesian model of grounded color semantics. *Transactions of the Association for Computational Linguistics*, 3:103–115.

Will Monroe, Robert X.D. Hawkins, Noah D. Goodman, and Christopher Potts. 2017. Colors in context: A pragmatic neural model for grounded language understanding. *Transactions of the Association for Computational Linguistics*, 5:325–338.

Sebastian Padó. 2006. *User's guide to `sigf`: Significance testing by approximate randomisation*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics.

Seymour Rosenberg and Bertram D. Cohen. 1964. Speakers' and listeners' processes in a word communication task. *Science*, 145(3637):1201–1203.

Gaurav Sharma, Wencheng Wu, and Edul N. Dalal. 2005. The CIEDE2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations. *Color Research & Application*, 30(1):21–30.

Kefei Wang and Hongwu Qin. 2010. A parallel corpus-based study of translational chinese. In Richard Xiao, editor, *Using Corpora in Contrastive and Translation Studies*, pages 164–181. Cambridge Scholars Publishing.

Yun Xia. 2014. *Normalization in Translation: Corpus-based Diachronic Research into Twentieth-century English Chinese Fictional Translation*. Cambridge Scholars Publishing.

Yaoji Xie. 2001. 汉语语法欧化综述 / Hànyǔ yǔfǎ Ōuhuà zōngshù (A review of Europeanized Chinese grammar). 语文研究 *Yǔwén Yánjiū (Chinese Language Research)*, 1:17–22.

Dexi Zhu. 1982. 语法讲义 / *Yǔfǎ Jiǎngyì (Lectures on Grammar)*. The Commercial Press, Beijing.

George Kingsley Zipf. 1949. *Human Behavior and the Principle of Least Effort*. Addison-Wesley, Cambridge.

A Model details

Hyperparameters for the two main models are given in Table 3. Both Chinese monolingual and bilingual model were tuned for perplexity on a held-out subset of the training set, by random search followed by a local search from the best candidate until no single parameter change produced a better result. However, the tuned settings for the Chinese monolingual model did not outperform the settings from Monroe et al. (2017) for the English model on the development set, so in our final experiments the monolingual models used the same parameters.

The vocabulary for each model consisted of all tokens that were seen at least twice in training; the bilingual model's vocabulary is larger than the union of the words in each monolingual model because some tokens occurred once in each language (largely meta-commentary—e.g., *dunno*, *HIT*, *xD*—and some English color word typos).

hyperparameter	mono.	biling.
optimizer	ADAM	RMSProp
learning rate	0.004	0.004
dropout	0.1	0.1
gradient clip norm	–	1
LSTM cell size	100	50
embedding size	100	100
initial forget bias	0	5
nonlinearity	tanh	sigmoid
vocabulary size	895 (en) 260 (zh)	1,326

Table 3: Values of hyperparameters optimized in tuning for the monolingual and bilingual models, plus vocabulary sizes.

B Data management

This work was done under Stanford IRB Protocol 17827, which has the title "Pragmatic enrichment and contextual inference". Data was only collected from workers who indicated their informed consent. Workers on Amazon Mechanical Turk were paid $2.00 to complete each game consisting of 50 dialogue contexts, plus a bonus of $0.01 for each target the listener correctly identified. All worker identifiers have been removed from data that is released; the only other information collected about the workers was their Chinese language proficiency.

Learning with Latent Language

Jacob Andreas Dan Klein Sergey Levine
Computer Science Division
University of California, Berkeley
{jda,klein,svlevine}@eecs.berkeley.edu

Abstract

The named concepts and compositional operators present in natural language provide a rich source of information about the abstractions humans use to navigate the world. Can this linguistic background knowledge improve the generality and efficiency of learned classifiers and control policies? This paper aims to show that using the space of natural language strings as a *parameter* space is an effective way to capture natural task structure. In a pretraining phase, we learn a language interpretation model that transforms inputs (e.g. images) into outputs (e.g. labels) given natural language descriptions. To learn a new concept (e.g. a classifier), we search directly in the space of descriptions to minimize the interpreter's loss on training examples. Crucially, our models do not require language data to learn these concepts: language is used only in pretraining to impose structure on subsequent learning. Results on image classification, text editing, and reinforcement learning show that, in all settings, models with a linguistic parameterization outperform those without.[1]

1 Introduction

The structure of natural language reflects the structure of the world. For example, the fact that it is easy for humans to communicate the concept *left of the circle* but comparatively difficult to communicate *mean saturation of the first five pixels in the third column* reveals something about the abstractions we find useful for interpreting and navigating our environment (Gopnik and Meltzoff, 1987). In machine learning, efficient automatic discovery of reusable abstract structure remains a major challenge. This paper investigates whether

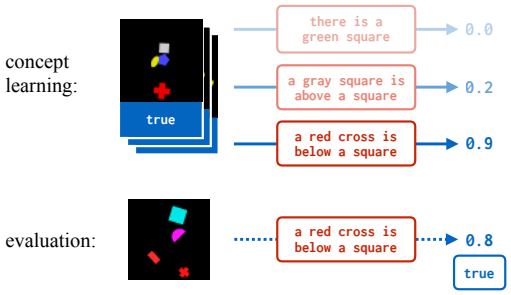


Figure 1: Example of our approach on a binary image classification task. We assume access to a pretrained language interpretation model that outputs the probability that an image matches a given description. To learn a new visual concept, we search in the space of natural language descriptions to maximize the interpretation model's score (top). The chosen description can be used with the interpretation model to classify new images (bottom). Figures best viewed in color.

background knowledge from language can provide a useful scaffold for acquiring it. We specifically propose to use language as a latent *parameter space* for few-shot learning problems of all kinds, including classification, transduction and policy search. We aim to show that this linguistic parameterization produces models that are both more accurate and more interpretable than direct approaches to few-shot learning.

Like many recent frameworks for multitask- and meta-learning, our approach consists of three phases: a pretraining phase, a concept-learning phase, and an evaluation phase. Here, the product of pretraining is a language interpretation model that maps from descriptions to predictors (e.g. image classifiers or reinforcement learners). Our thesis is that language learning is a powerful, general-purpose kind of pretraining, even for tasks that do not directly involve language.

New concepts are learned by searching directly in the space of natural language strings to mini-

[1]Code and data are available at https://github.com/jacobandreas/l3.

Proceedings of NAACL-HLT 2018, pages 2166–2179
New Orleans, Louisiana, June 1 - 6, 2018.

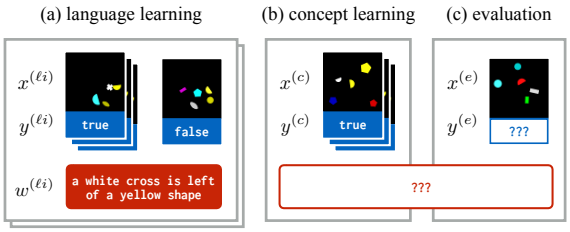


Figure 2: Formulation of the learning problem. Ultimately, we care about our model's ability to learn a concept from a small number of training examples (b) and successfully generalize it to held-out data (c). In this paper, concept learning is supported by a language learning phase (a) that makes use of natural language annotations on other learning problems. These annotations are not provided for the real target task in (b–c).

mize the loss incurred by the language interpretation model (Figure 1). Especially on tasks that require the learner to model high-level compositional structure shared by training examples, natural language hypotheses serve a threefold purpose: they make it easier to discover these compositional concepts, harder to overfit to few examples, and easier for humans to understand inferred patterns.

Our approach can be implemented using a standard kit of neural components, and is simple and general. In a variety of settings, we find that the structure imposed by a natural-language parameterization is helpful for efficient learning and exploration. The approach outperforms both multitask- and meta-learning approaches that map directly from training examples to outputs by way of a real-valued parameterization, as well as approaches that make use of natural language annotations as an additional supervisory signal rather than an explicit latent parameter. The natural language concept descriptions inferred by our approach often agree with human annotations when they are correct, and provide an interpretable debugging signal when incorrect. In short, by equipping models with the ability to "think out loud" when learning, they become both more comprehensible and more accurate.

2 Background

Suppose we wish to solve an image classification problem like the one shown in Figure 2b–c, mapping from images x to binary labels y. One straightforward way to do this is to solve a learning problem of the following form:

$$\arg\min_{\eta \in \mathbf{H}} \sum_{(x,y)} L(f(x;\eta), y)\,, \qquad (1)$$

where L is a loss function and f is a richly-parameterized class of models (e.g. convolutional networks) indexed by η (e.g. weight matrices) that map from images to labels. Given a new image x', $f(x';\eta)$ can be used to predict its label.

In the present work, we are particularly interested in *few-shot* learning problems where the number of (x, y) pairs is small—on the order of five or ten examples. Under these conditions, directly solving Equation 1 is a risky proposition—any model class powerful enough to capture the true relation between inputs and outputs is also likely to overfit. For few-shot learning to be successful, extra structure must be supplied to the learner. Existing approaches obtain this structure by either carefully structuring the hypothesis space or providing the learner with alternative training data. The approach we present in this paper combines elements of both, so we begin with a review of existing work.

(Inductive) **program synthesis** approaches (e.g. Gulwani, 2011) reduce the effective size of the hypothesis class $\mathbf{H}$ by moving the optimization problem out of the continuous space of weight vectors and into a discrete space of formal program descriptors (e.g. regular expressions or Prolog queries). Domain-specific structure like version space algebras (Lau et al., 2003) or type systems (Kitzelmann and Schmid, 2006) can be brought to bear on the search problem, and the bias inherent in the syntax of the formal language provides a strong prior. But while program synthesis techniques are powerful, they are also limited in their application: a human designer must hand-engineer the computational primitives necessary to compactly describe every learnable hypothesis. While reasonable for some applications (like string editing), this is challenging or impossible for others (like computer vision).

An alternative class of **multitask learning** approaches (Caruana, 1998) import the relevant structure from other learning problems rather than defining it manually (Figure 2a, top). Since we may not know *a priori* what set of learning problems we ultimately wish to evaluate on, it is useful to think of learning as taking places in three phases:

1. a **pretraining** (or "meta-training") phase that makes use of various different datasets i with examples $\{(x_1^{(\ell i)}, y_1^{(\ell i)}), \ldots, (x_n^{(\ell i)}, y_n^{(\ell i)})\}$ (Figure 2a)

2. a **concept-learning** phase in which the pretrained model is adapted to fit data $\{(x_1^{(c)}, y_1^{(c)}), \ldots, (x_n^{(c)}, y_n^{(c)})\}$ for a specific new task (Figure 2b)

3. an **evaluation** phase in which the learned concept is applied to a new input $x^{(e)}$ to predict $y^{(e)}$ (Figure 2c)

In these approaches, learning operates over two collections of parameters: shared parameters η and task-specific parameters θ. In pretraining, multitask approaches find:

$$\underset{\eta \in \mathbb{R}^a,\, \theta^{(\ell i)} \in \mathbb{R}^b}{\arg\min} \sum_{i,j} L\big(f(x_j^{(\ell i)};\, \eta,\, \theta^{(\ell i)}),\, y_j^{(\ell i)}\big)\,. \quad (2)$$

At concept learning time, they solve for:

$$\underset{\theta^{(c)} \in \mathbb{R}^b}{\arg\min} \sum_{j} L\big(f(x_j^{(c)};\, \eta,\, \theta^{(c)}),\, y_j^{(c)}\big) \quad (3)$$

on the new dataset, then make predictions for new inputs using $f(x^{(e)};\, \eta,\, \theta^{(c)})$.

Closely related **meta-learning** approaches (e.g. Schmidhuber, 1987; Santoro et al., 2016; Vinyals et al., 2016) make use of the same data, but collapse the inner optimization over $\theta^{(c)}$ and prediction of $y^{(e)}$ into a single learned model.

3 Learning with Language

In this work, we are interested in developing a learning method that enjoys the benefits of both approaches. In particular, we seek an intermediate language of task representations that, like in program synthesis, is both expressive and compact, but like in multitask approaches is learnable directly from training data without domain engineering. We propose to use natural language as this intermediate representation. We call our approach **learning with latent language** (L^3).

Natural language shares many structural advantages with the formal languages used in synthesis approaches: it is discrete, has a rich set of compositional operators, and comes equipped with a natural description length prior. But it also has a considerably more flexible semantics. And crucially, plentiful annotated data exists for *learning* this semantics: we cannot hand-write a computer program to recognize a *small dog*, but we can learn how to do it from image captions. More basically, the set of primitive operators available in language provides a strong prior about the kinds of abstractions that are useful for natural learning problems.

Concretely, we replace the pretraining phase above with a **language-learning** phase. We assume that at language-learning time we have access to natural-language **descriptions** $w^{(\ell i)}$ (Figure 2a, bottom). We use these w as *parameters*, in place of the task-specific parameters θ—that is, we learn a language **interpretation** model $f(x;\, \eta,\, w)$ that uses shared parameters η to turn a description w into a function from inputs to outputs. For the example in Figure 2, f might be an image rating model (Socher et al., 2014) that outputs a scalar judgment y of how well an image x matches a caption w.

Because these natural language parameters are observed at language-learning time, we need only learn the real-valued shared parameters η used for their interpretation (e.g. the weights of a neural network that implements the image rating model):

$$\underset{\eta \in \mathbb{R}^a}{\arg\min} \sum_{i,j} L\big(f(x_j^{(\ell i)};\, \eta,\, w^{(\ell i)}),\, y_j^{(\ell i)}\big)\,. \quad (4)$$

At concept-learning time, conversely, we solve only the part of the optimization problem over natural language strings:

$$\underset{w^{(c)} \in \Sigma^*}{\arg\min} \sum_{j} L\big(f(x_j^{(c)};\, \eta,\, w^{(c)}),\, y_j^{(c)}\big)\,. \quad (5)$$

This last step presents something of a challenge. When solving the corresponding optimization problem, synthesis techniques can exploit the algebraic structure of the formal language, while end-to-end learning approaches take advantage of differentiability. Here we can't do either—the language of strings is discrete, and any structure in the interpretation function is wrapped up inside the black box of f. Inspired by related techniques aimed at making synthesis more efficient (Devlin et al., 2017), we use learning to help us develop an effective optimization procedure for natural language parameters.

In particular, we simply use the language-learning datasets, consisting of pairs $(x_j^{(\ell i)}, y_j^{(\ell i)})$ and descriptions w_i, to fit a reverse **proposal** model, estimating:

$$\arg\max_\lambda \textstyle\sum_i \log q(w_i | x_1^{(\ell i)}, y_1^{(\ell i)}, \ldots, x_n^{(\ell i)}, y_n^{(\ell i)}; \lambda)$$

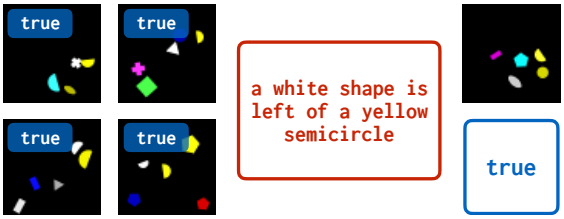


Figure 3: The few-shot image classification task. Learners are shown four positive examples of a visual concept (left) and must determine whether a fifth image matches the pattern (right). Natural language annotations are provided during language learning but must be inferred for concept learning.

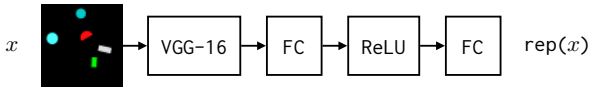
where q provides a (suitably normalized) approximation to the distribution of descriptions given task data. In the running example, this proposal distribution is essentially an image captioning model (Donahue et al., 2015). By sampling from q, we expect to obtain candidate descriptions that are likely to obtain small loss. But our ultimate inference criterion is still the true model f: at evaluation time we perform the minimization in Equation 5 by drawing a fixed number of samples, selecting the hypothesis $w^{(c)}$ that obtains the lowest loss, and using $f(x^{(e)}; \eta, w^{(c)})$ to make predictions.

What we have described so far is a generic procedure for equipping collections of related learning problems with a natural language hypothesis space. In Sections 4 and 5, we describe how this procedure can be turned into a concrete algorithm for supervised classification and sequence prediction. In Section 6, we describe how to extend these techniques to reinforcement learning.

4 Few-shot Classification

We begin by investigating whether natural language can be used to support high-dimensional few-shot classification. Our focus is on visual reasoning tasks like the one shown in Figure 3. In these problems, the learner is presented with four images, all positive examples of some visual concept like *a blue shape near a yellow triangle*, and must decide whether a fifth, held-out image matches the same concept. These kinds of reasoning problems have been well-studied in visual question answering settings (Johnson et al., 2017; Suhr et al., 2017). Our version of the problem, where the input and output feature no text data, but an explanation must be inferred, is similar to the visual reasoning problems proposed by Raven (1936) and Bongard (1968).

To apply the recipe in Section 2, we need to specify an implementation of the interpretation model f and the proposal model q. We begin by computing representations of input images x. We start with a pre-trained 16-layer VGGNet (Simonyan and Zisserman, 2014). Because spatial information is important for these tasks, we extract a feature representation from the final convolutional layer of the network. This initial featurization is passed through two fully-connected layers to form a final image representation, as follows:



We define interpretation and proposal models:[2]

$$f(x; w) = \sigma\big(\texttt{rnn-encode}(w)^\top \texttt{rep}(x)\big)$$
$$q(w \mid \{x_j\}) = \texttt{rnn-decode}\big(w \mid \tfrac{1}{n}\textstyle\sum_j \texttt{rep}(x_j)\big)$$

The interpretation model f outputs the probability that x is assigned a positive class label, and is trained to maximize log-likelihood. Because only positive examples are provided in each language learning set, the proposal model q can be defined in terms of inputs alone. Details regarding training hyperparameters, RNN implementations, etc. may be found in Appendix A.

Our evaluation aims to answer two questions. First, does the addition of language to the learning process provide any benefit over ordinary multitask or meta-learning? Second, is it specifically better to use language as a hypothesis space for concept learning rather than just an additional signal for pretraining? We use several baselines to answer these questions:

1. *Multitask*: a multitask baseline in which the definition of f above is replaced by $\sigma(\theta_i^\top \texttt{rep}(x))$ for task-specific parameters θ_i that are optimized during both pretraining and concept-learning.

2. *Meta*: a meta-learning baseline in which f is defined by $\sigma([\frac{1}{n}\sum_j \texttt{rep}(x_j)]^\top \texttt{rep}(x))$.[3]

[2] Suppressing shared parameters η and λ for clarity.

[3] Many state-of-the-art approaches to meta-learning for classification (e.g. Snell et al., 2017) are not well-defined for possibly-overlapping evaluation classes with only positive examples provideded. Here we have attempted to provide a robust implementation that is as close as possible to the other systems under evaluation.

3. *Meta+Joint*: as in *Meta*, but the pretraining objective includes an additional term for predicting q (discarded for concept learning).

We report results on a dataset derived from the ShapeWorld corpus of Kuhnle and Copestake (2017). In this dataset the held-out image matches the target concept 50% of the time. In the validation and test folds, half of learning problems feature a concept that also appears in the language learning set (but with different exemplar images), while the other half feature both new images and a new concept. Images contain two or three distractor shapes unrelated to the objects that define the target concept. Captions in this dataset were generated from DMRS representations using an HPS grammar (Copestake et al., 2016). (Our remaining experiments use human annotators.) The dataset contains a total of 9000 pretraining tasks and 1000 of each validation and test tasks. More dataset statistics are provided in Appendix B.

Results are shown in Table 1. It can be seen that L^3 provides consistent improvements over the baselines, and that these improvements are present both when identifying new instances of previously-learned concepts and when discovering new ones. Some example model predictions are shown in Figure 4. The model often succeeds in making correct predictions, even though its inferred descriptions rarely match the ground truth. Sometimes this is because of inherent ambiguity in the description language (Figure 4a), and sometimes because the model is able to rule out candidates on the basis of partial captions alone (Figure 4b, where it is sufficient to recognize that the target concept involves a *circle*). More examples are provided in Appendix C.

Model	Val (old)	Val (new)	Val	Test
Random	50	50	50	50
Multitask	64	49	57	59
Meta	63	62	62	64
Meta+Joint	63	69	66	64
L^3 (ours)	**70**	**72**	**71**	**70**
L^3 (oracle)	77	80	79	78

Table 1: Evaluation on image classification. *Val (old)* and *Val (new)* denote subsets of the validation set that contain respectively previously-used and novel visual concepts. L^3 consistently outperforms alternative learning methods based on multitask learning, meta-learning, and meta-learning jointly trained to predict descriptions (*Meta+Joint*). The last row shows results when the model is given a ground-truth concept description rather than having to infer it from examples.

5 Programming by Demonstration

Next we explore whether the same technique can be applied to tasks that involve more than binary similarity judgments. We focus on structured prediction: specifically a family of string processing tasks. In these tasks, the model is presented with examples of five strings transformed according to some rule; it must then apply an appropriate transformation to a sixth (Figure 5). Learning proceeds as in the previous section, with:

$$\texttt{rep}(x, y) = \texttt{rnn-encode}([x, y])$$
$$f(y \mid x; w) =$$
$$\quad \texttt{rnn-decode}\big(y \mid [\texttt{rnn-encode}(x), \texttt{rnn-encode}(w)]\big)$$
$$q(w \mid \{(x_j, y_j)\}) =$$
$$\quad \texttt{rnn-decode}\big(w \mid \tfrac{1}{n}\textstyle\sum_j \texttt{rep}(x_j, y_j)\big)$$

Baselines are analogous to those for classification.

While string editing tasks of the kind shown in Figure 5 are popular in both the programming by demonstration literature (Singh and Gulwani, 2012) and the semantic parsing literature (Kushman and Barzilay, 2013), we are unaware of any datasets that support both learning paradigms at the same time. We have thus created a new dataset of string editing tasks by (1) sampling random regular transducers, (2) applying these transducers to collections of dictionary words, and (3) showing the collected examples to Mechanical Turk users

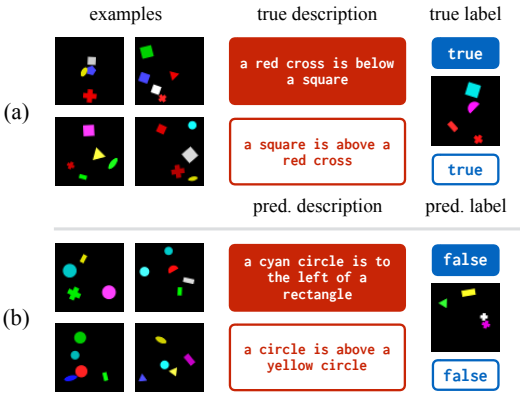


Figure 4: Example predictions for image classification. The model achieves high accuracy even though predicted descriptions rarely match the ground truth. High-level structure like the presence of certain shapes or spatial relations is consistently recovered.

Model	Val	Test
Identity	18	18
Multitask	54	50
Meta	66	62
Meta+Joint	63	59
L^3	**80**	**76**

Table 2: Results for string editing. The reported number is the percentage of cases in which the predicted string exactly matches the reference. L^3 is the best performing model; using language data for joint training rather than as a hypothesis space provides little benefit.

and asking them to provide a natural language explanation with their best guess about the underlying rule. The dataset thus features both multi-example learning problems, as well as structured and unstructured annotations for each target concept. There are 3000 tasks for language learning and 500 tasks for each of validation and testing (Appendix B). Annotations are included in the code release for this paper.

Results are shown in Table 2. In these experiments, all models that use descriptions have been trained on the natural language supplied by human annotators. While we did find that the Meta+Joint model converges considerably faster than all the others, its final performance is somewhat lower than the baseline Meta model. As before, L^3 outperforms alternative approaches for learning directly from examples with or without descriptions.

Because all of the transduction rules in this dataset were generated from known formal descriptors, these tasks provide an opportunity to perform additional analysis comparing natural language to more structured forms of annotation (since we have access to ground-truth regular expressions) and more conventional synthesis-based methods (since we have access to a ground-truth regular expression execution engine). We additionally investigate the effect of the number of

Figure 5: Example string editing task. Learners are presented with five examples of strings transformed according to some rule (left), and must apply an appropriate transformation to a sixth string (right). Language-learning annotations (center) may take the form of either natural language or regular expressions.

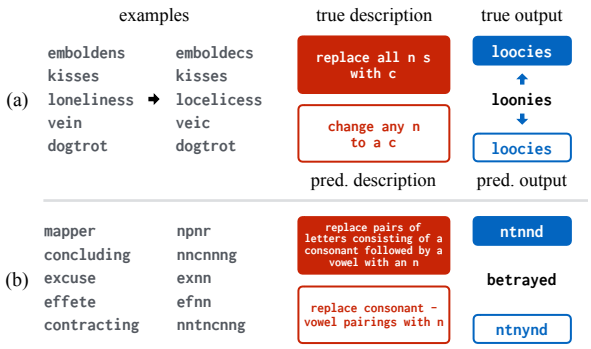


Figure 6: Example predictions for string editing.

samples drawn from the proposal model. These results are shown in Table 3.

A few interesting facts stand out. Under the ordinary evaluation condition (with no ground-truth annotations provided), language-learning with natural language data is actually better than language-learning with regular expressions. This might be because the extra diversity helps the model determine the relevant axes of variation and avoid overfitting to individual strings. Allowing the model to do its own inference is also better than providing ground-truth natural language descriptions, suggesting that it is actually better at generalizing from the relevant concepts than our human annotators (who occasionally write things like *I have no idea* for the inferred rule). Unsurprisingly, with ground truth REs (which unlike the human data are always correct) we can do better than any of the models that require inference. Coupling our inference procedure with an oracle RE evaluator, we essentially recover the synthesis-based approach of Devlin et al. (2017). Our findings are consistent with theirs: when an exact execution engine is available, there is no reason not to use it. But we can get almost 90% of the way there

Annotations	Samples		Oracle	
	1	100	Ann.	Eval.
None (Meta)	*66*	–	–	–
Natural language	66	*80*	75	–
Regular expressions	60	76	88	90

Table 3: Inference and representation experiments for string editing. Italicized numbers correspond to entries in Table 2. Allowing the model to use multiple samples rather than the 1-best decoder output substantially improves performance. The full model does better with inferred natural language descriptions than either regular expressions or ground-truth natural language.

with an execution model learned from scratch. Examples of model behavior are shown in Figure 6; more may be found in Appendix D.

6 Policy Search

The previous two sections examined supervised settings where the learning signal comes from few examples but is readily accessible. In this section, we move to a set of reinforcement learning problems, where the learning signal is instead sparse and time-consuming to obtain. We evaluate on a collection of 2-D treasure hunting tasks. These tasks require the agent to discover a rule that determines the location of buried treasure in a large collection of environments of the kind shown in Figure 7. To recover the treasure, the agent must navigate (while avoiding water) to its goal location, then perform a `DIG` action. At this point the episode ends; if the treasure is located in the agent's current position, it receives a reward, otherwise it does not. In every task, the treasure has consistently been buried at a fixed position relative to some landmark (in Figure 7 a heart). Both the offset and the identity of the target landmark are unknown to the agent, and the location of the landmark varies across maps. Indeed, there is nothing about the agent's observations or action space to suggest that landmarks and offsets are even the relevant axes of variation across tasks: only the language reveals this structure.

The interaction between language and learning in these tasks is rather different from the supervised settings. In the supervised case, language serves mostly as a guard against overfitting, and

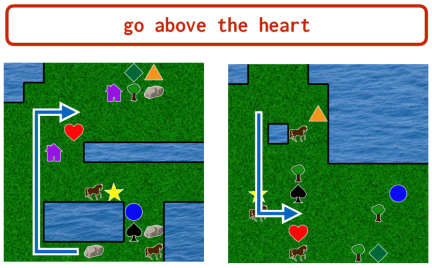
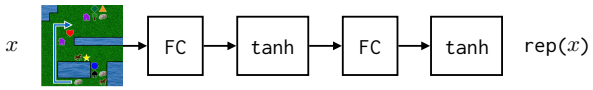


Figure 7: Example treasure hunting task: the agent is placed in a random environment and must collect a reward that has been hidden at a consistent offset with respect to some landmark. At language-learning time only, natural language instructions and expert policies are provided. The agent must both learn primitive navigation skills, like avoiding water, as well as the high-level structure of the reward functions for this domain.

can be generated conditioned on a set of pre-provided concept-learning observations. Here, agents are free to interact with the environment as much as they need, but receive observations only during interaction. Thus our goal here will be to build agents that can *adapt quickly* to new environments, rather than requiring them to immediately perform well on held-out data.

Why should we expect L^3 to help in this setting? In reinforcement learning, we typically encourage our models to explore by injecting randomness into either the agent's action space or its underlying parameterization. But most random policies exhibit nonsensical behaviors; as a result, it is inefficient both to sample in the space of network weights and to perform policy optimization from a random starting point. Our hope is that when parameters are chosen from within a structured family, a stochastic search in this structured space will only ever consider behaviors corresponding to a reasonable final policy, and in this way discover good behavior faster than ordinary RL.

Here the interpretation model f describes a policy that chooses actions conditioned on the current environment state and a linguistic parameterization. As the agent initially has no observations at all, we simply design the proposal model to generate unconditional samples from a prior over descriptions. Taking x to be an agent's current observation of the environment state, we define a state representation network and models:



$$f(a \mid x; w) \propto \texttt{rnn-encode}(w)^\top W_a \, \texttt{rep}(x)$$
$$q(w) = \texttt{rnn-decode}(w)$$

This parameterization assumes a discrete action space, and assigns to each action a probability proportional to a bilinear function of the encoded description and world state. f is an instruction following model of a kind well-studied in natural language processing (Branavan et al., 2009); the proposal model allows it to generate its own instructions without external direction. To learn, we sample a fixed number of descriptions w from q. For each description, we sample multiple rollouts of the policy it induces to obtain an estimate of its average reward. Finally, we take the highest-scoring description and fine-tune its induced policy.

At language-learning time, we assume access to both natural language descriptions of these tar-

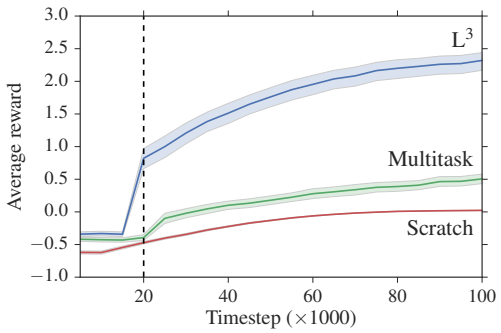


Figure 8: Treasure hunting reward obtained by each learning algorithm across multiple evaluation environments, after language learning has already taken place (bands show 95% confidence intervals for mean performance). *Multitask* learns an embedding for each task, while *Scratch* trains on every task individually. L^3 rapidly discovers high-scoring policies in most environments. Dashed line indicates the end of the concept-learning phase; subsequent performance comes from fine-tuning. The max reward for this task is 3.

get locations provided by human annotators, as well as expert policies for navigating to the location of the treasure. The multitask model we compare to replaces these descriptions with trainable task embeddings.[4] The learner is trained from task-specific expert policies using DAgger (Ross et al., 2011) during the language-learning phase, and adapts to individual environments using "vanilla" policy gradient (Williams, 1992) during the concept learning phase.

The environment implementation and linguistic annotations are in this case adapted from a natural language navigation dataset originally introduced by Janner et al. (2017). In our version of the problem (Figure 7), the agent begins each episode in a random position on a randomly-chosen map and must attempt to obtain the treasure. Relational concepts describing target locations are reused between language learning and concept-learning phases, but the environments themselves are distinct. For language learning the agent has access to 250 tasks, and is evaluated on an additional 50.

Averaged learning curves for held-out tasks are shown in Figure 8. As expected, reward for the L^3 model remains low during the initial exploration period, but once a description is chosen the score improves rapidly. Immediately L^3 achieves better reward than the multitask baseline, though it is not perfect; this suggests that the interpretation model is somewhat overfit to the pretraining environments. After fine-tuning even better results are rapidly obtained. Example rollouts are visualized in Appendix E. These results show that the model has used the structure provided by language to *learn* a better representation space for policies—one that facilitates sampling from a distribution over interesting and meaningful behaviors.

[4] In RL, the contribution of L^3 is orthogonal to that of meta-learning—one could use a technique like RL^2 (Duan et al., 2016) to generate candidate descriptions more efficiently, or MAML (Finn et al., 2017) rather than zero-shot reward as the training criterion for the interpretation model.

7 Other Related Work

This is the first approach we are aware of to frame a general learning problem as optimization over a space of natural language strings. However, many closely related ideas have been explored in the literature. String-valued latent variables are widely used in language processing tasks ranging from morphological analysis (Dreyer and Eisner, 2009) to sentence compression (Miao and Blunsom, 2016). Natural language annotations have been used in conjunction with training examples to guide the discovery of logical descriptions of concepts (Ling et al., 2017; Srivastava et al., 2017), and used as an auxiliary loss for training (Frome et al., 2013), analogously to the Meta+Joint baseline in this paper. Structured language-like annotations have been used to improve learning of generalizable structured policies (Oh et al., 2017; Andreas et al., 2017; Denil et al., 2017). Finally, natural language instructions available at *concept-learning* time (rather than language-learning time) have been used to provide side information to reinforcement learners about high-level strategy (Branavan et al., 2011), environments (Narasimhan et al., 2017) and exploration (Harrison et al., 2017).

8 Conclusion

We have presented an approach for learning in a space parameterized by natural language. Using simple models for representation and search in this space, we demonstrated that our approach outperforms standard baselines on classification, structured prediction and reinforcement learning tasks. We believe that these results suggest the following general conclusions:

Language encourages compositional generalization. Standard deep learning architectures are good at recognizing new instances of familiar

concepts, but not always at generalizing to new ones. By forcing decisions to pass through a linguistic bottleneck in which the underlying compositional structure of concepts is explicitly expressed, stronger generalization becomes possible.

Language simplifies structured exploration. Natural language scaffolding provides dramatic advantages in problems like reinforcement learning that require exploration: models with latent linguistic parameterizations can limit exploration to a class of behaviors that are likely *a priori* to be goal-directed and interpretable.

And generally, *language can help learning*. In multitask settings, it can even improve learning on tasks for which no language data is available at training or test time. While some of these advantages are also provided by techniques built on top of formal languages, natural language is at once more expressive and easier to obtain than formal supervision. We believe this work hints at broader opportunities for using naturally-occurring language data to improve machine learning for tasks of all kinds.

Acknowledgments

JA is supported by a Facebook graduate fellowship.

References

Jacob Andreas, Dan Klein, and Sergey Levine. 2017. Modular multitask reinforcement learning with policy sketches. In *Proceedings of the International Conference on Machine Learning*.

Mikhail Moiseevich Bongard. 1968. The recognition problem. Technical report.

S.R.K. Branavan, Harr Chen, Luke S. Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

S.R.K. Branavan, David Silver, and Regina Barzilay. 2011. Learning to win by reading manuals in a Monte-Carlo framework. In *Proceedings of the Human Language Technology Conference of the Association for Computational Linguistics*.

Rich Caruana. 1998. Multitask learning. In *Learning to learn*, Springer.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* .

Ann A Copestake, Guy Emerson, Michael Wayne Goodman, Matic Horvat, Alexander Kuhnle, and Ewa Muszynska. 2016. Resources for building applications with dependency minimal recursion semantics. In *Language Resources and Computation*.

Misha Denil, Sergio Gómez Colmenarejo, Serkan Cabi, David Saxton, and Nando de Freitas. 2017. Programmable agents. *arXiv preprint arXiv:1706.06383* .

Jacob Devlin, Jonathan Uesato, Surya Bhupatiraju, Rishabh Singh, Abdel-rahman Mohamed, and Pushmeet Kohli. 2017. RobustFill: Neural program learning under noisy I/O. In *Proceedings of the International Conference on Machine Learning*.

Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*.

Markus Dreyer and Jason Eisner. 2009. Graphical models over multiple strings. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. 2016. RL^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779* .

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the International Conference on Machine Learning*.

Andrea Frome, Greg Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, MarcAurelio Ranzato, and Tomas Mikolov. 2013. Devise: A deep visual-semantic embedding model. In *Advances in Neural Information Processing Systems*.

Alison Gopnik and Andrew Meltzoff. 1987. The development of categorization in the second year and its relation to other cognitive and linguistic developments. *Child Development* .

Sumit Gulwani. 2011. Automating string processing in spreadsheets using input-output examples. *ACM SIGPLAN Notices* 46(1).

Brent Harrison, Upol Ehsan, and Mark O Riedl. 2017. Guiding reinforcement learning exploration using natural language. *arXiv preprint arXiv:1707.08616* .

Michael Janner, Karthik Narasimhan, and Regina Barzilay. 2017. Representation learning for grounded spatial reasoning. *Transactions of the Association for Computational Linguistics* .

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. *arXiv preprint arXiv:1606.03622* .

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. *Proceedings of the Conference on Computer Vision and Pattern Recognition* .

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

Emanuel Kitzelmann and Ute Schmid. 2006. Inductive synthesis of functional programs: An explanation based generalization approach. *Journal of Machine Learning Research* 7.

Alexander Kuhnle and Ann Copestake. 2017. Shapeworld-a new test methodology for multimodal language understanding. *arXiv preprint arXiv:1704.04517* .

Nate Kushman and Regina Barzilay. 2013. Using semantic unification to generate regular expressions from natural language. In *Proceedings of the Annual Meeting of the North American Chapter of the Association for Computational Linguistics*.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in ccg grammar induction for semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Tessa Lau, Steven A Wolfman, Pedro Domingos, and Daniel S Weld. 2003. Programming by demonstration using version space algebra. *Machine Learning* 53(1).

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Karthik Narasimhan, Regina Barzilay, and Tommi Jaakkola. 2017. Deep transfer in reinforcement learning by language grounding. *arXiv preprint arXiv:1708.00133* .

Junhyuk Oh, Satinder Singh, Honglak Lee, and Pushmeet Kohli. 2017. Zero-shot task generalization with multi-task deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning*.

John C Raven. 1936. Mental tests used in genetic studies: The performance of related individuals on tests mainly educative and mainly reproductive. *Unpublished masters thesis, University of London* .

Stéphane Ross, Geoffrey J Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-learning with memory-augmented neural networks. In *Proceedings of the International Conference on Machine Learning*.

Jurgen Schmidhuber. 1987. Evolutionary principles in self-referential learning. *On learning how to learn. Diploma thesis, Institut f. Informatik, Tech. Univ. Munich* .

K Simonyan and A Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arxiv:1409.1556* .

Rishabh Singh and Sumit Gulwani. 2012. Learning semantic string transformations from examples. In *International Conference on Very Large Databases*.

Jake Snell, Kevin Swersky, and Richard S Zemel. 2017. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175* .

Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics* 2.

Shashank Srivastava, Igor Labutov, and Tom Mitchell. 2017. Joint concept learning and semantic parsing from natural language explanations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Alane Suhr, Mike Lewis, James Yeh, and Yoav Artzi. 2017. A corpus of natural language for visual reasoning. In *55th Annual Meeting of the Association for Computational Linguistics, ACL*.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4).

A Model and Training Details

In all models, RNN encoders and decoders use gated recurrent units (Cho et al., 2014).

Few-shot classification Models are trained with the ADAM optimizer (Kingma and Ba, 2015) with a step size of 0.0001 and batch size of 100. The number of pretraining iterations is tuned based on subsequent concept-learning performance on the development set. Neural network hidden states, task parameters, and word vectors are all of size 512. 10 hypotheses are sampled during for each evaluation task in the concept-learning phase.

Programming by demonstration Training as in the classification task, but with a step size of 0.001. Hidden states are of size 512, task parameters of size 128 and word vectors of size 32. 100 hypotheses are sampled for concept learning.

Policy search DAgger (Ross et al., 2011) is used for pre-training and vanilla policy gradient (Williams, 1992) for concept learning. Both learning algorithms use ADAM with a step size of 0.001 and a batch size of 5000 samples. For imitation learning, rollouts are obtained from the expert policy on a schedule with probability 0.95^t (for t the current epoch). For reinforcement learning, a discount of 0.9 is used. Because this dataset contains no development data, pretraining is run until performance on the pretraining tasks reaches a plateau. Hidden states and task embeddings are of size 64. 100 hypotheses are sampled for concept learning, and 1000 episodes (divided evenly among samples) are used to estimate hypothesis quality before fine-tuning.

B Dataset Information

ShapeWorld This is the only fully-synthetic dataset used in our experiments. Each scene features 4 or 5 non-overlapping entities. Descriptions refer to spatial relationships between pairs of entities identified by shape, color, or both. There are 8 colors and 8 shapes. The total vocabulary size is only 30 words, but the dataset contains 2643 distinct captions. Descriptions are on average 12.0 words long.

Regular expressions Annotations were collected from Mechanical Turk users. Each user was presented with the same task as the learner in this paper: they observed five strings being transformed, and had to predict how to transform a sixth. Only after they correctly generated the held-out word were they asked for a description of the rule. Workers were additionally presented with hints like "look at the beginning of the word" or "look at the vowels". Descriptions are automatically preprocessed to strip punctuation and ensure that every character literal appears as a single token.

The regular expression data has a vocabulary of 1015 rules and a total of 1986 distinct descriptions. Descriptions are on average 12.3 words in length but as long as 46 words in some cases.

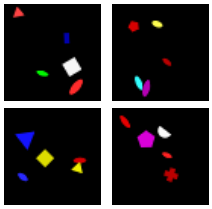
Navigation The data used was obtained from Janner et al. (2017). We created our own variant of the dataset containing collections of related tasks. Beginning with the "local" tasks in the dataset, we generated alternative goal positions at fixed offsets from landmarks as described in the main section of this paper. Natural-language descriptions were selected for each task collection from the human annotations provided with the dataset. The vocabulary size is 74 and the number of distinct hints 446. The original action space for the environment is also modified slightly: rather than simply reaching the goal cell (achieved with reasonably high frequency by a policy that takes random moves), we require the agent to commit to an individual goal cell and end the episode with a special `DIG` action.

Data augmentation Due to their comparatively small size, a data augmentation scheme (Jia and Liang, 2016) is employed for the regular expression and navigation datasets. In particular, wherever a description contains a recognizable entity name (i.e. a character literal or a landmark name), a description template is extracted. These templates are then randomly swapped in at training time on other examples with the same high-level semantics. For example, the description *replace first b with e* is abstracted to *replace first CHAR1 with CHAR2*, and can subsequently be specialized to, e.g., *replace first c with d*. This templating is easy to implement because we have access to ground-truth structured concept representations at training time. If these were not available it would be straightforward to employ an automatic template induction system (Kwiatkowski et al., 2011) instead.

C Examples: ShapeWorld

(Examples in this and the following appendices were not cherry-picked.)

Positive examples:



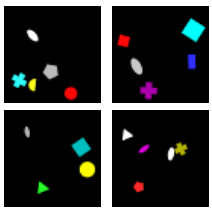
True description:
a red ellipse is to the right of an ellipse

Inferred description:
a red shape is to the right of a red semicircle

Input:

True label: true

Pred. label: true

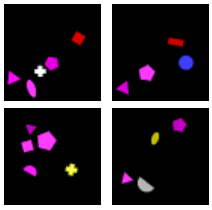


a shape is below a white ellipse

a white shape is to the left of a yellow ellipse

false

true

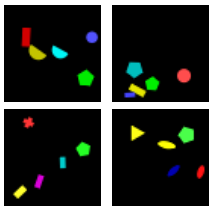


a magenta triangle is to the left of a magenta pentagon

a magenta triangle is to the left of a pentagon

true

true

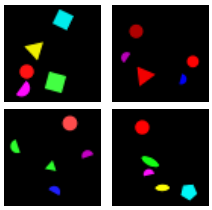


a green pentagon is to the right of a yellow shape

a green shape is to the right of a red semicircle

false

false

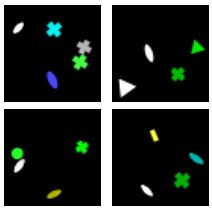


a red circle is above a magenta semicircle

a green triangle is above a red circle

false

true



a white ellipse is to the left of a green cross

a green cross is to the right of a white ellipse

true

true

D Examples: Regular Expressions

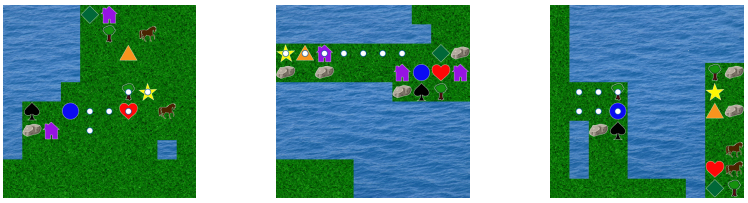
Example in:	Example out:	Human description:		True out:
mediaeval paneling wafer conventions handsprings	ilediaeval ilaneling ilafer ilonventions ilandsprings	leading consonant si replaced with i l Inferred description: first consonant of a word is replaced with i l	Input: chaser	ilhaser Pred. out: ilhaser
uptakes pouching embroidery rebelliousness stoplight	uptakes punuching embrunidery rebelliunusness stunplight	replace every o with u n change all o to u n	regulation	regulatiunn regulatinun
fluffiest kidnappers matting griping disagreements	fluffiest kidnappers eeatting griping disagreeeeents	the leter m is replaced by ee change every m to ee	chartering	chartering chartering
clandestine limning homes lifeblood inflates	clandqtine limning homq lifqlood inflatq	e where e appears , replace it and the following letter with q	gratuity	gratuity gratuity
fruitlessly sandier washers revelries dewlaps	fruitlessly sandier washemu revelrimu dewlamu	if the word ends with an s , replace the last two letters with m u change last to m u if consonant	prompters	promptemu promptemu
ladylike flintlocks student surtaxes bedecks	ladylike flintlocknl studennl surtaxenl bedecknl	ending consonant is replaced with n l drop last two and add n l	initials	initialnl initialnl
porringer puddling synagog curtseying monsieur	porringeer puddlinge synageoge curtseyinge monsieur	add e next to letter g when a letter is preceded by a g , e is added after that letter	rag	rage rage
trivializes tried tearfully hospitalize patronizing	trivializes tried gxarfully gxspitalize gxtronizing	replace the 1st 2 letters of the word with a g x if the word begins with a consonant then a vowel if the second letter is a vowel , replace the first two letters with g x	landlords	gxndlords gxndlords
microseconds antiviral flintlock appreciable stricter	microsecnyr antiviral flintloyr appreciabyr stricter	replace consonants with y r the last two letters are replaced by y r	exertion	exertion exertiyr

E Examples: Navigation

White breadcrumbs show the path taken by the agent.

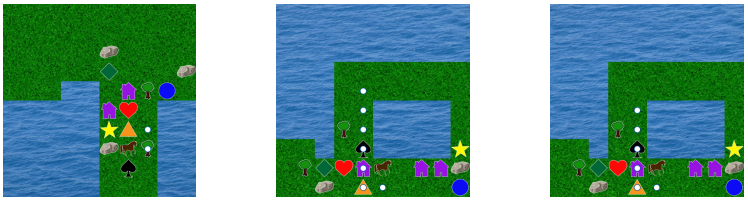
Human description:
`move to the star`

Inferred description:
`reach the star cell`



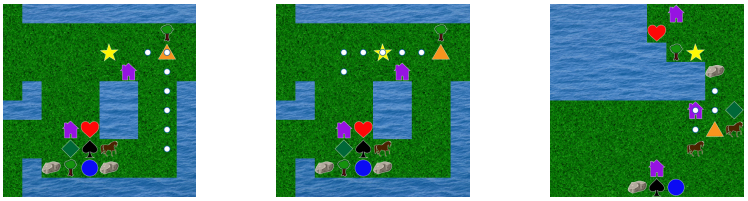
`reach square one right of triangle`

`reach cell to the right of the triangle`



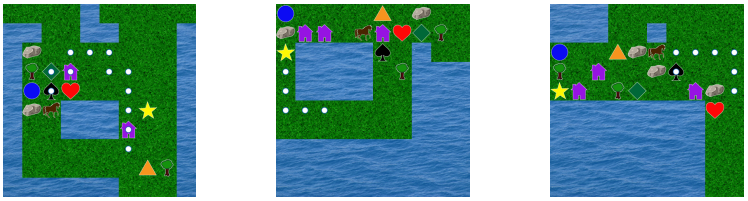
`reach cell on left of triangle`

`reach square left of triangle`



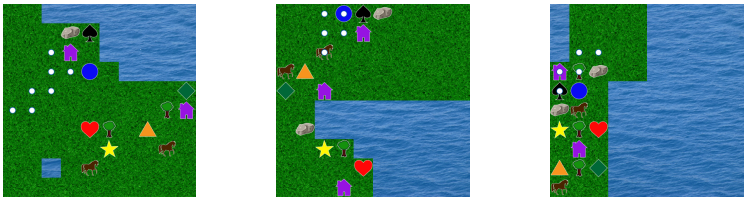
`reach spade`

`go to the spade`



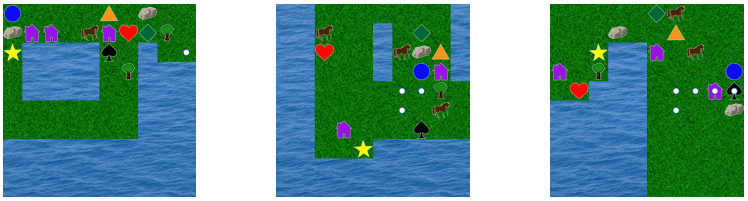
`left of the circle`

`go to the cell to the left of the circle`



`reach cell below the circle`

`reach cell below circle`



Object Counts! Bringing Explicit Detections Back into Image Captioning

Josiah Wang, Pranava Madhyastha and **Lucia Specia**
Department of Computer Science
University of Sheffield, UK
{j.k.wang, p.madhyastha, l.specia}@sheffield.ac.uk

Abstract

The use of explicit object detectors as an intermediate step to image captioning – which used to constitute an essential stage in early work – is often bypassed in the currently dominant end-to-end approaches, where the language model is conditioned directly on a mid-level image embedding. We argue that explicit detections provide rich semantic information, and can thus be used as an interpretable representation to better understand why end-to-end image captioning systems work well. We provide an in-depth analysis of end-to-end image captioning by exploring a variety of cues that can be derived from such object detections. Our study reveals that end-to-end image captioning systems rely on matching image representations to generate captions, and that encoding the frequency, size and position of objects are complementary and all play a role in forming a good image representation. It also reveals that different object categories contribute in different ways towards image captioning.

1 Introduction

Image captioning (IC), or image description generation, is the task of automatically generating a sentential textual description for a given image. Early work on IC tackled the task by first running object detectors on the image and then using the resulting explicit detections as input to generate a novel textual description, e.g. (Kulkarni et al., 2011; Yang et al., 2011). With the advent of sequence-to-sequence approaches to IC, e.g. (Karpathy and Fei-Fei, 2015; Vinyals et al., 2015), coupled with the availability of large image description datasets, the performance of IC systems showed marked improvement, at least according to automatic evaluation metrics like Meteor (Denkowski and Lavie, 2014) and CIDEr (Vedantam et al., 2015).

The currently dominant neural-based IC systems are often trained end-to-end, using parallel (image, caption) datasets. Such systems are essentially sequential language models conditioned directly on some mid-level image features, such as an image embedding extracted from a pre-trained Convolutional Neural Network (CNN). Thus, they bypass the explicit detection phase of previous methods and instead generate captions *directly* from image features. Despite significant progress, it remains unclear why such systems work. A major problem with these IC systems is that they are less interpretable than conventional pipelined methods which use explicit detections.

We believe that it is timely to again start exploring the use of explicit object detections for image captioning. Explicit detections offer rich semantic information, which can be used to model the entities in the image as well as their interactions, and can be used to better understand image captioning.

Recent work (Yin and Ordonez, 2017) showed that conditioning an end-to-end IC model on visual representations that implicitly encode object details yields reasonably good captions. Nevertheless, it is still unclear *why* this works, and *what* aspects of the representation allow for such a good performance. In this paper, we study end-to-end IC in the context of explicit detections (Figure 1) by exploring a variety of cues that can be derived from such detections to determine *what* information from such representations helps image captioning, and *why*. To our best knowledge, our work is the first experimental analysis of end-to-end IC frameworks that uses object-level information that is highly interpretable as a tool for understanding such systems. Our main contributions are as follows:

1. We provide an in-depth analysis of the performance of end-to-end IC using a simple, yet effective 'bag of objects' representation that

Proceedings of NAACL-HLT 2018, pages 2180–2193
New Orleans, Louisiana, June 1 - 6, 2018.

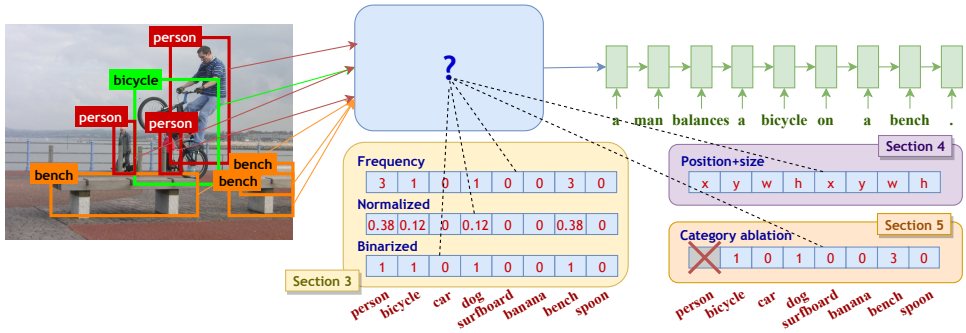


Figure 1: Using explicit detections as an intermediate step towards end-to-end image captioning. The question we investigate is what information can we extract from explicit detections that can be useful for image captioning.

is interpretable, and generates good captions despite being low-dimensional and highly sparse (Section 3).

2. We investigate whether other spatial cues can be used to provide information complementary to frequency counts (Section 3).
3. We study the effect of incorporating different spatial information of individual object instances from explicit detections (Section 4).
4. We analyze the contribution of the categories in representations for IC by ablating individual categories from them (Section 5).

Our hypothesis is that there are important components derived from explicit detections that can be used to effectively inform IC. Our study confirms our hypothesis, and that features such as the frequency, size and position of objects all play a role in forming a good image representation to match their corresponding representations in the training set. Our findings also show that different categories contribute differently to IC, and this partly depends on how likely they are to be mentioned in the caption given that they are depicted in the image. The results of our investigation will help further work towards more interpretable image captioning.

2 Related work

Early work on IC apply object detectors explicitly on an image as a first step to identify entities present in the image, and then use these detected objects as input to an image caption generator. The caption generator typically first performs content selection (selecting a subset of objects to be described) and generates an intermediate representation (e.g. semantic tuples or abstract trees), and then performs surface realization using rules, templates, n-grams or a maximum entropy language model. The main body of work uses object detectors for 20 pre-specified PASCAL VOC (Visual Object Classes) (Everingham et al., 2015) (Yang et al., 2011; Kulkarni et al., 2011; Li et al., 2011; Mitchell et al., 2012), builds a detector inferred from captions (Fang et al., 2015), or assumes gold standard annotations are available (Elliott and Keller, 2013; Yatskar et al., 2014).

Currently, deep learning end-to-end approaches dominate IC work (Donahue et al., 2015; Karpathy and Fei-Fei, 2015; Vinyals et al., 2015). Such approaches do not use an explicit detection step, but instead use a 'global' image embedding as input (generally a CNN) and learn a language model (generally an LSTM) conditioned on this input. Thus, they are trained to learn image caption generation directly from a parallel image–caption dataset. The advantage is that no firm decisions need to be made about object categories. However, such approaches are hard to interpret and are dataset dependent (Vinyals et al., 2017).

Some recent work use object-level semantics for end-to-end IC (Gan et al., 2017; Wu et al., 2016; You et al., 2016). Such systems represent images as predictions of semantic concepts occurring in the image. These predictions, however, are at a global, *image* level ("does this image contain a chair?"), rather than at object *instance* level ("there is a big chair at position x"). In addition, most previous work regard surface-level terms extracted directly from captions as 'objects', while we use off-the-shelf predefined object categories which have a looser connection between the image and the caption (e.g. objects can be described in captions using different terms, depicted objects might not be mentioned in captions, and captions might mention objects that are not depicted).

Yin and Ordonez (2017) propose conditioning an end-to-end IC model on information derived from explicit detections. They implicitly encode the category label, position and size of object instances as an 'object-layout' LSTM and condition the language model on the final hidden state of this LSTM, and produce reasonably good image captions based only on those cues, without the direct use of images. Our work is different in that we feed information from explicit object detections *directly* to the language model in contrast to an object-layout LSTM which abstracts away such information, thereby retaining the *interpretability* of the input image representation. This gives us more control over the image representation which is simply encoded as a bag of categorical variables.

There is also recent work applying attention-based models (Xu et al., 2015) on explicit object proposals (Anderson et al., 2018; Li et al., 2017), which may capture object-level information from the attention mechanism. However, attention-based models require object information in the form of vectors, whereas our models use information of objects as categorical variables which allow for easy manipulation but are not compatible with the standard attention-based models. The model that we use, *under similar conditions* (i.e. under similar parametric settings), is comparable to the state-of-the-art models.

3 Bag of objects

We base our experiments on the MS COCO dataset (Lin et al., 2014). From our preliminary experiments, we found that a simple *bag of object categories* used as an image representation for end-to-end IC led to good scores according to automatic metrics, comparable to and perhaps even higher than those using CNN embeddings. This is surprising given that this bag of objects vector is *low-dimensional* (each element represents the frequency of one of 80 COCO categories) and *sparse* (mainly zeros, as only a few object categories tend to occur in a given image). In simple terms, it appears that the IC model can generate a reasonable caption by merely knowing what is in the image, e.g. that there are three *person*s, three *benches* and a *bicycle* in Figure 1.

This observation raises the following questions. What is it in this simple bag of objects representation that contributes to the surprisingly high performance on IC? Does it lie in the frequency counts? Or the choice of categories themselves? It is also worth noting that the image captions in COCO were crowd-sourced *independent* of the COCO object annotations, i.e. image captions were written based only on the image, without object-level annotations. The words used in the captions thus do not correspond directly to the 80 COCO categories (e.g. a *cup* may not be mentioned in a description even though it is present in the image, and vice versa, i.e. objects described in the caption may not correspond to any of the categories).

In order to shed some light into what makes bag of object categories representations work so well for IC, we first investigate whether the *frequency counts* is the main contributor. We then proceed to studying what else can be exploited from explicit object detections to improve on the bag of objects model, for example the size of object instances. We also perform an analysis on these representations to gain more insights into why the bag of objects model performs well.

3.1 Image captioning model

Our implementation is based on the end-to-end approach of Karpathy and Fei-Fei (2015). We use an LSTM (Hochreiter and Schmidhuber, 1997) language model as described in Zaremba et al. (2014). To condition the image information, we first perform a linear projection of the image representation followed by a non-linearity:

$$x = \sigma(W \cdot I_m) \tag{1}$$

where $I_m \in \mathcal{R}^d$ is the d-dimensional initial image representation, $W \in \mathcal{R}^{n \times d}$ is the linear transformation matrix, σ is the non-linearity. We use Exponential Linear Units (Clevert et al., 2016) as the non-linear activation in all our experiments. We initialize the LSTM-based caption generator with the projected image representation, x.

Training and inference. The caption generator is trained to generate sentences conditioned on x. We train the model by minimizing the cross-entropy, i.e. the sentence-level loss corresponds to the sum of the negative log likelihood of the correct word at each time step:

$$\Pr(S|x;\theta) = \sum_t \log(\Pr(w_t|w_{t-1}..w_0;x)) \tag{2}$$

where $\Pr(S|x;\theta)$ is the sentence-level loss con-

ditioned on the image feature x and $\Pr(w_t)$ is the probability of the word at time step t. This is trained with standard teacher forcing as described in Sutskever et al. (2014), where the correct word information is fed to the next state in the LSTM.

Inference is usually performed using approximate techniques like beam search and sampling methods. As we are mainly interested in studying different image representations, we focus on the language output that the models can most confidently produce. In order to isolate any other variables from the experiments, we generate captions using a greedy $\arg\max$ approach. We use a 2-layer LSTM with 128-dimensional word embeddings and 256-dimensional hidden dimensions. As training vocabulary we retain only words that appear at least twice. We provide details about hyperparameters and tuning in Appendix A.

3.2 Visual representations

The first part of our experiments studies the role of *frequency counts* of the 80-dimensional bag of objects representation. We explore the effects of using the following variants of the bag of objects representation: (i) **Frequency:** The number of instances per category; (ii) **Normalized:** The frequency counts normalized such that the vector sums to 1. This represents the *proportion* of object occurrences in the image; (iii) **Binarized:** An object category's entry is set to 1 if at least one instance of the category occurs, and 0 otherwise.

Berg et al. (2012) explore various factors that dictate what objects are mentioned in image descriptions, and found that object size and its position relative to the image centre are important. Inspired by these findings, we explore alternative representations based on these cues: (i) **Object size:** The area of the region provided by COCO, normalized by image size; we encode the largest object if multiple objects occur for the same category (max pooling). (ii) **Object distance:** The Euclidean distance from the object bounding box centre to the image centre, normalized by image size; we encode the object closest to the centre if multiple instances occur (min pooling). We also explore **concatenating** these features to study their complementarity.

Finally, we study the effects of **removing information** from the bag of objects representation. More specifically, we compare the results of retaining only a certain number of object instances in the *frequency*-based bag of objects representation, rather than representing an image with all objects present. We experiment with retaining only the frequency counts for one object category and 25%, 50%, and 75% of object categories; the remaining entries in the vector are set to zero. The object categories to be retained are selected, per image: (i) randomly; (ii) by the $N\%$ most frequent categories of the image; (iii) by the $N\%$ largest categories of the image; (iv) by the $N\%$ categories closest to the centre of the image.

We performed these evaluations based on (i) ground truth COCO annotations and (ii) the output of an off-the-shelf object detector (Redmon and Farhadi, 2017) trained on 80 COCO categories. With ground truth annotations we can isolate issues stemming from incorrect detections.

Representation	GT	Detect
CNN (ResNet-152 POOL5)	-	0.749
Frequency	0.807	0.752
Normalized	0.762	0.703
Binarized	0.751	0.703
Object min distance	0.759	0.691
Object max size	0.793	0.725
Obj max size + Obj min distance	0.799	0.743
Frequency + Obj min distance	0.830	0.769
Frequency + Obj max size	0.836	0.769
All three features	0.849	0.743

Table 1: CIDEr scores for image captioning using bag of objects variants as visual representations. We compare the results of using ground truth annotations (**GT**) and the output of a detector (**Detect**). As comparison we also provide, in the first row, the results of using a ResNet-152 POOL5 CNN image embedding with our implementation of an end-to-end IC system.

3.3 Experiments

We train our models on the full COCO training set, and use the standard, publicly available splits[1] of the validation set as in previous work (Karpathy and Fei-Fei, 2015) for validation and testing (5,000 images each). We use CIDEr (Vedantam et al., 2015) – the official metric for COCO – as our evaluation metrics for all experiments. For completeness, we present scores for other common IC metrics in Appendix B.

Table 1 shows the CIDEr scores of IC systems using variants of the bag of objects representation, for both ground truth annotations and

[1] http://cs.stanford.edu/people/karpathy/deepimagesent

Feature vs. Pooling	Min	Max	Mean
Obj. Size	0.748	0.793	0.789
Obj. Distance	0.759	0.768	0.740

Table 2: CIDEr scores for captioning comparing the use of min, max or average pooling of either object size or distance features, using ground truth annotations.

the output of an object detector. Compared to a pure CNN embedding (ResNet-152 POOL5), our object-based representations show higher (for ground truth annotations) or comparable CIDEr scores (for detectors). Our first observation is that frequency **counts** are essential to IC. Using normalized counts as a representation gives poorer results, which intuitively makes sense: An image with 20 cars and 10 people is significantly different from an image with two cars and one person. Using binarized counts (presence or absence) brings the score further down. This is to be expected: An image with one person is very different from one with 10 people.

Using spatial information (size or distance) also proved useful. Encoding the object size in place of frequency gave reasonably better results over using object distance from the image centre. We can conclude that the size and centrality of objects are important factors for captioning, with object size being more informative than position.

We also experimented with different methods for aggregating multiple instances of the same category, in addition to choosing the biggest instance and the instance closest to the image centre. For example, choosing the *smallest* instance (min pooling) or the instance *furthest* away from the image centre (max pooling), or just averaging them (mean pooling). Table 2 shows the results. For object size, the findings are as expected: Smaller object instances are less important for IC, although averaging them works comparably well. Surprisingly, in the case of distance, using the object *furthest* from the image centre actually gave slightly better results than the one closest. Further inspection revealed that aggregating instances is not effective in some cases. We found that the positional information (and interaction with other objects) captured by the object further away may sometimes represent the semantics of the image better than the object in the centre of the image. For example, in Figure 2, encoding only the position of the person in the middle will result in the

Obj. min distance:
• a man in a kitchen preparing food in a kitchen .
Obj. max distance:
• a group of people standing around a kitchen counter .

Figure 2: Example where encoding the distance of the object furthest away (solid green) is better than that of the one closest to the image centre (dashed red). The IC model assumes that only one person is in the middle in the former case, and infers that many people may be gathered around a table in the latter.

representation being similar to other images with only one person in the centre of the image (and also on a kitchen counter). Representing the person as the one furthest from the image will result in some inference (from training data) that there could be more than one person in the image sitting *around* the kitchen counter rather than a single person standing *at* the kitchen counter.

The combination of results (bottom row of Table 1) shows that the three features (frequency, object max size and min distance) are complementary, and that combining any pair gives better CIDEr scores than each alone. The combination of all three features produces the best results. These results are interesting, as adding spatial information of even just one object per category can produce a better score. This has, to our knowledge, not been previously demonstrated. The performance of using an explicit detector rather than ground truth annotations is poorer, as expected from noisy detections. However, the overall trend generally remains similar, except for the combination of all three features which gave poorer scores.

Finally, Figure 3 shows the results of partially removing or masking the information captured by the bag of object representation (frequency). As expected, IC performance degrades when less than 75% of information is retained. The performance of the system where the representation is reduced using frequency information suffers the most (even worst than removing categories randomly), suggesting that frequency does not correspond to an object category's importance, i.e. just because there is only one *person* in the image does

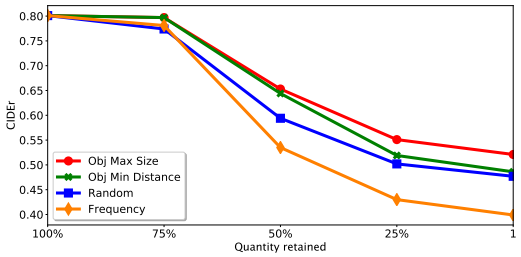


Figure 3: Change in CIDEr scores for image captioning by reducing the number of (ground truth) object instances in the image representation, based on different heuristics.

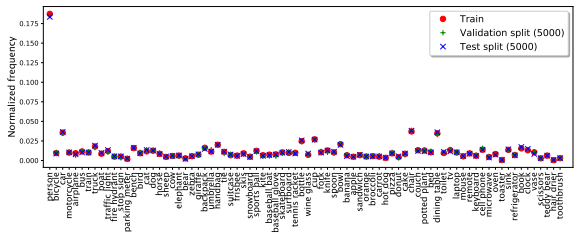


Figure 4: Object category distributions for COCO train, validation and test splits: normalized document frequency of each category. The distribution between the training and test sets are almost identical. A higher resolution version can be found in Appendix B.

not mean that it is less important than the ten *car*s depicted. On the other hand, object size correlates with object importance in IC, i.e. larger objects are more important than smaller objects for IC: The performance does not degrade as much as removing categories by their frequency in the image.

3.4 Analysis

We hypothesize that the bag of objects representation performs well because it serves as a good representation for the dataset and allows for better image matching. One observation is that the category distribution between the training and test sets are very similar (Figure 4), thus increasing the chance of the bag of objects representation producing a close match to one in the training set. From this observation, we posit that end-to-end IC models leverage COCO being repetitive to find similar matches for a test image to a combination of images in the training set. Further investigation on the category distribution (e.g. by splitting the dataset such that the test set contains unseen categories) is left for future work.

k-Nearest neighbour analysis. We further investigate our claim that end-to-end IC systems essentially perform complex image matching against the training set with the following experiment. The idea is that if the IC model performs some form of image matching and text retrieval from the training set, then the nearest neighbour (from training) of a test image should have a caption similar to the one generated by the model. However, the model does not always perform text retrieval as the LSTM is known to sometimes generate novel captions, possibly by aggregating or 'averaging' the captions of similar images and performing some factorization. We first generate captions for every training image using the bag of objects model (with ground truth frequency counts). We then compute the k-nearest training images for each given test image using both the bag of objects representation and its projection (Eq. 1). Finally, we compute the similarity score between the generated caption of the test image against all k-nearest captions. The similarity score measures how well a generated caption matches its nearest neighbour's captions. We expect the score to be high if the IC system generates an image similar to something 'summarized' from the training set.

Type	BLEU	Meteor	CIDEr	SPICE
Freq.	0.868	0.591	6.956	0.737
Proj.	0.912	0.634	7.651	0.799
Exact (2301)	1.000	1.000	10.000	1.000
Freq. (¬ Exact)	0.757	0.498	4.337	0.512
Proj. (¬ Exact)	0.837	0.560	5.638	0.628

Table 3: k-Nearest Neighbour (k=5) trial on the ground truth bag of objects (**Freq.**) and the projected bag of objects (**Proj.**) representations. The references are captions of 5-nearest images in each space. *Exact* represents a subset of 2301 samples where all the 5 neighbours have 0 distance (replicas) and ¬ represents nearest neighbours that are not replicas of the test image.

As reported in Table 3, overall the captions seem to closely match the captions of 5 nearest training images. Further analysis showed that 2301 out of 5000 captions had nearest images at a zero distance, i.e., the same exact representation was seen at least 5 times in training (note that CIDEr gives a score of 10 only if the test caption and *all* references are the same). We found that among the non-exact image matches, the projected image representation better captures candidates in the training set than bag of objects. Figure 5 shows the five nearest neighbours of an example non-exact match and their generated captions in the

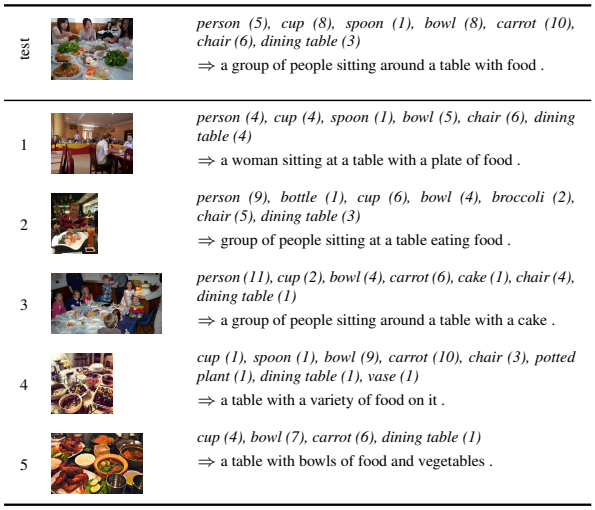


Figure 5: Five nearest neighbours from the training set in the projected space, for an example test image's (top row) original bag of objects representation that does not have an exact match in the training. For each image, we show the ground truth categories (and frequencies in parenthesis) and the generated caption. More examples can be found in Appendix D.

projection space. Note that the nearest neighbours are an approximation since we do not know the exact distance metric derived from the LSTM. We observe that the captions for unseen representations seem to be interpolated from multiple neighbouring points in the projection space, but further work is needed to analyze the hidden representations of the LSTM to understand the language model and to give firmer conclusions.

4 Spatial information on instances

Here we further explore the effect of incorporating spatial information of object detections for IC. More specifically, we enrich the representations by encoding positional and size information for more object instances, rather than restricting the encoding to only one instance per category which makes the representation less informative.

4.1 Spatial representation

We explore encoding **object instances and their spatial properties** as a fixed-size vector. In contrast to Section 3, we propose handling multiple instances of the same category by encoding spatial properties of *individual* instances rather than aggregating them as a single value. Each instance is represented as a tuple (x, y, w, h, a), where x and y are the coordinates of the centre of the bounding box and are normalized to the image width and height respectively, w and h are the width and height of the bounding box respectively, and a is the area covered by the object segment and normalized to the image size. Note that $w \times h \geq a$ (box encloses the segment). We assume that there are maximum 10 instances per vector, and instances of the same category are ordered by a (largest instance first). We encode each of the 80 categories as separate sets. Non-existent objects are represented with zeros. The dimension of the final vector is 4000 ($80 \times 10 \times 5$). We also perform a **feature ablation** experiment to isolate the contribution of different spatial components.

Feature set	Fixed	Tuned
Bag of objects	0.807	0.834
(x, y, w, h, a)	0.870	0.915
(x, y, w, h)	0.859	0.898
(x, y, a)	0.850	0.900
(w, h)	0.870	0.920
(a)	0.869	0.857
(x, y)	0.810	0.863
LSTM Yin and Ordonez (2017)†		0.922

Table 4: CIDEr scores for image captioning using representations encoding spatial information of instances derived from ground truth annotations, with either fixed hyperparameters (Section 3.1) or with hyperparameter tuning. † Results taken from (Yin and Ordonez, 2017).

4.2 Experiments

All experiments in this subsection use ground truth annotations – we expect the results of using an object detector to be slightly worse but in most cases follow a similar trend, as shown in the previous section. Table 4 shows the CIDEr scores using the same setup as Section 3, but using representations with spatial information about individual object instances. Encoding spatial information led to substantially better performance over bag of objects alone. Consistent with our previous observation, w and h (bounding box width and height) seems to be the most informative feature combination – it performs well even without positional information. Area (a) is less informative than the combination of w and h, possibly because it compresses width-height ratio information despite discarding noise from background regions. Positional information (x, y) does not seem to be as informative, consistent with observations from previous work (Wang and Gaizauskas, 2016).

The last column in Table 4 shows the CIDEr

Image ID: 378657

Objects in the image	*person*, *clock*
Representation	**Caption**
Frequency	a large clock tower with a large clock on it .
Object min distance	a clock tower with a large clock on it 's face .
Object max size	a man standing in front of a clock tower .
All three features	a clock tower with people standing in the middle of the water .
(x, y)	a large clock tower with a clock on the front .
(w, h)	a clock on a pole in front of a building
(a)	a large clock tower with people walking around it
(x, y, w, h, a)	a group of people standing around a clock tower .
CNN (ResNet-152)	a large building with a clock tower in the middle of it .
person removed	a clock tower with a weather vane on top of it .

Figure 6: Example captions with different models. The models with explicit object detection and additional spatial information ((x, y, w, h, a)) are more precise in most cases. The output of a standard ResNet-152 POOL5 is also shown, as well as that of the model where the most salient category – *person* – is removed from the feature vector. More example outputs are available in Appendix C.

scores when training the models by performing hyperparameter tuning during training. We note that the results with our simpler image representation are comparable to the ones reported in Yin and Ordonez (2017), which use more complex models to encode similar image information. Interestingly, we observe that positional information (x, y) work better than before tuning in this case.

Example outputs from the models in Sections 3 and 4 can be found in Figure 6.

5 Importance of different categories

In the previous sections, we explore IC based on explicit detections for 80 object categories. However, not all categories are made equal. Some categories could impact IC more than others (Berg et al., 2012). In this section we investigate ***which* categories are more important** for IC on the COCO dataset. Our **category ablation** experiment involves removing *one* category from the 80-dimensional bag of objects (ground truth frequency) representation at a time, resulting in 80 sets of 79D vectors without each ablated category.

We postulate that salient categories should lead to larger performance degradation than others. However, what makes a category 'salient' in general (*dog* vs. *cup*)? We hypothesize that it could be due to (i) how frequently it is depicted across images; (ii) how frequently it is mentioned in the captions when depicted in the image. To quantify these hypotheses, we **compute the rank correlation** between changes in CIDEr from removing the category and each of the statistic below:

- $f(v_c) = \sum_i^N 1(c \in C_i)$: frequency of the ablated category c being annotated among N images in the training set, where C_i is the set of all categories annotated in image i, and $1(x)$ is the indicator function.
- $p(t_c|v_c) \approx \frac{f(t_c,v_c)}{f(v_c)}$: proportion of ablated category being mentioned in any of the reference captions given that it is annotated in the image in the training set.

For determining whether a depicted category is mentioned in the caption, the matching method described in Ramisa et al. (2015) is used to increase recall by matching category labels with (i) the term themselves; (ii) the head noun for multiword expressions; (iii) WordNet synonyms and hyponyms. We treat these statistics as an approximation because of the potential noise from the matching process, although it is clean enough for our purposes.

We have also tried computing the correlation with $f(t_c)$ (frequency of the category being mentioned regardless of whether or not it is depicted). However, we found the word matching process too noisy as it is not constrained or grounded on the image (e.g. "hot dog" is matched to the *dog* category). Thus, we do not report the results for this.

5.1 Experiments

Figure 7 shows the result of the category ablation experiment. Categories like *train*, *sandwich*, *person* and *spoon* led to the largest drop in CIDEr scores. On the other end, categories like *surfboard*, *carrot* and *book* can be removed without negatively affecting the overall score.

By comparing the CIDEr score changes against the frequency counts of object annotations in the training set (top row), there does not seem to be a clear correlation between depiction frequency and CIDEr. Categories like *bear* are infrequent but led to a large drop in score; likewise, *chair* and *dining table* are frequent but do not affect the results

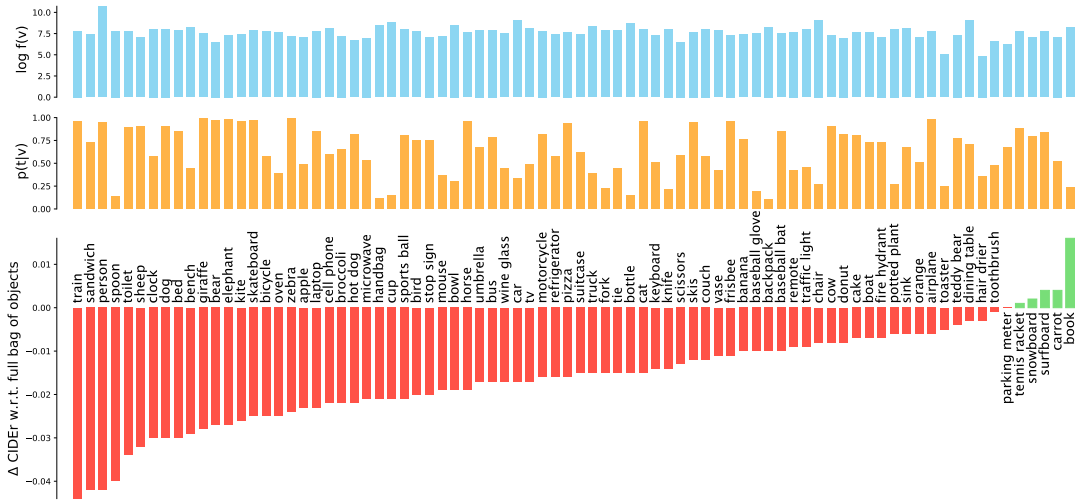


Figure 7: Difference in CIDEr scores when removing each category from the bag of objects representation (79 dimensions), compared to using the full 80D vector (bottom plot). See main text for details.

	Coefficient value (p-value)	
	$f(v_c)$	$p(t_c\|v_c)$
Spearman's ρ	0.137 (0.226)	0.227 (0.043)
Kendall's τ	0.093 (0.223)	0.153 (0.047)

Table 5: Correlation between changes in CIDEr score from category ablation and the frequency of depiction of the category ($f(v_c)$) against the probably of it being mentioned in the caption given depiction ($(p(t_c|v_c)$).

as negatively. In contrast, the frequency of a category being mentioned given that it is depicted is a better predictor for the changes in CIDEr scores in general (middle row). Animate objects seem to be important to IC and are often mentioned in captions (Berg et al., 2012). Interestingly, removing *spoon* greatly affects the results even though it is not frequent in captions.

Table 5 presents the rank correlation (Spearman's ρ and Kendall's τ, two-tailed test) between changes in CIDEr and the two heuristics. While both heuristics are positively correlated with the changes in CIDEr, we can conclude that the frequency of being mentioned (given that it is depicted) is better correlated with the score changes than the frequency of depiction. Of course, the categories are not mutually exclusive and object co-occurrence may also play a role. However, we leave this analysis for future work.

Figure 6 shows an example when the category *person* is removed from the feature vector. Here, the model does not generate any text related to *person*, as the training set contains images of clocks without people in it.

6 Conclusions

In this paper we investigated end-to-end image captioning by using highly interpretable representations derived from explicit object detections. We provided an in-depth analysis on the efficacy of a variety of cues derived from object detections for IC. We found that frequency counts, object size and position are informative and complementary. We also found that some categories have a bigger impact on IC than others. Our analysis showed that end-to-end IC systems are image matching systems that project image representations into a learned space and allow the LSTM to generate captions for images in that projected space.

Future work includes (i) investigating how object category information can be better used or expanded to improve IC; (ii) analyzing end-to-end IC systems by using interpretable representations that rely on other explicit detectors (e.g. actions, scenes, attributes). The use of such explicit information about object instances could help improve our understanding of image captioning.

Acknowledgments

This work is supported by the MultiMT project (H2020 ERC Starting Grant No. 678017). The authors also thank the anonymous reviewers for their valuable feedback on an earlier draft of the paper.

References

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and VQA. In *Proceedings of the IEEE Conference on Computer Vision & Pattern Recognition (CVPR)*. IEEE.

Alexander C. Berg, Tamara L. Berg, Hal Daumé III, Jesse Dodge, Amit Goyal, Xufeng Han, Alyssa Mensch, Margaret Mitchell, Aneesh Sood, Karl Stratos, and Kota Yamaguchi. 2012. Understanding and predicting importance in images. In *Proceedings of the IEEE Conference on Computer Vision & Pattern Recognition (CVPR)*, pages 3562–3569. IEEE.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2016. Fast and accurate deep network learning by exponential linear units (ELUs). In *Proceedings of the International Conference on Learning Representation (ICLR)*. arXiv.org.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380. Association for Computational Linguistics.

Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision & Pattern Recognition (CVPR)*, pages 2625–2634. IEEE.

Desmond Elliott and Frank Keller. 2013. Image description using visual dependency representations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1292–1302. Association for Computational Linguistics.

Mark Everingham, S. M. Ali Eslami, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. 2015. The PASCAL Visual Object Classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136.

Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K. Srivastava, Li Deng, Piotr Dollar, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C. Platt, C. Lawrence Zitnick, and Geoffrey Zweig. 2015. From captions to visual concepts and back. In *Proceedings of the IEEE Conference on Computer Vision & Pattern Recognition (CVPR)*, pages 1473–1482. IEEE.

Zhe Gan, Chuang Gan, Xiaodong He, Yunchen Pu, Kenneth Tran, Jianfeng Gao, Lawrence Carin, and Li Deng. 2017. Semantic compositional networks for visual captioning. In *Proceedings of the IEEE Conference on Computer Vision & Pattern Recognition (CVPR)*, pages 1141–1150. IEEE.

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

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision & Pattern Recognition (CVPR)*, pages 3128–3137. IEEE.

Girish Kulkarni, Visruth Premraj, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C. Berg, and Tamara L. Berg. 2011. Baby talk: Understanding and generating simple image descriptions. In *Proceedings of the IEEE Conference on Computer Vision & Pattern Recognition (CVPR)*, pages 1601–1608. IEEE.

Linghui Li, Sheng Tang, Lixi Deng, Yongdong Zhang, and Qi Tian. 2017. Image caption with global-local attention. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 4133–4139. AAAI.

Siming Li, Girish Kulkarni, Tamara L. Berg, Alexander C. Berg, and Yejin Choi. 2011. Composing simple image descriptions using web-scale n-grams. In *Proceedings of the SIGNLL Conference on Computational Natural Language Learning (CoNLL)*, pages 220–228. Association for Computational Linguistics.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common objects in context. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 740–755. Springer International Publishing.

Margaret Mitchell, Jesse Dodge, Amit Goyal, Kota Yamaguchi, Karl Stratos, Xufeng Han, Alyssa Mensch, Alex Berg, Tamara Berg, and Hal Daumé III. 2012. Midge: Generating image descriptions from computer vision detections. In *Proceedings of the European Chapter of the Association for Computational Linguistics*, pages 747–756. Association for Computational Linguistics.

Arnau Ramisa, Josiah Wang, Ying Lu, Emmanuel Dellandrea, Francesc Moreno-Noguer, and Robert Gaizauskas. 2015. Combining geometric, textual and visual features for predicting prepositions in image descriptions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 214–220. Association for Computational Linguistics.

Joseph Redmon and Ali Farhadi. 2017. YOLO9000: Better, Faster, Stronger. In *Proceedings of the IEEE Conference on Computer Vision & Pattern Recognition (CVPR)*, pages 6517–6525. IEEE.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2015. CIDEr: Consensus-based image description evaluation. In *Proceedings of the IEEE Conference on Computer Vision & Pattern Recognition (CVPR)*, pages 4566–4575. IEEE.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision & Pattern Recognition (CVPR)*, pages 3156–3164. IEEE.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2017. Show and tell: Lessons learned from the 2015 MSCOCO image captioning challenge. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):652–663.

Josiah Wang and Robert Gaizauskas. 2016. Don't mention the shoe! A learning to rank approach to content selection for image description generation. In *Proceedings of the Ninth International Natural Language Generation Conference (INLG)*, pages 193–202. Association for Computational Linguistics.

Qi Wu, Chunhua Shen, Lingqiao Liu, Anthony Dick, and Anton van den Hengel. 2016. What value do explicit high level concepts have in vision to language problems? In *Proceedings of the IEEE Conference on Computer Vision & Pattern Recognition (CVPR)*, pages 203–212. IEEE.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2048–2057. PMLR.

Yezhou Yang, Ching Teo, Hal Daumé III, and Yiannis Aloimonos. 2011. Corpus-guided sentence generation of natural images. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 444–454. Association for Computational Linguistics.

Mark Yatskar, Michel Galley, Lucy Vanderwende, and Luke Zettlemoyer. 2014. See no evil, say no evil: Description generation from densely labeled images. In *Proceedings of the Third Joint Conference on Lexical and Computational Semantics (*SEM 2014)*, pages 110–120. Association for Computational Linguistics and Dublin City University.

Xuwang Yin and Vicente Ordonez. 2017. Obj2Text: Generating visually descriptive language from object layouts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 177–187. Association for Computational Linguistics.

Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. 2016. Image captioning with semantic attention. In *Proceedings of the IEEE Conference on Computer Vision & Pattern Recognition (CVPR)*, pages 4651–4659. IEEE.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *CoRR*, abs/1409.2329.

A Hyperparmeter Settings

The hyperparameter settings for our model are as follows:

- **LSTM layers**: 2-Layer LSTM
- **Word Embedding Dimensionality**: 128
- **Hidden Layer Dimensionality**: 256
- **Maximum Epochs**: 50
- **Batch Size**: [50, 100]
- **LSTM dropout settings**: [0.2, 0.7]
- **Vocabulary threshold**: 2
- **Learning Rate**: [1e-4, 4e-4]
- **Optimiser**: Adam

For items in a range of values, we used grid search to tune the hyperparmeters.

B Full Experimental Results

Tables 6 and 7 show the results of several of our experiments with the most common metrics used in image captioning: BLEU, Meteor, ROUGE$_L$, CIDEr and SPICE.

Figure 8 gives a high resolution version of Figure 4, showing the similarity between train and test distributions in terms of object categories.

C Example captions for different models

Figure 9 shows example images from COCO and the output captions from different models. We compare the outputs of selected models from Sections 3 and 4, and a model where the *person* category is removed from the input vector (Section 5).

D Example nearest neighbours for test images

Figure 10 shows the five nearest neighbours in the training set of each non-replica example from the test set (where the exact ground truth frequency representation does not occur in training). See Section 3.4 for a more detailed description of the experiment.

Representation	B1	B2	B3	B4	M	R	C	S
ResNet-152 POOL5	0.664	0.480	0.335	0.233	0.220	0.486	0.749	0.150
Frequency	0.668	0.481	0.334	0.231	0.223	0.486	0.807	0.155
Normalized	0.656	0.468	0.324	0.226	0.218	0.477	0.762	0.148
Binarized	0.652	0.465	0.317	0.217	0.218	0.473	0.751	0.146
Object min distance	0.650	0.460	0.316	0.219	0.218	0.474	0.759	0.147
Object max size	0.661	0.476	0.332	0.232	0.224	0.483	0.793	0.151
Obj max size + Obj min distance	0.670	0.482	0.333	0.231	0.225	0.485	0.799	0.153
Frequency + Obj min distance	0.675	0.491	0.345	0.239	0.229	0.495	0.836	0.160
Frequency + Obj max size	0.684	0.496	0.349	0.244	0.228	0.495	0.830	0.159
All three features	0.683	0.501	0.355	0.250	0.229	0.498	0.849	0.162

Table 6: Full results for image captioning using ground truth bag of objects variants as visual representations, for metrics **B**LEU, **M**eteor, **R**OUGE$_L$, **C**IDEr and **S**PICE.

Representation	B1	B2	B3	B4	M	R	C	S
Bag of objects	0.668	0.481	0.334	0.231	0.223	0.486	0.807	0.155
(x, y, w, h, a)	0.683	0.503	0.359	0.255	0.233	0.503	0.870	0.163
(x, y, w, h)	0.687	0.503	0.355	0.251	0.233	0.501	0.859	0.166
(x, y, a)	0.683	0.502	0.356	0.250	0.232	0.501	0.850	0.164
(w, h)	0.693	0.511	0.364	0.256	0.233	0.505	0.870	0.165
(a)	0.684	0.503	0.358	0.254	0.232	0.501	0.869	0.162
(x, y)	0.675	0.488	0.341	0.237	0.224	0.491	0.810	0.155

Table 7: Full results for image captioning, using representations encoding spatial information of instances derived from ground truth annotations with fixed hyperparameters, for metrics **B**LEU, **M**eteor, **R**OUGE$_L$, **C**IDEr and **S**PICE.

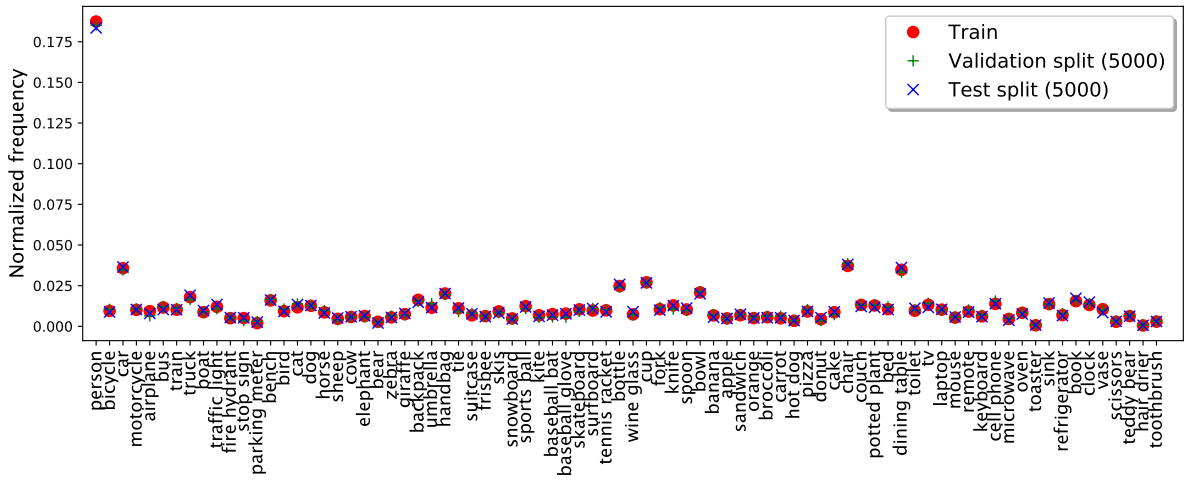


Figure 8: Object category distributions for COCO train, validation and test splits: normalized document frequency of each category. The distribution between the training and test sets are almost identical.

Image ID: 165225

Objects in the image	*person*, *tie*, *tv*, *laptop*, *chair*
Representation	**Caption**
Frequency	a group of people sitting around a table with a laptop .
Object min distance	a man sitting at a desk with a laptop computer .
Object max size	a man standing in front of a tv in a living room .
All three features	a group of people sitting at a table with a laptop .
(x, y)	a group of people standing around a table with a microphone .
(w, h)	a group of people sitting around a table with a laptop .
(a)	a group of people standing in a living room .
(x, y, w, h, a)	a group of people sitting at a table with laptops .
CNN (ResNet-152)	a group of people sitting around the table .
person removed	a man standing in front of a laptop computer .

Image ID: 196715

Objects in the image	*person*, *car*, *truck*, *surfboard*
Representation	**Caption**
Frequency	a man riding a surfboard on top of a wave .
Object min distance	a man riding a surfboard on top of a wave .
Object max size	a red and white truck driving down a street .
All three features	a red and white truck driving down a street .
(x, y)	a large white car parked in a parking lot .
(w, h)	a large white truck with a surfboard on the back of it .
(a)	a man riding a horse on a beach next to a dog .
(x, y, w, h, a)	a car is parked on the side of a street with a car in the background .
CNN (ResNet-152)	a boat is parked on the side of the road .
person removed	a man is standing on a surfboard in the water .

Image ID: 491793

Objects in the image	*person*, *car*, *motorcycle*, *bus*, *handbag*
Representation	**Caption**
Frequency	a group of people riding motorcycles down a street .
Object min distance	a city bus driving down a street next to a traffic light .
Object max size	a bus is driving down the street in the city .
All three features	a group of people standing around a parking lot .
(x, y)	a group of people standing around motorcycles in a parking lot .
(w, h)	a group of people standing around a bus stop .
(a)	a group of people riding bikes down a street .
(x, y, w, h, a)	a group of motorcycles parked next to each other on a street .
CNN (ResNet-152)	a bus is driving down a street with a lot of people .
person removed	a group of people riding motorcycles on a city street .

Image ID: 378657

Objects in the image	*person*, *clock*
Representation	**Caption**
Frequency	a large clock tower with a large clock on it .
Object min distance	a clock tower with a large clock on it 's face .
Object max size	a man standing in front of a clock tower .
All three features	a clock tower with people standing in the middle of the water .
(x, y)	a large clock tower with a clock on the front .
(w, h)	a clock on a pole in front of a building
(a)	a large clock tower with people walking around it
(x, y, w, h, a)	a group of people standing around a clock tower .
CNN (ResNet-152)	a large building with a clock tower in the middle of it .
person removed	a clock tower with a weather vane on top of it .

Figure 9: Examples of descriptions where models differ. The models with explicit object detection and additional spatial information ((x, y, w, h, a)) is more precise in most cases (even though still incorrect in the second example). In the first example, aggregating multiple instances for size and distance cues clearly removes the information about the group of people in the image. The output of a standard CNN (ResNet-152 POOL5) is also shown, as well as that of the model where the most salient category – *person* – is removed.

Test Image ID: 242946

test		*person (5), cup (8), spoon (1), bowl (8), carrot (10), chair (6), dining table (3)* ⇒ a group of people sitting around a table with food .
1		*person (4), cup (4), spoon (1), bowl (5), chair (6), dining table (4)* ⇒ a woman sitting at a table with a plate of food .
2		*person (9), bottle (1), cup (6), bowl (4), broccoli (2), chair (5), dining table (3)* ⇒ a group of people sitting at a table eating food .
3		*person (11), cup (2), bowl (4), carrot (6), cake (1), chair (4), dining table (1)* ⇒ a group of people sitting around a table with a cake .
4		*cup (1), spoon (1), bowl (9), carrot (10), chair (3), potted plant (1), dining table (1), vase (1)* ⇒ a table with a variety of food on it .
5		*cup (4), bowl (7), carrot (6), dining table (1)* ⇒ a table with bowls of food and vegetables .

Test Image ID: 378962

test		*person (14), backpack (3), umbrella (4), handbag (1), banana (4), apple (6), orange (10), chair (7), dining table (2)* ⇒ a group of people standing around a fruit stand .
1		*person (14), backpack (1), banana (5), apple (5), orange (13)* ⇒ a group of people standing around a fruit stand .
2		*person (13), truck (1), backpack (1), apple (2), orange (10)* ⇒ a group of people standing around a fruit stand .
3		*person (12), bicycle (1), handbag (3), banana (2), apple (2), orange (7)* ⇒ a group of people standing around a fruit stand .
4		*person (11), banana (4), apple (1), orange (14)* ⇒ a man standing next to a fruit stand with bananas .
5		*person (14), backpack (1), handbag (3), apple (7), orange (14)* ⇒ a group of people standing around a fruit stand .

Test Image ID: 223648

test		*fork (3), spoon (14), chair (6), dining table (1), book (14)* ⇒ a dining room with a table and chairs .
1		*fork (1), knife (1), spoon (13), scissors (1)* ⇒ a drawer of a variety of different types of food .
2		*person (1), cup (3), knife (1), spoon (14), bowl (1), potted plant (1), dining table (1), vase (1)* ⇒ a woman is sitting at a table with a glass of wine .
3		*person (1), cup (1), fork (3), spoon (6), chair (9), dining table (5), vase (1)* ⇒ a woman sitting at a table with a plate of food .
4		*person (3), bottle (3), wine glass (5), cup (4), fork (3), knife (2), spoon (11), bowl (5), chair (4), dining table (2), book (2)* ⇒ a group of people sitting around a table with food .
5		*fork (4), knife (4), spoon (7), chair (1), couch (2), dining table (1)* ⇒ a table with a table and chairs and a table .

Figure 10: Five nearest neighbours from the training set in the projected space, for several example test images' (top row of each table) original bag of objects representation that does not have an exact match in the training. For each image, we show the ground truth categories (and frequencies in parenthesis) and the generated caption.

Quantifying the visual concreteness of words and topics in multimodal datasets

Jack Hessel
Cornell University
jhessel@cs.cornell.edu

David Mimno
Cornell University
mimno@cornell.edu

Lillian Lee
Cornell University
llee@cs.cornell.edu

Abstract

Multimodal machine learning algorithms aim to learn visual-textual correspondences. Previous work suggests that concepts with *concrete* visual manifestations may be easier to learn than concepts with abstract ones. We give an algorithm for automatically computing the visual concreteness of words and topics within multimodal datasets. We apply the approach in four settings, ranging from image captions to images/text scraped from historical books. In addition to enabling explorations of concepts in multimodal datasets, our concreteness scores predict the capacity of machine learning algorithms to learn textual/visual relationships. We find that 1) concrete concepts are indeed easier to learn; 2) the large number of algorithms we consider have similar failure cases; 3) the precise positive relationship between concreteness and performance varies between datasets. We conclude with recommendations for using concreteness scores to facilitate future multimodal research.

1 Introduction

Text and images are often used together to serve as a richer form of content. For example, news articles may be accompanied by photographs or infographics; images shared on social media are often coupled with descriptions or tags; and textbooks include illustrations, photos, and other visual elements. The ubiquity and diversity of such "text+image" material (henceforth referred to as *multimodal* content) suggest that, from the standpoint of sharing information, images and text are often natural complements.

Ideally, machine learning algorithms that incorporate information from both text and images should have a fuller perspective than those that consider either text or images in isolation. But Hill and Korhonen (2014b) observe that for their particular multimodal architecture, the level of *concreteness* of a concept being represented — intuitively, the idea of a *dog* is more concrete than that of *beauty* — affects whether multimodal or single-channel representations are more effective. In their case, concreteness was derived for 766 nouns and verbs from a fixed psycholinguistic database of human ratings.

In contrast, we introduce an adaptive algorithm for characterizing the visual concreteness of all the concepts indexed textually (e.g., "dog") in a given multimodal dataset. Our approach is to leverage the geometry of image/text space. Intuitively, a visually concrete concept is one associated with more locally similar sets of images; for example, images associated with "dog" will likely contain dogs, whereas images associated with "beautiful" may contain flowers, sunsets, weddings, or an abundance of other possibilities — see Fig. 1.

Allowing concreteness to be dataset-specific is an important innovation because concreteness is contextual. For example, in one dataset we work with, our method scores "London" as highly concrete because of a preponderance of iconic London images in it, such as Big Ben and double-decker buses; whereas for a separate dataset, "London" is used as a geotag for diverse images, so the same word scores as highly non-concrete.

In addition to being dataset-specific, our method readily scales, does not depend on an external search engine, and is compatible with both discrete and continuous textual concepts (e.g., topic distributions).

Dataset-specific visual concreteness scores enable a variety of purposes. In this paper, we

Proceedings of NAACL-HLT 2018, pages 2194–2205
New Orleans, Louisiana, June 1 - 6, 2018.

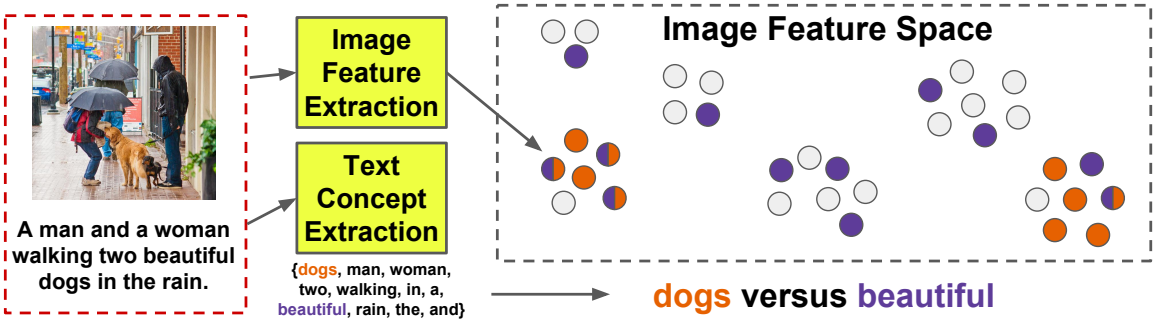


Figure 1: Demonstration of visual concreteness estimation on an example from the COCO dataset. The degree of visual clustering of textual concepts is measured using a nearest neighbor technique. The concreteness of "dogs" is greater than the concreteness of "beautiful" because images associated with "dogs" are packed tightly into two clusters, while images associated with "beautiful" are spread evenly.[1]

focus on using them to: 1) explore multimodal datasets; and 2) predict how easily concepts will be learned in a machine learning setting. We apply our method to four large multimodal datasets, ranging from image captions to image/text data scraped from Wikipedia,[2] to examine the relationship between concreteness scores and the performance of machine learning algorithms. Specifically, we consider the cross-modal retrieval problem, and examine a number of NLP, vision, and retrieval algorithms. Across all 320 significantly different experimental settings (= 4 datasets $\times$ 2 image-representation algorithms $\times$ 5 textual-representation algorithms $\times$ 4 text/image alignment algorithms $\times$ 2 feature pre-processing schemes), we find that more concrete instances are easier to retrieve, and that different algorithms have similar failure cases. Interestingly, the relationship between concreteness and retrievability varies significantly based on dataset: some datasets appear to have a linear relationship between the two, whereas others exhibit a concreteness threshold beyond which retrieval becomes much easier.

We believe that our work can have a positive impact on future multimodal research. §8 gives more detail, but in brief, we see implications in (1) evaluation — more credit should perhaps be assigned to performance on non-concrete concepts; (2) creating or augmenting multimodal datasets, where one might *a priori* consider the desired relative proportion of concrete vs. non-concrete concepts; and (3) *curriculum learning* (Bengio et al., 2009), where ordering of training examples could take concreteness levels into account.

2 Related Work

Applying machine learning to understand visual-textual relationships has enabled a number of new applications, e.g., better accessibility via automatic generation of alt text (Garcia et al., 2016), cheaper training-data acquisition for computer vision (Joulin et al., 2016; Veit et al., 2017), and cross-modal retrieval systems, e.g., Rasiwasia et al. (2010); Costa Pereira et al. (2014).

Multimodal datasets often have substantially differing characteristics, and are used for different tasks (Baltrušaitis et al., 2017). Some commonly used datasets couple images with a handful of unordered tags (Barnard et al., 2003; Cusano et al., 2004; Grangier and Bengio, 2008; Chen et al., 2013, inter alia) or short, literal natural language captions (Farhadi et al., 2010; Ordóñez et al., 2011; Kulkarni et al., 2013; Fang et al., 2015, inter alia). In other cross-modal retrieval settings, images are paired with long, only loosely thematically-related documents. (Khan et al., 2009; Socher and Fei-Fei, 2010; Jia et al., 2011; Zhuang et al., 2013, inter alia). We provide experimental results on both types of data.

Concreteness in datasets has been previously studied in either text-only cases (Turney et al., 2011; Hill et al., 2013) or by incorporating human judgments of perception into models (Silberer and Lapata, 2012; Hill and Korhonen, 2014a). Other work has quantified characteristics of concreteness in multimodal datasets (Young et al., 2014; Hill et al., 2014; Hill and Korhonen, 2014b; Kiela and Bottou, 2014; Jas and Parikh, 2015; Lazari-

[1] Image copyright information is provided in the supplementary material.

[2] We release our Wikipedia and British Library data at http://www.cs.cornell.edu/~jhessel/concreteness/concreteness.html

dou et al., 2015; Silberer et al., 2016; Lu et al., 2017; Bhaskar et al., 2017). Most related to our work is that of Kiela et al. (2014); the authors use Google image search to collect 50 images each for a variety of words and compute the average cosine similarity between vector representations of returned images. In contrast, our method can be tuned to specific datasets without reliance on an external search engine. Other algorithmic advantages of our method include that: it more readily scales than previous solutions, it makes relatively few assumptions regarding the distribution of images/text, it normalizes for word frequency in a principled fashion, and it can produce confidence intervals. Finally, the method we propose can be applied to both discrete and continuous concepts like topic distributions.

3 Quantifying Visual Concreteness

To compute visual concreteness scores, we adopt the same general approach as Kiela et al. (2014): for a fixed text concept (i.e., a word or topic), we measure the variance in the corresponding visual features. The method is summarized in Figure 1.

3.1 Concreteness of discrete words

We assume as input a multimodal dataset of n images represented in a space where nearest neighbors may be computed. Additionally, each image is associated with a set of discrete words/tags. We write w_v for the set of words/tags associated with image v, and V_w for the set of all images associated with a word w. For example, if the v^{th} image is of a dog playing frisbee, w_v might be $\{\text{frisbee}, \text{dog}, \text{in}, \text{park}\}$, and $v \in V_{\text{park}}$.

Our goal is to measure how "clustered" a word is in image feature space. Specifically, we ask: for each image $v \in V_w$, how often are v's nearest neighbors also associated with w? We thus compute the expected value of MNI_w^k, the number of mutually neighboring images of word w:

$$\mathbb{E}_{P_{data}}[\text{MNI}_w^k] = \frac{1}{|V_w|} \sum_{v \in V_w} |\text{NN}^k(v) \cap V_w| \,, \quad (1)$$

where $\text{NN}^k(v)$ denotes the set of v's k nearest neighbors in image space.

While Equation 1 measures clusteredness, it does not properly normalize for frequency. Consider a word like "and"; we expect it to have low concreteness, but its associated images will share neighbors simply because "and" is a frequent unigram. To correct for this, we compute the *concreteness* of a word as the ratio of $\mathbb{E}[\text{MNI}_w^k]$ under the true distribution of the image data to a random distribution of the image data:

$$\text{concreteness}(w) = \frac{\mathbb{E}_{P_{data}}[\text{MNI}_w^k]}{\mathbb{E}_{P_{random}}[\text{MNI}_w^k]} \quad (2)$$

While the denominator of this expression can be computed in closed form, we use $\mathbb{E}_{P_{random}}[\text{MNI}_w^k] \approx \frac{k|V_w|}{n}$; this approximation is faster to compute and is negligibly different from the true expectation in practice.

3.2 Extension to continuous topics

We extend the definition of concreteness to continuous concepts, so that our work applies also to topic model outputs; this extension is needed because the intersection in Equation 1 cannot be directly applied to real values. Assume we are given a set of topics T and an image-by-topic matrix $Y \in \mathbb{R}^{n \times |T|}$, where the v^{th} row[3] is a topic distribution for the text associated with image v, i.e., $Y_{ij} = P(\text{topic } j | \text{image } i)$. For each topic t, we compute the average topic weight for each image v's neighbors, and take a weighted average as:

$$\text{concreteness}(t) = \frac{k}{n} \cdot \frac{\sum_{v=1}^{n} [Y_{vt} \sum_{j \in \text{NN}^k(v)} Y_{jt}]}{\sum_{v=1}^{n} Y_{vt}} \quad (3)$$

Note that Equations 1 and 3 are computations of means. Therefore, confidence intervals can be computed in both cases either using a normality assumption or bootstrapping.

4 Datasets

We consider four datasets that span a variety of multimodal settings. Two are publicly available and widely used (COCO/Flickr); we collected and preprocessed the other two (Wiki/BL). The Wikipedia and British Library sets are available for download at `http://www.cs.cornell.edu/~jhessel/concreteness/concreteness.html`. Dataset statistics are given in Table 1, and summarized as follows:

Wikipedia (Wiki). We collected a dataset consisting of 192K articles from the English Wikipedia, along with the 549K images contained in those

[3] The construction is necessarily different for different types of datasets, as described in §4.

Figure 2: Examples of text and images from our new Wiki/BL datasets.

	# Images	Mean Len	Train/Test
Wiki	549K	1397.8	177K/10K
BL	405K	2269.6	69K/ 5K
COCO	123K	10.5	568K/10K
Flickr	754K	9.0	744K/10K

Table 1: Dataset statistics: total number of images, average text length in words, and size of the train/test splits we use in §6.

articles. Following Wilson's popularity filtering technique,[4] we selected this subset of Wikipedia by identifying articles that received at least 50 views on March 5th, 2016.[5] To our knowledge, the previous largest publicly available multimodal Wikipedia dataset comes from ImageCLEF's 2011 retrieval task (Popescu et al., 2010), which consists of 137K images associated with English articles.

Images often appear on multiple pages: an image of the Eiffel tower might appear on pages for Paris, for Gustave Eiffel, and for the tower itself.

Historical Books from British Library (BL). The British Library has released a set of digitized books (British Library Labs, 2016) consisting of 25M pages of OCRed text, alongside 500K+ images scraped from those pages of text. The release splits images into four categories; we ignore "bound covers" and "embellishments" and use images identified as "plates" and "medium sized." We associated images with all text within a 3-page window.

This raw data collection is noisy. Many books are not in English, some books contain far more images than others, and the images themselves are of varying size and rotation. To combat these issues we only keep books that have identifiably English text; for each cross-validation split in our machine-learning experiments (§6) we sample at most 10 images from each book; and we use *book-level* holdout so that no images/text in the test set are from books in the training set.

Captions and Tags. We also examine two popular existing datasets: Microsoft COCO (captions) (Lin et al., 2014) (**COCO**) and MIRFLICKR-1M (tags) (Huiskes et al., 2010) (**Flickr**). For COCO, we construct our own training/validation splits from the 123K images, each of which has 5 captions. For Flickr, as an initial preprocessing step we only consider the 7.3K tags that appear at least 200 times, and the 754K images that are associated with at least 3 of the 7.3K valid tags.

5 Validation of Concreteness Scoring

We apply our concreteness measure to the four datasets. For COCO and Flickr, we use unigrams as concepts, while for Wiki and BL, we extract 256-dimensional topic distributions using Latent Dirichlet Allocation (LDA) (Blei et al., 2003). For BL, topic distributions are derived from text in the aforementioned 3 page window; for Wiki, for each image, we compute the mean topic distribution of all articles that image appears in; for Flickr, we associate images with all of their tags; for COCO, we concatenate all captions for a given image. For computing concreteness scores for COCO/Flickr, we only consider unigrams associated with at least 100 images, so as to ensure the stability of MNI as defined in Equation 1.

We extract image features from the pre-softmax layer of a deep convolutional neural network, ResNet50 (He et al., 2016), pretrained for the ImageNet classification task (Deng et al., 2009); this method is known to be a strong baseline (Sharif Razavian et al., 2014).[6] For nearest neighbor search, we use the Annoy library,[7] which computes approximate kNN efficiently. We use $k = 50$ nearest neighbors, though the results presented are stable for reasonable choices of k, e.g., $k = 25, 100$.

5.1 Concreteness and human judgments

Following Kiela et al. (2014), we borrow a dataset of human judgments to validate our concreteness

[4] https://goo.gl/B11yyO

[5] The articles were extracted from an early March, 2016 data dump.

[6] We explore different image/text representations in later sections.

[7] github.com/spotify/annoy

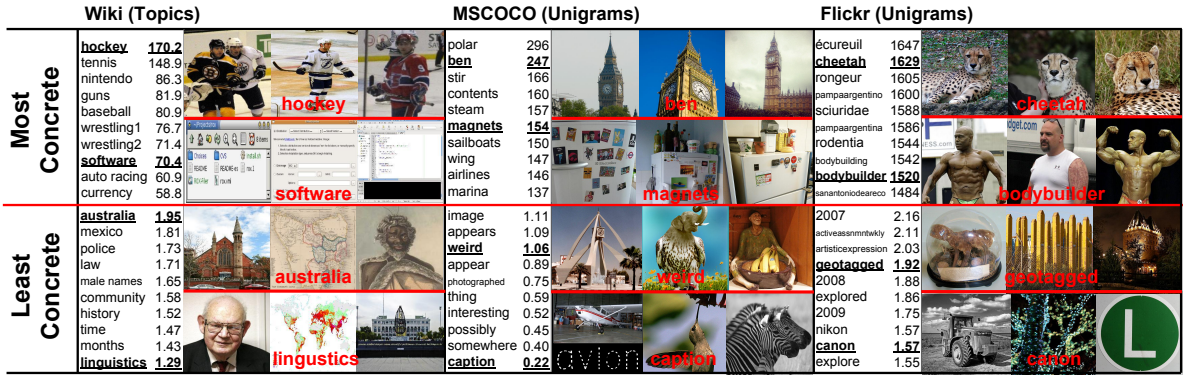


Figure 3: Examples of the most and least concrete words/topics from Wiki, COCO, and Flickr, along with example images associated with each **highlighted** word/topic.

computation method.[8] The concreteness of words is a topic of interest in psychology because concreteness relates to a variety of aspects of human behavior, e.g., language acquisition, memory, etc. Schwanenflugel and Shoben (1983); Paivio (1991); Walker and Hulme (1999); De Groot and Keijzer (2000).

We compare against the human-gathered unigram concreteness judgments provided in the USF Norms dataset (USF) (Nelson et al., 2004); for each unigram, raters provided judgments of its concreteness on a 1-7 scale. For Flickr/COCO, we compute Spearman correlation using these per-unigram scores (the vocabulary overlap between USF and Flickr/COCO is 1.3K/1.6K), and for Wiki/BL, we compute topic-level human judgment scores via a simple average amongst the top 100 most probable words in the topic.

As a null hypothesis, we consider the possibility that our concreteness measure is simply mirroring frequency information.[9] We measure frequency for each dataset by measuring how often a particular word/topic appears in it. A useful concreteness measure should correlate with USF more than a simple frequency baseline does.

For COCO/Flickr/Wiki, concreteness scores output by our method positively correlate with human judgments of concreteness more than frequency does (see Figure 4). For COCO, this pattern holds even when controlling for part-of-speech (not shown), whereas Flickr adjectives are not correlated with USF. For BL, neither frequency nor our concreteness scores are significantly correlated with USF. Thus, in three of our four datasets, our measure tends to predict human concreteness judgments better than frequency.

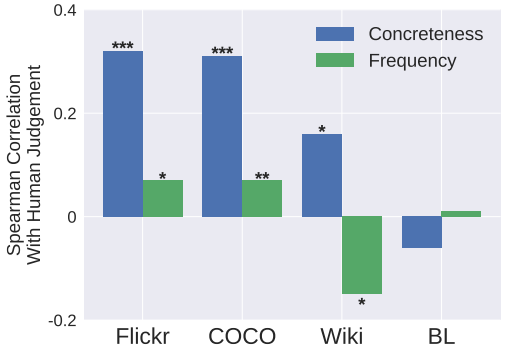


Figure 4: Spearman correlations between human judgment (USF) and our algorithm's outputs, and dataset frequency. In the case of Flickr/COCO/WIKI our concreteness scores correlate with human judgement to a greater extent than frequency. For BL, neither frequency nor our concreteness measure is correlated with human judgement. ***/**/* := $p < .001/.01/.05$

Concreteness and frequency. While concreteness measures correlate with human judgment better than frequency, we do expect *some* correlation between a word's frequency and its concreteness (Gorman, 1961). In all cases, we observe a moderate-to-strong positive correlation between infrequency and concreteness ($\rho_{wiki}, \rho_{coco}, \rho_{flickr}, \rho_{bl} = .06, .35, .40, .71$) indicating that rarer words/topics are more concrete, in general. However, the correlation is not perfect, and concreteness and frequency measure different

[8] Note that because concreteness of words/topics varies from dataset to dataset, we don't expect one set of human judgments to correlate perfectly with our concreteness scores. However, partial correlation with human judgment offers a common-sense "reality check."

[9] We return to this hypothesis in §6.1 as well; there, too, we find that concreteness and frequency capture different information.

properties of words.

5.2 Concreteness within datasets

Figure 3 gives examples from Wiki, COCO, and Flickr illustrating the concepts associated with the smallest and largest concreteness scores according to our method.[10] The scores often align with intuition, e.g., for Wiki, sports topics are often concrete, whereas country-based or abstract-idea-based topics are not.[11] For COCO, *polar* (because of polar bears) and *ben* (because of Big Ben) are concrete; whereas *somewhere* and *possibly* are associated with a wide variety of images.

Concreteness scores form a continuum, making explicit not only the extrema (as in Figure 3) but also the middle ground, e.g., in COCO, "wilderness" (rank 479) is more visually concrete than "outside" (rank 2012). Also, dataset-specific intricacies that are not obvious *a priori* are highlighted, e.g., in COCO, 150/151 references to "magnets" (rank 6) are in the visual context of a refrigerator (making "magnets" visually concrete) though the converse is not true, as both "refrigerator" (rank 329) and "fridge" (rank 272) often appear without magnets; 61 captions in COCO are exactly "There is no image here to provide a *caption* for," and this dataset error is made explicit through concreteness score computations.

5.3 Concreteness varies across datasets

To what extent are the concreteness scores dataset-specific? To investigate this question, we compute the correlation between Flickr and COCO unigram concreteness scores for 1129 overlapping terms. While the two are positively correlated ($\rho = .48, p < .01$) there are many exceptions that highlight the utility of computing dataset-independent scores. For instance, "London" is extremely concrete in COCO (rank 9) as compared to in Flickr (rank 1110). In COCO, images of London tend to be iconic (i.e., Big Ben, double decker buses); in contrast, "London" often serves as a geotag for a wider variety of images in Flickr. Conversely, "watch" in Flickr is concrete (rank 196) as it tends to refer to the timepiece, whereas "watch" is not concrete in COCO (rank 958) as it tends to refer to the verb; while these relationships are not obvious *a priori*, our concreteness method has helped to highlight these usage differences between the image tagging and captioning datasets.

6 Learning Image/Text Correspondences

Previous work suggests that incorporating visual features for less concrete concepts can be harmful in word similarity tasks (Hill and Korhonen, 2014b; Kiela et al., 2014; Kiela and Bottou, 2014; Hill et al., 2014). However, it is less clear if this intuition applies to more practical tasks (e.g., retrieval), or if this problem can be overcome simply by applying the "right" machine learning algorithm. We aim to tackle these questions in this section.

The learning task. The task we consider is the construction of a joint embedding of images and text into a shared vector space. Truly corresponding image/text pairs (e.g., if the text is a caption of that image) should be placed close together in the new space relative to image/text pairs that do not match. This task is a good representative of multimodal learning because computing a joint embedding of text and images is often a "first step" for downstream tasks, e.g., cross-modal retrieval (Rasiwasia et al., 2010), image tagging (Chen et al., 2013), and caption generation (Kiros et al., 2015).

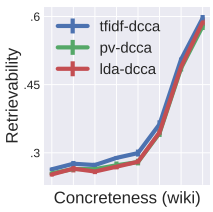
Evaluations. Following previous work in cross-modal retrieval, we measure performance using the top-$k\%$ hit rate (also called recall-at-k-percent, $R@k\%$; higher is better). Cross-modal retrieval can be applied in either direction, i.e., searching for an image given a body of text, or vice-versa. We examine both the image-search-text and text-search-image cases. For simplicity, we average retrieval performance from both directions, producing a single metric;[12] higher is better.

Visual Representations. Echoing Wei et al. (2016), we find that features extracted from convolutional neural networks (CNNs) outperform classical computer vision descriptors (e.g., color histograms) for multimodal retrieval. We consider two different CNNs pretrained on different datasets: ResNet50 features trained on the ImageNet classification task (**RN-Imagenet**), and InceptionV3 (Szegedy et al., 2015) trained on the OpenImages (Krasin et al., 2017) image tagging task (**I3-OpenImages**).

[10] The BL results are less interpretable and are omitted for space reasons.

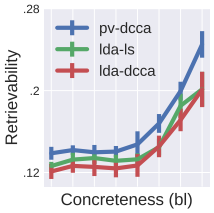
[11] Perhaps fittingly, the "linguistics" topic (top words: term, word, common, list, names, called, form, refer, meaning) is the least visually concrete of all 256 topics.

[12] Averaging is done for ease of presentation; the performance in both directions is similar. Among the parametric approaches (LS/DCCA/NS) across all datasets/NLP algorithms, the mean difference in performance between the directions is



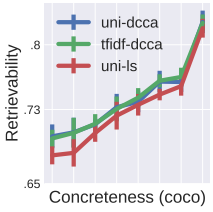
	NP	LS	NS	DCCA
BTM	14.1	19.8	20.7	27.9
LDA	19.8	36.2	33.1	**37.0**
PV	22.0	30.8	29.4	**37.1**
uni	17.3	29.3	30.2	36.3
tfidf	18.1	35.2	33.2	**38.7**

(a) Wikipedia



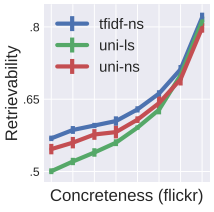
	NP	LS	NS	DCCA
BTM	6.7	7.3	7.2	9.5
LDA	10.2	**17.1**	13.8	**16.4**
PV	12.6	14.1	14.1	**17.8**
uni	11.0	13.2	12.4	15.6
tfidf	10.9	15.1	13.5	15.5

(b) British Library



	NP	LS	NS	DCCA
BTM	27.3	39.9	52.5	58.6
LDA	23.2	51.6	51.9	51.8
PV	14.1	28.4	25.7	33.5
uni	28.7	**74.6**	72.5	**75.0**
tfidf	32.9	74.0	74.1	**74.9**

(c) COCO



	NP	LS	NS	DCCA
BTM	23.9	19.1	31.0	32.4
LDA	18.4	32.2	34.4	34.7
PV	13.9	21.3	20.0	26.6
uni	34.7	**62.5**	**62.0**	59.6
tfidf	35.1	61.6	**63.9**	60.2

(d) Flickr

Figure 5: Concreteness scores versus retrievability (plotted) for each dataset, along with Recall at 1% (in tables, higher is better) for each algorithm combination. Tables give average retrieval performance over 10-fold cross-validation for each combination of NLP/alignment algorithm; the **best**, **second best**, and **third best** performing combinations are bolded and colored. The concreteness versus retrievability curves are plotted for the top-3 performing algorithms, though similar results hold for all algorithms. Our concreteness scores and performance are positively correlated, though the shape of the relationship between the two differs from dataset to dataset (note the differing scales of the y-axes). All results are for RN-ImageNet; the similar I3-OpenImages results are omitted for space reasons.

Text Representations. We consider sparse **unigram** and **tfidf** indicator vectors. In both cases, we limit the vocabulary size to 7.5K. We next consider latent-variable bag-of-words models, including LDA (Blei et al., 2003) (256 topics, trained with Mallet (McCallum, 2002)) a specialized biterm topic model (**BTM**) (Yan et al., 2013) for short texts (30 topics), and paragraph vectors (**PV**) (Le and Mikolov, 2014) (PV-DBOW version, 256 dimensions, trained with Gensim (Řehůřek and Sojka, 2010)).[13]

Alignment of Text and Images. We explore four algorithms for learning correspondences between image and text vectors. We first compare against Hodosh et al. (2013)'s nonparametric baseline (**NP**), which is akin to a nearest-neighbor search. This algorithm is related to the concreteness score algorithm we previously introduced in that it exploits the geometry of the image/text spaces using nearest-neighbor techniques. In general, performance metrics for this algorithm provide an estimate of how "easy" a particular task is in terms of the initial image/text representations.

We next map image features to text features via a simple linear transformation. Let (t_i, v_i) be a text/image pair in the dataset. We learn a linear transformation W that minimizes

$$\sum_i \|W f_{\text{image}}(v_i) - f_{\text{text}}(t_i)\|_2^2 + \lambda \|W\|_F \quad (4)$$

for feature extraction functions f_{image} and f_{text}, e.g., RN-ImageNet/LDA. It is possible to map images onto text as in Equation 4, or map text onto images in an analogous fashion. We find that the directionality of the mapping is important. We train models in both directions, and combine their best-performing results into a single least-squares (**LS**) model.

Next we consider Negative Sampling (**NS**), which balances two objectives: true image/text pairs should be close in the shared latent space, while randomly combined image/text pairs should be far apart. For a text/image pair (t_i, v_i), let $s(t_i, v_i)$ be the cosine similarity of the pair in the shared space. The loss for a single positive example (t_i, v_i) given a negative sample (t'_i, v'_i) is

$$h\big(s(t_i, v_i), s(t_i, v'_i)\big) + h\big(s(t_i, v_i), s(t'_i, v_i)\big) \quad (5)$$

for the hinge function $h(p, n) = \max\{0, \alpha - p + n\}$. Following Kiros et al. (2015) we set $\alpha = .2$.

1.7% (std. dev=2%).

[13] We also ran experiments encoding text using order-aware recurrent neural networks, but we did not observe significant performance differences. Those results are omitted for space reasons.

Finally, we consider Canonical Correlation Analysis (CCA), which projects image and text representations down to independent dimensions of high multimodal correlation. CCA-based methods are popular within the IR community for learning multimodal embeddings (Costa Pereira et al., 2014; Gong et al., 2014). We use Wang et al. (2015b)'s stochastic method for training deep CCA (Andrew et al., 2013) (**DCCA**), a method that is competitive with traditional kernel CCA (Wang et al., 2015a) but less memory-intensive to train.

Training details. LS, NS, and DCCA were implemented using Keras (Chollet et al., 2015).[14] In total, we examine all combinations of: four datasets, five NLP algorithms, two vision algorithms, four cross-modal alignment algorithms, and two feature preprocessing settings; each combination was run using 10-fold cross-validation.

Absolute retrieval quality. The tables in Figure 5 contain the retrieval results for RN-ImageNet image features across each dataset, alignment algorithm, and text representation scheme. We show results for $R@1\%$, but $R@5\%$ and $R@10\%$ are similar. I3-OpenImages image features underperform relative to RN-ImageNet and are omitted for space reasons, though the results are similar.

The BL corpus is the most difficult of the datasets we consider, yielding the lowest retrieval scores. The highly-curated COCO dataset appears to be the easiest, followed by Flickr and then Wikipedia. No single algorithm combination is "best" in all cases.

6.1 Concreteness scores and performance

We now examine the relationship between retrieval performance and concreteness scores. Because concreteness scores are on the word/topic level, we define a *retrievability* metric that summarizes an algorithm's performance on a given concept; for example, we might expect that retrievability(dog) is greater than retrievability(beautiful).

Borrowing the $R@1\%$ metric from the previous section, we let $\mathbb{I}[r_i < 1\%]$ be an indicator variable indicating that test instance i was retrieved correctly, i.e., $\mathbb{I}[r_i < 1\%]$ is 1 if the the average rank r_i of the image-search-text/text-search-image directions is better than 1%, and 0 otherwise. Let s_{ic} be the affinity of test instance i to concept c. In the case of topic distributions, s_{ic} is the proportion of topic c in instance i; in the case of unigrams, s_{ic} is the length-normalized count of unigram c on instance i. Retrievability is defined using a weighted average over test instances i as:

$$\text{retrievability}(c) = \frac{\sum_i s_{ic} \cdot \mathbb{I}[r_i < 1\%]}{\sum_i s_{ic}} \quad (6)$$

The retrievability of c will be higher if instances more associated with c are more easily retrieved by the algorithm.

Retrievability vs. Concreteness. The graphs in Figure 5 plot our concreteness scores versus retrievability of the top 3 performing NLP/alignment algorithm combinations for all 4 datasets. In all cases, there is a strong positive correlation between concreteness and retrievability, which provides evidence that more concrete concepts are easier to retrieve.

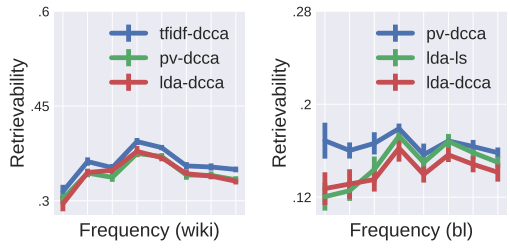
The shape of the concreteness-retrievability curve appears to vary between datasets more than between algorithms. In COCO, the relationship between the two appears to smoothly increase. In Wiki, on the other hand, there appears to be a concreteness threshold, beyond which retrieval becomes much easier.

There is little relationship between retrievability and frequency, further suggesting that our concreteness measure is not simply mirroring frequency. We re-made the plots in Figure 5, except we swapped the x-axis from concreteness to frequency; the resulting plots, given in Figure 6, are much flatter, indicating that retrievability and frequency are mostly uncorrelated. Additional regression analyses reveal that for the top-3 performing algorithms on Flickr/Wiki/BL/COCO, concreteness explains 33%/64%/11%/15% of the variance in retrievability, respectively. In contrast, for all datasets, frequency explained less than 1% of the variance in retrievability.

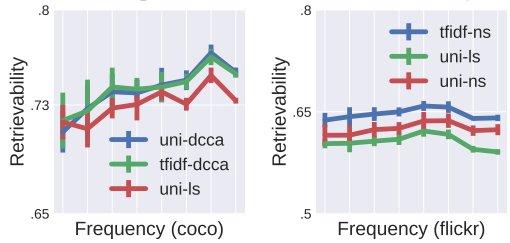
7 Beyond Cross-Modal Retrieval

Concreteness scores do more than just predict retrieval performance; they also predict the difficulty of image classification. Two popular shared tasks from the ImageNet 2015 competition published class-level errors of all entered systems. We used the unigram concreteness scores from

[14] We used Adam (Kingma and Ba, 2015), batch normalization (Ioffe and Szegedy, 2015), and ReLU activations. Regularization and architectures (e.g., number of layers in DCCA/NS, regularization parameter in LS) were chosen over a validation set separately for each cross-validation split. Training is stopped when retrieval metrics decline over the validation set. All models were trained twice, using both raw features and zero-mean/unit-variance features.



(a) Wikipedia (b) British Library



(c) COCO (d) Flickr

Figure 6: Correlation between word/topic frequency and retrievability for each of the four datasets. Compared to our concreteness measure (see Figure 5; note that the while x-axes are different, the y-axes are the same) frequency explains relatively little variance in retrievability.

Flickr/COCO computed in §3 to derive concreteness scores for the ImageNet classes.[15] We find that for both classification and localization, for all 10 top performing entries, and for both Flickr/COCO, there exists a moderate-to-strong Spearman correlation between concreteness and performance among the classes for which concreteness scores were available ($n_{\text{flickr}}, n_{\text{coco}} = 171, 288$; $.18 < \rho < .44$; $p < .003$ in all cases). This result suggests that concrete concepts may tend to be easier on tasks other than retrieval, as well.

8 Future Directions

At present, it remains unclear if abstract concepts should be viewed as noise to be discarded (as in Kiela et al. (2014)), or more difficult, but learnable, signal. Because large datasets (e.g., social media) increasingly mix modalities using ambiguous, abstract language, researchers will need to tackle this question going forward. We hope that visual concreteness scores can guide investigations of the trickiest aspects of multimodal tasks. Our work suggests the following future directions:

Evaluating algorithms: Because concreteness scores are able to predict performance prior to training, evaluations could be reported over concrete and abstract instances separately, as opposed to aggregating into a single performance metric. A new algorithm that consistently performs well on non-concrete concepts, even at the expense of performance on concrete concepts, would represent a significant advance in multimodal learning.

Designing datasets: When constructing a new multimodal dataset, or augmenting an existing one, concreteness scores can offer insights regarding how resources should be allocated. Most directly, these scores enable focusing on "concrete visual concepts" (Huiskes et al., 2010; Chen et al., 2015), by issuing image-search queries could be issued exclusively for concrete concepts during dataset construction. The opposite approach could also be employed, by prioritizing less concrete concepts.

Curriculum learning: During training, instances could be up/down-weighted in the training process in accordance with concreteness scores. It is not clear if placing more weight on the trickier cases (down-weighting concreteness), or giving up on the harder instances (up-weighting concreteness) would lead to better performance, or differing algorithm behavior.

[15] There are 1K classes in both ImageNet tasks, but we were only able to compute concreteness scores for a subset, due to vocabulary differences.

9 Acknowledgments

This work was supported in part by NSF grants SES-1741441/IIS-1652536/IIS-1526155, a Yahoo Faculty Research and Engagement Program grant, the Alfred P. Sloan Foundation, and a Cornell Library Grant. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF, Sloan Foundation, Yahoo, or other sponsors. Nvidia kindly provided the Titan X GPU used in this work.

We would additionally like to thank Maria Antoniak, Amit Bahl, Bharath Hariharan, Arzoo Katiyar, Vlad Niculae, Xanda Schofield, Laure Thompson, Shaomei Wu, and the anonymous reviewers for their constructive comments.

References

Galen Andrew, Raman Arora, Jeff A. Bilmes, and Karen Livescu. 2013. Deep canonical correlation analysis. In *ICML*.

Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. 2017. Multimodal machine learning: A survey and taxonomy. *arXiv preprint 1705.09406*.

Kobus Barnard, Pinar Duygulu, David Forsyth, Nando De Freitas, David M. Blei, and Michael I. Jordan. 2003. Matching words and pictures. *JMLR*.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *ICML*.

Sai Abishek Bhaskar, Maximilian Köper, Sabine Schulte im Walde, and Diego Frassinelli. 2017. Exploring multi-modal text+image models to distinguish between abstract and concrete nouns. In *IWCS Workshop on Foundations of Situated and Multimodal Communication*.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *JMLR*.

British Library Labs. 2016. Digitised books. https://data.bl.uk/digbks/.

Minmin Chen, Alice X. Zheng, and Kilian Q. Weinberger. 2013. Fast image tagging. In *ICML*.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C. Lawrence Zitnick. 2015. Microsoft COCO captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*.

François Chollet et al. 2015. Keras. https://github.com/fchollet/keras.

Jose Costa Pereira, Emanuele Coviello, Gabriel Doyle, Nikhil Rasiwasia, Gert R.G. Lanckriet, Roger Levy, and Nuno Vasconcelos. 2014. On the role of correlation and abstraction in cross-modal multimedia retrieval. *TPAMI*.

Claudio Cusano, Gianluigi Ciocca, and Raimondo Schettini. 2004. Image annotation using SVM. In *Electronic Imaging*.

Annette De Groot and Rineke Keijzer. 2000. What is hard to learn is easy to forget: The roles of word concreteness, cognate status, and word frequency in foreign-language vocabulary learning and forgetting. *Language Learning*, 50(1):1–56.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *CVPR*.

Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K. Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C. Platt, C. Lawrence Zitnick, and Geoffrey Zweig. 2015. From captions to visual concepts and back. In *CVPR*.

Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. 2010. Every picture tells a story: Generating sentences from images. In *ECCV*.

Dario Garcia Garcia, Manohar Paluri, and Shaomei Wu. 2016. Under the hood: Building accessibility tools for the visually impaired on facebook. https://code.facebook.com/posts/457605107772545/under-the-hood-building-accessibility-tools-for-the-visually-impaired-on-facebook/.

Yunchao Gong, Qifa Ke, Michael Isard, and Svetlana Lazebnik. 2014. A multi-view embedding space for modeling internet images, tags, and their semantics. *IJCV*.

Aloysia M. Gorman. 1961. Recognition memory for nouns as a function of abstractness and frequency. *Journal of Experimental Psychology*, 61(1):23–29.

David Grangier and Samy Bengio. 2008. A discriminative kernel-based approach to rank images from text queries. *TPAMI*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

Felix Hill, Douwe Kiela, and Anna Korhonen. 2013. Concreteness and corpora: A theoretical and practical analysis. In *Workshop on Cognitive Modeling and Computational Linguistics*.

Felix Hill and Anna Korhonen. 2014a. Concreteness and subjectivity as dimensions of lexical meaning. In *ACL*, pages 725–731.

Felix Hill and Anna Korhonen. 2014b. Learning abstract concept embeddings from multi-modal data: Since you probably can't see what I mean. In *EMNLP*.

Felix Hill, Roi Reichart, and Anna Korhonen. 2014. Multi-modal models for concrete and abstract concept meaning. *TACL*.

Micah Hodosh, Peter Young, and Julia Hockenmaier. 2013. Framing image description as a ranking task: Data, models and evaluation metrics. *JAIR*.

Mark J. Huiskes, Bart Thomee, and Michael S. Lew. 2010. New trends and ideas in visual concept detection: the MIR flickr retrieval evaluation initiative. In *ACM MIR*.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *JMLR*.

Mainak Jas and Devi Parikh. 2015. Image specificity. In *CVPR*.

Yangqing Jia, Mathieu Salzmann, and Trevor Darrell. 2011. Learning cross-modality similarity for multinomial data. In *ICCV*.

Armand Joulin, Laurens van der Maaten, Allan Jabri, and Nicolas Vasilache. 2016. Learning visual features from large weakly supervised data. In *ECCV*.

Inayatullah Khan, Amir Saffari, and Horst Bischof. 2009. TVGraz: Multi-modal learning of object categories by combining textual and visual features. In *Workshop of the Austrian Association for Pattern Recognition*.

Douwe Kiela and Léon Bottou. 2014. Learning image embeddings using convolutional neural networks for improved multi-modal semantics. In *EMNLP*.

Douwe Kiela, Felix Hill, Anna Korhonen, and Stephen Clark. 2014. Improving multi-modal representations using image dispersion: Why less is sometimes more. In *ACL*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S. Zemel. 2015. Unifying visual-semantic embeddings with multimodal neural language models. *TACL*.

Ivan Krasin, Tom Duerig, Neil Alldrin, Vittorio Ferrari, Sami Abu-El-Haija, Alina Kuznetsova, Hassan Rom, Jasper Uijlings, Stefan Popov, Andreas Veit, Serge Belongie, Victor Gomes, Abhinav Gupta, Chen Sun, Gal Chechik, David Cai, Zheyun Feng, Dhyanesh Narayanan, and Kevin Murphy. 2017. OpenImages: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from* https://github.com/openimages.

Girish Kulkarni, Visruth Premraj, Vicente Ordóñez, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C. Berg, and Tamara L. Berg. 2013. Babytalk: Understanding and generating simple image descriptions. *TPAMI*.

Angeliki Lazaridou, Nghia The Pham, and Marco Baroni. 2015. Combining language and vision with a multimodal skip-gram model. In *NAACL*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common objects in context. In *ECCV*.

Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. 2017. Knowing when to look: Adaptive attention via a visual sentinel for image captioning.

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Douglas L. Nelson, Cathy L. McEvoy, and Thomas A. Schreiber. 2004. The University of South Florida free association, rhyme, and word fragment norms. *Behavior Research Methods, Instruments, & Computers*, 36(3):402–407.

Vicente Ordóñez, Girish Kulkarni, and Tamara L. Berg. 2011. Im2text: Describing images using 1 million captioned photographs. In *NIPS*.

Allan Paivio. 1991. Dual coding theory: Retrospect and current status. *Canadian Journal of Psychology*, 45(3):255–287.

Adrian Popescu, Theodora Tsikrika, and Jana Kludas. 2010. Overview of the Wikipedia retrieval task at ImageCLEF 2010. In *CLEF*.

Nikhil Rasiwasia, Jose Costa Pereira, Emanuele Coviello, Gabriel Doyle, Gert R.G. Lanckriet, Roger Levy, and Nuno Vasconcelos. 2010. A new approach to cross-modal multimedia retrieval. In *ACM MM*.

Radim Řehůřek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *LREC Workshop on NLP Frameworks*.

Paula J. Schwanenflugel and Edward J. Shoben. 1983. Differential context effects in the comprehension of abstract and concrete verbal materials. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 9(1):82–102.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. 2014. CNN features off-the-shelf: an astounding baseline for recognition. In *CVPR Workshop*.

Carina Silberer, Vittorio Ferrari, and Mirella Lapata. 2016. Visually grounded meaning representations. *TPAMI*.

Carina Silberer and Mirella Lapata. 2012. Grounded models of semantic representation. In *EMNLP*.

Richard Socher and Li Fei-Fei. 2010. Connecting modalities: Semi-supervised segmentation and annotation of images using unaligned text corpora. In *CVPR*.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *CVPR*.

Peter D. Turney, Yair Neuman, Dan Assaf, and Yohai Cohen. 2011. Literal and metaphorical sense identification through concrete and abstract context. In *EMNLP*.

Andreas Veit, Neil Alldrin, Gal Chechik, Ivan Krasin, Abhinav Gupta, and Serge Belongie. 2017. Learning from noisy large-scale datasets with minimal supervision. In *CVPR*.

Ian Walker and Charles Hulme. 1999. Concrete words are easier to recall than abstract words: Evidence for a semantic contribution to short-term serial recall. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 25(5):1256–1271.

Weiran Wang, Raman Arora, Karen Livescu, and Jeff A. Bilmes. 2015a. On deep multi-view representation learning. In *ICML*.

Weiran Wang, Raman Arora, Karen Livescu, and Nathan Srebro. 2015b. Stochastic optimization for deep cca via nonlinear orthogonal iterations. In *Communication, Control, and Computing*.

Yunchao Wei, Yao Zhao, Canyi Lu, Shikui Wei, Luoqi Liu, Zhenfeng Zhu, and Shuicheng Yan. 2016. Cross-modal retrieval with cnn visual features: A new baseline. *Transactions on Cybernetics*.

Xiaohui Yan, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. 2013. A biterm topic model for short texts. In *WWW*.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *TACL*.

Yue Ting Zhuang, Yan Fei Wang, Fei Wu, Yin Zhang, and Wei Ming Lu. 2013. Supervised coupled dictionary learning with group structures for multi-modal retrieval. In *AAAI*.

Speaker Naming in Movies

Mahmoud Azab, Mingzhe Wang, Max Smith, Noriyuki Kojima, Jia Deng, Rada Mihalcea
Computer Science and Engineering, University of Michigan
{mazab,mzwang,mxsmith,kojimano,jiadeng,mihalcea}@umich.edu

Abstract

We propose a new model for speaker naming in movies that leverages visual, textual, and acoustic modalities in an unified optimization framework. To evaluate the performance of our model, we introduce a new dataset consisting of six episodes of the Big Bang Theory TV show and eighteen full movies covering different genres. Our experiments show that our multimodal model significantly outperforms several competitive baselines on the average weighted F-score metric. To demonstrate the effectiveness of our framework, we design an end-to-end memory network model that leverages our speaker naming model and achieves state-of-the-art results on the subtitles task of the MovieQA 2017 Challenge.

1 Introduction

Identifying speakers and their names in movies, and videos in general, is a primary task for many video analysis problems, including automatic subtitle labeling (Hu et al., 2015), content-based video indexing and retrieval (Zhang et al., 2009), video summarization (Tapaswi et al., 2014), and video storyline understanding (Tapaswi et al., 2014). It is a very challenging task, as the visual appearance of the characters changes over the course of the movie due to several factors such as scale, clothing, illumination, and so forth (Arandjelovic and Zisserman, 2005; Everingham et al., 2006). The annotation of movie data with speakers' names can be helpful in a number of applications, such as movie question answering (Tapaswi et al., 2016), automatic identification of character relationships (Zhang et al., 2009), or automatic movie captioning (Hu et al., 2015).

Most previous studies relied primarily on visual information (Arandjelovic and Zisserman, 2005; Everingham et al., 2006), and aimed for the slightly different task of face track labeling;

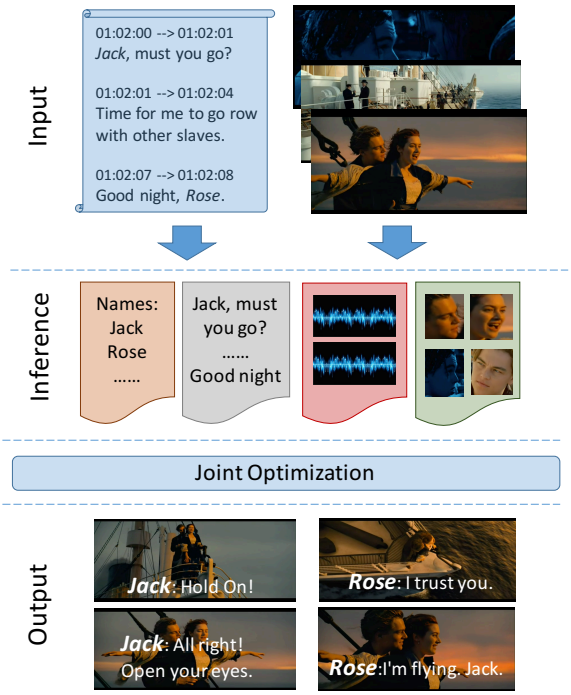


Figure 1: Overview of our approach for speaker naming.

speakers who did not appear in the video frame were not assigned any names, which is common in movies and TV shows. Other available sources of information such as scripts were only used to extract cues about the speakers' names to associate the faces in the videos with their corresponding character name (Everingham et al., 2006; Tapaswi et al., 2015; Bäuml et al., 2013; Sivic et al., 2009); however since scripts are not always available, the applicability of these methods is somehow limited.

Other studies focused on the problem of speaker recognition without naming, using the speech modality as a single source of information. While some of these studies attempted to incorporate the visual modality, their goal was to cluster the speech segments rather than name the speakers

Proceedings of NAACL-HLT 2018, pages 2206–2216
New Orleans, Louisiana, June 1 - 6, 2018.

(Erzin et al., 2005; Bost and Linares, 2014; Kapsouras et al., 2015; Bredin and Gelly, 2016; Hu et al., 2015; Ren et al., 2016). None of these studies used textual information (e.g., dialogue), which prevented them from identifying speaker names.

In our work, we address the task of speaker naming, and propose a new multimodal model that leverages in an unified framework of the visual, speech, and textual modalities that are naturally available while watching a movie. We do not assume the availability of a movie script or a cast list, which makes our model fully unsupervised and easily applicable to unseen movies.

The paper makes two main contributions. First, we introduce a new unsupervised system for speaker naming for movies and TV shows that exclusively depends on videos and subtitles, and relies on a novel unified optimization framework that fuses visual, textual, and acoustic modalities for speaker naming. Second, we construct and make available a dataset consisting of 24 movies with 31,019 turns manually annotated with character names. Additionally, we also evaluate the role of speaker naming when embedded in an end-to-end memory network model, achieving state-of-the-art performance results on the subtitles task of the MovieQA 2017 Challenge.

2 Related Work

The problem of speaker naming in movies has been explored by the computer vision and the speech communities. In the computer vision community, the speaker naming problem is usually considered as a face/person naming problem, in which names are assigned to their corresponding faces on the screen (Everingham et al., 2006; Cour et al., 2010; Bäuml et al., 2013; Haurilet et al., 2016; Tapaswi et al., 2015). On the other hand, the speech community considered the problem as a speaker identification problem, which focuses on recognizing and clustering speakers rather than naming them (Reynolds, 2002; Campbell, 1997). In this work, we aim to solve the problem of speaker naming in movies, in which we label each segment of the subtitles with its corresponding speaker name whether the speaker's face appeared on in the video or not.

Previous work can be furthered categorized according to the type of supervision used to build the character recognition and speaker recognition models: supervised vs. weakly supervised models. In the movie and television domains, utilizing scripts in addition to subtitles to obtain timestamped speaker information was also studied in (Everingham et al., 2006; Tapaswi et al., 2015; Bäuml et al., 2013; Sivic et al., 2009). Moreover, they utilized this information to resolve the ambiguity introduced by co-occurring faces in the same frame. Features were extracted through the period of speaking (detected via lip motion on each face). Then they assigned the face based on candidate names from the time-stamped script. Thus, these studies used speaker recognition as an essential step to construct cast-specific face classifiers. (Tapaswi et al., 2012) extended the face identification problem to include person tracking. They utilized available face recognition results to learn clothing models for characters to identify person tracks without faces.

In (Cour et al., 2010; Haurilet et al., 2016), the authors proposed a weakly supervised model depending on subtitles and a character list. They extracted textual cues from the dialog: first, second, and third person references, such as "I'm Jack", "Hey, Jack!", and "Jack left". Using a character list from IMDB, they mapped these references onto true names using minimum edit distance, and then they ascribed the references to face tracks. Other work removed the dependency on a true character list by determining all names through coreference resolution. However, this work also depended on the availability of scripts (Ramanathan et al., 2014). In our model, we removed the dependency on both the true cast list and the script, which makes it easier to apply our model to other movies and TV shows.

Recent work proposed a convolutional neural network (CNN) and Long Short-Term Memory (LSTM) based learning framework to automatically learn a function that combines both facial and acoustic features (Hu et al., 2015; Ren et al., 2016). Using these cues, they tried to learn matching face-audio pairs and non-matching face-audio pairs. They then trained a SVM classifier on the audio-video pairings to discriminate between the non-overlapping speakers. In order to train their models, they manually identified the leading characters in two TV shows, Friends and The Big Bang Theory (BBT), and collected their face tracks and corresponding audio segments using pre-annotated subtitles. Despite the very high performance reported in these studies, it is very hard to generalize their approach since it requires a lot

of training data.

On the other hand, talking faces have been used to improve speaker recognition and diarization in TV shows (Bredin and Gelly, 2016; Bost and Linares, 2014; Li et al., 2004). In the case of (Liu et al., 2008), they modeled the problem of speaker naming as facial recognition to identify speakers in news broadcasts. This work leveraged optical character recognition to read the broadcasters' names that were displayed on screen, requiring the faces to already be annotated.

3 Datasets

Our dataset consists of a mix of TV show episodes and full movies. For the TV show, we use six full episodes of season one of the BBT. The number of named characters in the BBT episodes varies between 5 to 8 characters per episode, and the background noise level is low. Additionally, we also acquired a set of eighteen full movies from different genres, to evaluate how our model works under different conditions. In this latter dataset, the number of named characters ranges between 6 and 37, and it has varied levels of background noise.

We manually annotated this dataset with the character name of each subtitle segment. To facilitate the annotation process, we built an interface that parses the movies subtitles files, collects the cast list from IMDB for each movie, and then shows one subtitle segment at a time along with the cast list so that the annotator can choose the correct character. Using this tool, human annotators watched the movies and assigned a speaker name to each subtitle segment. If a character name was not mentioned in the dialogue, the annotators labeled it as "unknown." To evaluate the quality of the annotations, five movies in our dataset were double annotated. The Cohen's Kappa inter-annotator agreement score for these five movies is 0.91, which shows a strong level of agreement.

To clean the data, we removed empty segments, as well as subtitle description parts written between brackets such as "[groaning]" and "[sniffing]". We also removed segments with two speakers at the same time. We intentionally avoided using any automatic means to split these segments, to preserve the high-quality of our gold standard.

Table 1 shows the statistics of the collected data. Overall, the dataset consists of 24 videos with a total duration of 40.28 hours, a net dialogue duration of 21.99 hours, and a total of 31,019 turns spoken by 463 different speakers. Four of the movies in this dataset are used as a development set to develop supplementary systems and to fine tune our model's parameters; the remaining movies are used for evaluation.

	Min	Max	Mean	σ
# characters/video	5	37	17.8	9.55
# Subtitle turns/video	488	2212	1302.4	563.06
# words/turn	1	28	8.02	4.157
subtitles duration (sec)	0.342	9.59	2.54	1.02

Table 1: Statistics on the annotated movie dataset.

4 Data Processing and Representations

We process the movies by extracting several textual, acoustic, and visual features.

4.1 Textual Features

We use the following representations for the textual content of the subtitles:

SkipThoughts uses a Recurrent Neural Network to capture the underlying semantic and syntactic properties, and map them to a vector representation (Kiros et al., 2015). We use their pretrained model to compute a 4,800 dimensional sentence representation for each line in the subtitles.[1]

TF-IDF is a traditional weighting scheme in information retrieval. We represent each subtitle as a vector of tf-idf weights, where the length of the vector (i.e., vocabulary size) and the idf scores are obtained from the movie including the subtitle.

4.2 Acoustic Features

For each movie in the dataset, we extract the audio from the center channel. The center channel is usually dedicated to the dialogue in movies, while the other audio channels carry the surrounding sounds from the environment and the musical background. Although doing this does not fully eliminate the noise in the audio signal, it still improves the speech-to-noise ratio of the signal. When a movie has stereo sound (left and right channels only), we down-mix both channels of the stereo stream into a mono channel.

In this work, we use the subtitles timestamps as an estimate of the boundaries that correspond to the uttered speech segments. Usually, each subtitle corresponds to a segment being said by a single speaker. We use the subtitle timestamps for segmentation so that we can avoid automatic speaker diarization errors and focus on the speaker naming problem.

[1] https://github.com/ryankiros/skip-thoughts

To represent the relevant acoustic information from each spoken segment, we use iVectors, which is the state-of-the-art unsupervised approach in speaker verification (Dehak et al., 2011). While other deep learning-based speaker embeddings models also exist, we do not have access to enough supervised data to build such models. We train unsupervised iVectors for each movie in the dataset, using the iVector extractor used in (Khorram et al., 2016). We extract iVectors of size 40 using a Gaussian Mixture Model-Universal Background Model (GMM-UBM) with 512 components. Each iVector corresponds to a speech segment uttered by a single speaker. We fine tune the size of the iVectors and the number of GMM-UBM components using the development dataset.

4.3 Visual Features

We detect faces in the movies every five frames using the recently proposed MTCNN (Zhang et al., 2016) model, which is pretrained for face detection and facial landmark alignment. Based on the results of face detection, we apply the forward and backward tracker with an implementation of the Dlib library (King, 2009; Danelljan et al., 2014) to extract face tracks from each video clip. We represent a face track using its best face in terms of detection score, and use the activations of the fc7 layer of pretrained VGG-Face (Parkhi et al., 2015) network as visual features.

We calculate the distance between the upper lip center and the lower lip center based on the 68-point facial landmark detection implemented in the Dlib library (King, 2009; Kazemi and Sullivan, 2014). This distance is normalized by the height of face bounding boxes and concatenated across frames to represent the amount of mouth opening. A human usually speaks with lips moving with a certain frequency (3.75 Hz to 7.5 Hz used in this work) (Tapaswi et al., 2015). We apply a band-pass filter to amplify the signal of true lip motion in these segments. The overall sum of lip motion is used as the score for the talking face.

5 Unified Optimization Framework

We tackle the problem of speaker naming as a transductive learning problem with constraints. In this approach, we want to use the sparse positive labels extracted from the dialogue and the underlying topological structure of the rest of the unlabeled data. We also incorporate multiple cues extracted from both textual and multimedia information. A unified learning framework is proposed to enable the joint optimization over the automatically labeled and unlabeled data, along with multiple semantic cues.

5.1 Character Identification and Extraction

In this work, we do not consider the set of character names as given because we want to build a model that can be generalized to unseen movies. This strict setting adds to the problem's complexity. To extract the list of characters from the subtitles, we use the Named Entity Recognizer (NER) in the Stanford CoreNLP toolkit (Manning et al., 2014). The output is a long list of person names that are mentioned in the dialogue. This list is prone to errors including, but not limited to, nouns that are misclassified by the NER as person's name such as "Dad" and "Aye", names that are irrelevant to the movie such as "Superman" or named animals, or uncaptured character names.

To clean the extracted names list of each movie, we cluster these names based on string minimum edit distance and their gender. From each cluster, we then pick a name to represent it based on its frequency in the dialogue. The result of this step consists of name clusters along with their distribution in the dialogue. The distribution of each cluster is the sum of all the counts of its members. To filter out irrelevant characters, we run a name reference classifier, which classifies each name into first, second or third person references. If a name was only mentioned as a third person throughout the whole movie, we discard it from the list of characters. We remove any name cluster that has a total count less than three, which takes care of the misclassified names' reference types.

5.2 Grammatical Cues

We use the subtitles to extract the name mentions in the dialogue. These mentions allow us to obtain cues about the speaker name and the absence or the presence of the mentioned character in the surrounding subtitles. Thus, they affect the probability that the mentioned character is the speaker or not. We follow the same name reference categories used in (Cour et al., 2010; Haurilet et al., 2016). We classify a name mention into: first (e.g., "I'm Sheldon"), second (e.g., "Oh, hi, Penny") or third person reference (e.g., "So how did it go with Leslie?"). The first person reference represents a positive constraint that allows us to label the corresponding iVector of the speaker and his face if

it exists during the segment duration. The second person reference represents a multi-instance constraint that suggests that the mentioned name is one of the characters that are present in the scene, which increases the probability of this character to be one of the speakers of the surrounding segments. On the other hand, the third person reference represents a negative constraint, as it suggests that the speaker does not exist in the scene, which lowers the character probability of the character being one of the speakers of the next or the previous subtitle segments.

To identify first, second and third person references, we train a linear support vector classifier. The first person, the second and third person classifier's training data are extracted and labeled from our development dataset, and fine tuned using 10-fold cross-validation. Table 2 shows the results of the classifier on the test data. The average number of first, second and third-person references in each movie are 14.63, 117.21, and 95.71, respectively.

	Precision	Recall	F1-Score
First Person	0.625	0.448	0.522
Second Person	0.844	0.863	0.853
Third Person	0.806	0.806	0.806
Average / Total	0.819	0.822	0.820

Table 2: Performance metrics of the reference classifier on the test data.

5.3 Unified Optimization Framework

Given a set of data points that consist of l labeled[2] and u unlabeled instances, we apply an optimization framework to infer the best prediction of speaker names. Suppose we have $l+u$ instances $X = \{x_1, x_2, ..., x_l, x_{l+1}, ..., x_{l+u}\}$ and K possible character names. We also get the dialogue-based positive labels y_i for instances x_i, where y_i is a k-dimension one-hot vector and $y_i^j = 1$ if x_i belongs to the class j, for every $1 \leq i \leq l$ and $1 \leq j \leq K$. To name each instance x_i, we want to predict another one-hot vector of naming scores $f(x_i)$ for each x_i, such that $\text{argmax}_j f^j(x_i) = z_i$ where z_i is the ground truth number of class for instance x_i.

To combine the positive labels and unlabeled data, we define the objective function for predictions f as follows:

$$L_{initial}(f) = \frac{1}{l}\sum_{i=1}^{l}||f(x_i) - y_i||^2 + \frac{1}{l+u}\sum_{i=1}^{l+u}\sum_{j=1}^{l+u} w_{ij}||f(x_i) - f(x_j)||^2 \quad (1)$$

Here w_{ij} is the similarity between x_i and x_j, which is calculated as the weighted sum of textual, acoustic and visual similarities. The inverse Euclidean distance is used as similarity function for each modality. The weights for different modalities are selected as hyperparameters and tuned on the development set. This objective leads to a convex loss function which is easier to optimize over feasible predictions.

Besides the positive labels obtained from first person name references, we also introduce other semantic constraints and cues to enhance the power of our proposed approach. We implement the following four types of constraints:

Multiple Instance Constraint. Although the second person references cannot directly provide positive constraints, they imply that the mentioned characters have high probabilities to be in this conversation. Following previous work (Cour et al., 2010), we incorporate the second person references as multiple instances constraints into our optimization: if x_i has a second person reference j, we encourage j to be assigned to its neighbors, i.e., its adjacent subtitles with similar timestamps. For the implementation, we simply include multiple instances constraints as a variant of positive labels with decreasing weights s, where $s = 1/(l-i)$ for each neighbor x_l.

Negative Constraint. For the third person references, the mentioned characters may not occur in the conversation and movies. So we treat them as negative constraints, which means they imply that the mentioned characters should not be assigned to corresponding instances. This constraint is formulated as follows:

$$L_{neg}(f) = \sum_{(i,j)\in N}[f^j(x_i)]^2 \quad (2)$$

where N is the set of negative constraints x_i doesn't belong class j.

Gender Constraint. We train a voice-based gender classifier by using the subtitles segments from the four movies in our development dataset (5,543

[2]Note that in our setup, all the labeled instances are obtained automatically, as described above.

segments of subtitles). We use the segments in which we know the speaker's name and manually obtain the ground truth gender label from IMDB. We extract the signal energy, 20 Mel-frequency cepstral coefficients (MFCCs) along with their first and second derivatives, in addition to time- and frequency-based absolute fundamental frequency (f0) statistics as features to represent each segment in the subtitles. The f0 statistics has been found to improve the automatic gender detection performance for short speech segments (Levitan et al., 2016), which fits our case since the median duration of the dialogue turns in our dataset is 2.6 seconds.

The MFCC features are extracted using a step size of 16 msec over a 64 msec window using the method from (Mathieu et al., 2010), while the f0 statistics are extracted using a step size of 25 msec over a 50 msec window as the default configuration in (Eyben et al., 2013). We then use these features to train a logistic regression classifier using the Scikit-learn library (Pedregosa et al., 2011). The average accuracy of the gender classifier on a 10-fold cross-validation is 0.8867.

Given the results for the gender classification of audio segments and character names, we define the gender loss to penalize inconsistency between the predicted gender and character names:

$$L_{gender}(f) = \sum_{(i,j)\in Q_1} P_{ga}(x_i)(1 - P_{gn}(j))f^j(x_i) + \sum_{(i,j)\in Q_2} (1 - P_{ga}(x_i))P_{gn}(j)f^j(x_i) \quad (3)$$

where $P_{ga(x_i)}$ is the probability for instance x_i to be a male, and $P_{gn(j)}$ is the probability for name j to be a male, and $Q_1 = \{(i,j)|P_{ga}(x_i) < 0.5, P_{gn}(j) > 0.5\}$, $Q_2 = \{(i,j)|P_{ga}(x_i) > 0.5, P_{gn}(j) < 0.5\}$.

Distribution Constraint. We automatically analyze the dialogue and extract the number of mentions of each character in the subtitles using Stanford CoreNLP and string matching to capture names that are missed by the named entity recognizer. We then filter the resulting counts by removing third person mention references of each name as we assume that this character does not appear in the surrounding frames. We use the results to estimate the distribution of the speaking characters and their importance in the movies. The main goal of this step is to construct a prior probability distribution for the speakers in each movie.

To encourage our predictions to be consistent with the dialogue-based priors, we penalize the square error between the distributions of predictions and name mentions priors in the following equation:

$$L_{dis}(f) = \sum_{j=1}^{K}(\sum(f^j(x_i)) - d_j)^2 \quad (4)$$

where d_j is the ratio of name j mentions in all subtitles.

Final Framework. Combining the loss in Eqn. 1 and multiple losses with different constraints, we obtain our unified optimization problem:

$$f^* = \arg\min_f \lambda_1 L_{initial}(f) + \lambda_2 L_{MI}(f) + \lambda_3 L_{neg}(f) + \lambda_4 L_{gender}(f) + \lambda_5 L_{dis}(f) \quad (5)$$

All of the λs are hyper-parameters to be tuned on development set. We also include the constraint that predictions for different character names must sum to 1. We solve this constrained optimization problem with projected gradient descent (PGD). Our optimization problem in Eqn. 5 is guaranteed to be a convex optimization problem and therefore projected gradient descent is guaranteed to stop with global optima. PGD usually converges after 800 iterations.

6 Evaluation

We model our task as a classification problem, and use the unified optimization framework described earlier to assign a character name to each subtitle.

Since our dataset is highly unbalanced, with a few main characters usually dominating the entire dataset, we adopt the weighted F-score as our evaluation metric, instead of using an accuracy metric or a micro-average F-score. This allows us to take into account that most of the characters have only a few spoken subtitle segments, while at the same time placing emphasis on the main characters. This leads sometimes to an average weighted F-score that is not between the average precision and recall.

One aspect that is important to note is that characters are often referred to using different names. For example, in the movie "The Devil's Advocate," the character Kevin Lomax is also referred to as Kevin or Kev. In more complicated situations, characters may even have multiple identities, such as the character Saul Bloom in the movie "Ocean's Eleven," who pretends to be another character named Lyman Zerga. Since our

	Precision	Recall	F-score
B1: MFMC	0.0910	0.2749	0.1351
B2: DRA	0.2256	0.1819	0.1861
B3: Gender-based DRA	0.2876	0.2349	0.2317
Our Model (Skip-thoughts)*	0.3468	0.2869	0.2680
Our Model (TF-IDF)*	0.3579	0.2933	0.2805
Our Model (iVectors)	0.2151	0.2347	0.1786
Our Model (Visual)*	0.3348	0.2659	0.2555
Our Model (Visual+iVectors)*	0.3371	0.2720	0.2617
Our Model (TF-IDF+iVectors)*	0.3549	0.2835	0.2643
Our Model (TF-IDF+Visual)*	0.3385	0.2975	0.2821
Our Model (all)*	**0.3720**	**0.3108**	**0.2920**

Table 3: Comparison between the average of macro-weighted average of precision, recall and f-score of the baselines and our model. * means statistically significant (t-test p-value < 0.05) when compared to baseline B3.

goal is to assign names to speakers, and not necessarily solve this coreference problem, we consider the assignment of the subtitle segments to any of the speaker's aliases to be correct. Thus, during the evaluation, we map all the characters' aliases from our model's output to the names in the ground truth annotations. Our mapping does not include other referent nouns such as "Dad," "Buddy," etc.; if a segment gets assigned to any such terms, it is considered a misprediction.

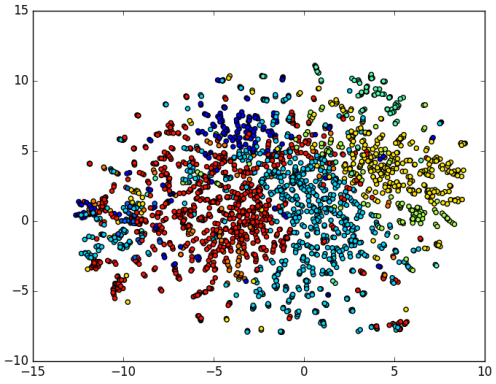
We compare our model against three baselines:

B1: Most-frequently mentioned character consists of selecting the most frequently mentioned character in the dialogue as the speaker for all the subtitles. Even though it is a simple baseline, it achieves an accuracy of 27.1%, since the leading characters tend to speak the most in the movies.

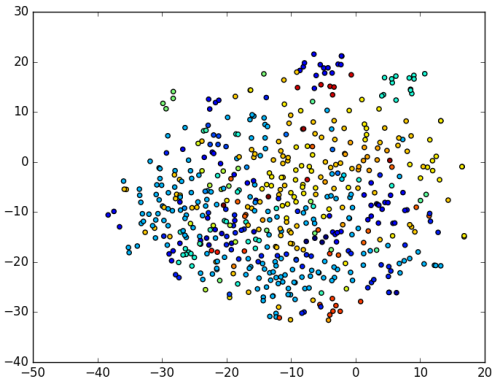
B2: Distribution-driven random assignment consists of randomly assigning character names according to a distribution that reflects their fraction of mentions in all the subtitles.

B3: Gender-based distribution-driven random assignment consists of selecting the speaker names based on the voice-based gender detection classifier. This baseline randomly selects the character name that matches the speaker's gender according to the distribution of mentions of the names in the matching gender category.

The results obtained with our proposed unified optimization framework and the three baselines are shown in Table 3. We also report the performance of the optimization framework using different combinations of the three modalities. The model that uses all three modalities achieves the best results, and outperforms the strongest baseline (B3) by more than 6% absolute in average



(a) The Big Bang Theory



(b) Titanic

Figure 2: For each speech segment, we applied t-SNE (Van Der Maaten, 2014) on their corresponding iVectors. The points with the same color represent instances with the same character name.

weighted F-score. It also significantly outperforms the usage of the visual and acoustic features combined, which have been frequently used together in previous work, suggesting the importance of textual features in this setting.

The ineffectiveness of the iVectors might be a result of the background noise and music, which are difficult to remove from the speech signal. Figure 2 shows the t-Distributed Stochastic Neighbor Embedding (t-SNE) (Van Der Maaten, 2014), which is a nonlinear dimensionality reduction technique that models points in such a way that similar vectors are modeled by nearby points and dissimilar objects are modeled by distant points, visualization of the iVectors over the whole BBT show and the movie "Titanic." In the BBT there is almost no musical background or background noise, while, Titanic has musical background in addition to the background noise such as the screams of the drowning people. From the graph,

the difference between the quality of the iVectors clusters on different noise-levels is clear.

Table 4 shows the effect of adding components of our loss function to the initial loss L_{init} function. The performance of the model using only L_{init} without the other parts is very low due to the sparsity of first person references and errors that the person reference classifier introduces.

	Precision	Recall	F-score
$L_{initial}$	0.0631	0.1576	0.0775
$L_{initial} + L_{gender}$	0.1160	0.1845	0.1210
$L_{initial} + L_{negative}$	0.0825	0.0746	0.0361
$L_{initial} + L_{distribution}$	0.1050	0.1570	0.0608
$L_{initial} + L_{MultipleInstance}$	0.3058	0.2941	0.2189

Table 4: Analysis of the effect of adding each component of the loss function to the initial loss.

In order to analyze the effect of the errors that several of the modules (e.g., gender and name reference classifiers) propagate into the system, we also test our framework by replacing each one of the components with its ground truth information. As seen in Table 5, the results obtained in this setting show significant improvement with the replacement of each component in our framework, which suggests that additional work on these components will have positive implications on the overall system.

	Precision	Recall	F-score
Our Model	0.3720	0.3108	0.2920
Voice Gender (VG)	0.4218	0.3449	0.3259
VG + Name Gender (NG)	0.4412	0.3790	0.3645
VG + NG + Name Ref	0.4403	0.3938	0.3748

Table 5: Comparison between our model while replacing different components with their ground truth information.

7 Speaker Naming for Movie Understanding

Identifying speakers is a critical task for understanding the dialogue and storyline in movies. MovieQA is a challenging dataset for movie understanding. The dataset consists of 14,944 multiple choice questions about 408 movies. Each question has five answers and only one of them is correct. The dataset is divided into three splits: train, validation, and test according to the movie titles. Importantly, there are no overlapping movies between the splits. Table 6 shows examples of the question and answers in the MovieQA dataset.

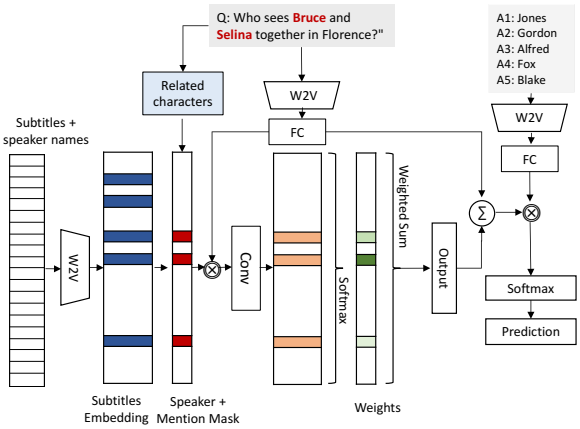


Figure 3: The diagram describing our Speaker-based Convolutional Memory Network (SC-MemN2N) model.

The MovieQA 2017 Challenge[3] consists of six different tasks according to the source of information used to answer the questions. Given that for many of the movies in the dataset the videos are not completely available, we develop our initial system so that it only relies on the subtitles; we thus participate in the challenge subtitles task, which includes the dialogue (without the speaker information) as the only source of information to answer questions.

To demonstrate the effectiveness of our speaker naming approach, we design a model based on an end-to-end memory network (Sukhbaatar et al., 2015), namely Speaker-based Convolutional Memory Network (SC-MemN2N), which relies on the MovieQA dataset, and integrates the speaker naming approach as a component in the network. Specifically, we use our speaker naming framework to infer the name of the speaker for each segment of the subtitles, and prepend the predicted speaker name to each turn in the subtitles.[4] To represent the movie subtitles, we represent each turn in the subtitles as the mean-pooling of a 300-dimension pretrained word2vec (Mikolov et al., 2013) representation of each word in the sentence. We similarly represent the input questions and their corresponding answers. Given a question, we use the SC-MemN2N memory to find an answer. For questions asking about specific characters, we keep the memory slots that have the characters in question as speakers or mentioned in, and mask out the rest of the memory slots. Figure

[3] http://movieqa.cs.toronto.edu/workshops/iccv2017/

[4] We strictly follow the challenge rules, and only use text to infer the speaker names.

Movie	Question	Answers	
Fargo	What did Mike's wife, as he says, die from?	A1: She was killed	**A2: Breast cancer**
		A3: Leukemia	A4: Heart disease
		A5: Complications due to child birth	
Titanic	What does Rose ask Jack to do in her room?	A1: Sketch her in her best dress	A2: Sketch her nude
		A3: Take a picture of her nude	**A4: Paint her nude**
		A5: Take a picture of her in her best dress	

Table 6: Example of questions and answers from the MQA benchmark. The answers in bold are the correct answers to their corresponding question.

3 shows the architecture of our model.

Table 7 includes the results of our system on the validation and test sets, along with the best systems introduced in previous work, showing that our SC-MemN2N achieves the best performance. Furthermore, to measure the effectiveness of adding the speaker names and masking, we test our model after removing the names from the network (C-MemN2N). As seen from the results, the gain of SC-MemN2N is statistically significant[5] compared to a version of the system that does not include the speaker names (C-MemN2N). Figure 4 shows the performance of both C-MemN2N and SC-MemN2N models by question type. The results suggest that our speaker naming helps the model better distinguish between characters, and that prepending the speaker names to the subtitle segments improves the ability of the memory network to correctly identify the supporting facts from the story that answers a given question.

Method	Subtitles	
	val	test
SSCB-W2V (Tapaswi et al., 2016)	24.8	23.7
SSCB-TF-IDF (Tapaswi et al., 2016)	27.6	26.5
SSCB Fusion (Tapaswi et al., 2016)	27.7	-
MemN2N (Tapaswi et al., 2016)	38.0	36.9
Understanding visual regions	-	37.4
RWMN (Na et al., 2017)	40.4	38.5
C-MemN2N (w/o SN)	40.6	-
SC-MemN2N (Ours)	**42.7**	**39.4**

Table 7: Performance comparison for the subtitles task on the MovieQA 2017 Challenge on both validation and test sets. We compare our models with the best existing models (from the challenge leaderboard).

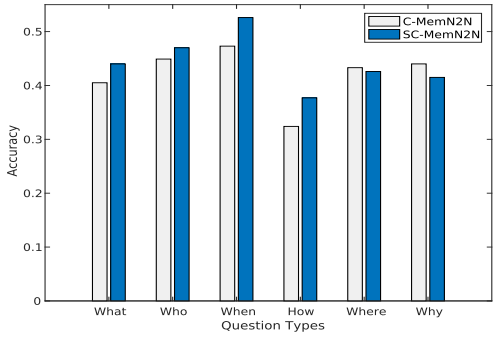


Figure 4: Accuracy comparison according to question type.

8 Conclusion

In this paper, we proposed a unified optimization framework for the task of speaker naming in movies. We addressed this task under a difficult setup, without a cast-list, without supervision from a script, and dealing with the complicated conditions of real movies. Our model includes textual, visual, and acoustic modalities, and incorporates several grammatical and acoustic constraints. Empirical experiments on a movie dataset demonstrated the effectiveness of our proposed method with respect to several competitive baselines. We also showed that an SC-MemN2N model that leverages our speaker naming model can achieve state-of-the-art results on the subtitles task of the MovieQA 2017 Challenge.

The dataset annotated with character names introduced in this paper is publicly available from `http://lit.eecs.umich.edu/downloads.html`.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments and suggestions. This work is supported by a Samsung research grant and by a DARPA grant HR001117S0026-AIDA-FP-045.

[5]Using a t-test p-value<0.05 with 1,000 folds each containing 20 samples.

References

Ognjen Arandjelovic and Andrew Zisserman. 2005. Automatic face recognition for film character retrieval in feature-length films. In *Computer Vision and Pattern Recognition (CVPR)*.

Martin Bäuml, Makarand Tapaswi, and Rainer Stiefelhagen. 2013. Semi-supervised learning with constraints for person identification in multimedia data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xavier Bost and Georges Linares. 2014. Constrained speaker diarization of tv series based on visual patterns. In *IEEE Spoken Language Technology Workshop (SLT)*.

Hervé Bredin and Grégory Gelly. 2016. Improving speaker diarization of tv series using talking-face detection and clustering. In *Proceedings of the 24th Annual ACM Conference on Multimedia*.

Joseph P Campbell. 1997. Speaker recognition: A tutorial. *Proceedings of the IEEE* .

Timothee Cour, Benjamin Sapp, Akash Nagle, and Ben Taskar. 2010. Talking pictures: Temporal grouping and dialog-supervised person recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Martin Danelljan, Gustav Häger, Fahad Khan, and Michael Felsberg. 2014. Accurate scale estimation for robust visual tracking. In *British Machine Vision Conference (BMVC)*.

Najim Dehak, Patrick J Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet. 2011. Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing* .

Engin Erzin, Yücel Yemez, and A Murat Tekalp. 2005. Multimodal speaker identification using an adaptive classifier cascade based on modality reliability. *IEEE Transactions on Multimedia* .

Mark Everingham, Josef Sivic, and Andrew Zisserman. 2006. "hello! my name is... buffy" – automatic naming of characters in tv video. In *BMVC*.

Florian Eyben, Felix Weninger, Florian Gross, and Björn Schuller. 2013. Recent developments in openSMILE, the munich open-source multimedia feature extractor. In *Proceedings of the 21st ACM International Conference on Multimedia*.

Monica-Laura Haurilet, Makarand Tapaswi, Ziad Al-Halah, and Rainer Stiefelhagen. 2016. Naming tv characters by watching and analyzing dialogs. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*.

Yongtao Hu, Jimmy SJ Ren, Jingwen Dai, Chang Yuan, Li Xu, and Wenping Wang. 2015. Deep multimodal speaker naming. In *Proceedings of the 23rd Annual ACM Conference on Multimedia Conference*.

Ioannis Kapsouras, Anastasios Tefas, Nikos Nikolaidis, and Ioannis Pitas. 2015. Multimodal speaker diarization utilizing face clustering information. In *International Conference on Image and Graphics*.

Vahid Kazemi and Josephine Sullivan. 2014. One millisecond face alignment with an ensemble of regression trees. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Soheil Khorram, John Gideon, Melvin McInnis, and Emily Mower Provost. 2016. Recognition of depression in bipolar disorder: Leveraging cohort and person-specific knowledge. In *Interspeech*.

Davis E King. 2009. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research* .

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*.

Sarah Ita Levitan, Taniya Mishra, and Srinivas Bangalore. 2016. Automatic identification of gender from speech. *Speech Prosody* .

Ying Li, Shrikanth S Narayanan, and C-C Jay Kuo. 2004. Adaptive speaker identification with audiovisual cues for movie content analysis. *Pattern Recognition Letters* .

Chunxi Liu, Shuqiang Jiang, and Qingming Huang. 2008. Naming faces in broadcast news video by image google. In *Proceedings of the 16th ACM international conference on Multimedia*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*.

Benoit Mathieu, Slim Essid, Thomas Fillon, Jacques Prado, and Gal Richard. 2010. Yaafe, an easy to use and efficient audio feature extraction software. In *Proceedings of the 11th International Society for Music Information Retrieval Conference*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NIPS)*.

Seil Na, Sangho Lee, Jisung Kim, and Gunhee Kim. 2017. A read-write memory network for movie story understanding. In *International Conference on Computer Vision (ICCV)*.

O. M. Parkhi, A. Vedaldi, and A. Zisserman. 2015. Deep face recognition. In *British Machine Vision Conference (BMVC)*.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* .

V. Ramanathan, A. Joulin, P. Liang, and L. Fei-Fei. 2014. Linking people with "their" names using coreference resolution. In *IEEE Conference on European Conference on Computer Vision (ECCV)*.

Jimmy Ren, Yongtao Hu, Yu-Wing Tai, Chuan Wang, Li Xu, Wenxiu Sun, and Qiong Yan. 2016. Look, listen and learna multimodal lstm for speaker identification. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Douglas A Reynolds. 2002. An overview of automatic speaker recognition technology. In *Acoustics, speech, and signal processing (ICASSP)*.

Josef Sivic, Mark Everingham, and Andrew Zisserman. 2009. "who are you?" – learning person specific classifiers from video. In *IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems (NIPS)*.

Makarand Tapaswi, Martin Bäuml, and Rainer Stiefelhagen. 2012. "knock! knock! who is it?" probabilistic person identification in tv-series. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Makarand Tapaswi, Martin Bäuml, and Rainer Stiefelhagen. 2014. Storygraphs: Visualizing character interactions as a timeline. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Makarand Tapaswi, Martin Bäuml, and Rainer Stiefelhagen. 2015. Improved weak labels using contextual cues for person identification in videos. In *IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*.

Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2016. Movieqa: Understanding stories in movies through question-answering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Laurens Van Der Maaten. 2014. Accelerating t-sne using tree-based algorithms. *Journal of machine learning research* .

Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. 2016. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters* .

Yi-Fan Zhang, Changsheng Xu, Hanquig Lu, and Yeh-Min Huang. 2009. Character identification in feature-length films using global face-name matching. In *IEEE Transactions on Multimedia*.

Stacking With Auxiliary Features for Visual Question Answering

Nazneen Fatema Rajani
Department of Computer Science
University of Texas at Austin
nrajani@cs.utexas.edu

Raymond J. Mooney
Department of Computer Science
University of Texas at Austin
mooney@cs.utexas.edu

Abstract

Visual Question Answering (VQA) is a well-known and challenging task that requires systems to jointly reason about natural language and vision. Deep learning models in various forms have been the standard for solving VQA. However, some of these VQA models are better at certain types of image-question pairs than other models. Ensembling VQA models intelligently to leverage their diverse expertise is, therefore, advantageous.

Stacking With Auxiliary Features (SWAF) is an intelligent ensembling technique which learns to combine the results of multiple models using features of the current problem as context. We propose four categories of auxiliary features for ensembling for VQA. Three out of the four categories of features can be inferred from an image-question pair and do not require querying the component models. The fourth category of auxiliary features uses model-specific explanations. In this paper, we describe how we use these various categories of auxiliary features to improve performance for VQA. Using SWAF to effectively ensemble three recent systems, we obtain a new state-of-the-art. Our work also highlights the advantages of explainable AI models.

1 Introduction

Visual Question Answering (VQA), the task of addressing open-ended questions about images (Malinowski and Fritz, 2014; Antol et al., 2015), has attracted significant attention in recent years (Andreas et al., 2016a; Goyal et al., 2016; Agrawal et al., 2016; Teney et al., 2017). Given an image and a natural language question about the image, the task is to provide an accurate natural language answer. VQA requires visual and linguistic comprehension, language grounding as well as common-sense knowledge. A variety of methods to address these challenges have been developed in recent years (Fukui et al., 2016; Xu and Saenko, 2016; Lu et al., 2016; Chen et al., 2015). The vision component of a typical VQA system extracts visual features using a deep convolutional neural network (CNN), and the linguistic component encodes the question into a semantic vector using a recurrent neural network (RNN). An answer is then generated conditioned on the visual features and the question vector.

Most VQA systems have a single underlying method that optimizes a specific loss function and do not leverage the advantage of using multiple diverse models. One recent ensembling approach to VQA (Fukui et al., 2016) combined multiple models that use multimodal compact bilinear pooling with attention and achieved state-of-the-art accuracy on the VQA 2016 challenge. However, their ensemble uses simple softmax averaging to combine outputs from multiple systems. Also, their model is pre-trained on the Visual Genome dataset (Krishna et al., 2017) and they concatenate learned word embeddings with pre-trained GloVe vectors (Pennington et al., 2014). Several other deep and non-deep learning approaches for solving VQA have also been proposed (Lu et al., 2016; Zhou et al., 2015; Noh et al., 2016). Although these models perform fairly well on certain image-question (IQ) pairs, they fail spectacularly on certain other IQ pairs. This led us to conclude that the various VQA models have learned to perform well on specific types of questions and images. Therefore, there is an opportunity to combine these models intelligently so as to leverage their diverse strengths.

Ensembling multiple systems is a well known standard approach to improving accuracy in machine learning (Dietterich, 2000). Stacking with Auxiliary Features (SWAF) (Rajani and Mooney, 2017) is a recent ensembling algorithm that learns to combine outputs of multiple systems using fea-

Proceedings of NAACL-HLT 2018, pages 2217–2226
New Orleans, Louisiana, June 1 - 6, 2018.

Q. Is that a frisbee?
A. Yes
Q. Is this a man or a woman?
A. Woman
Q. What color is the frisbee?
A. Red

Q. Is this a romantic spot that couples would like to go?
A. Yes
Q. What time of day is it?
A. Night
Q. How many spires below big ben's clock?
A. 10

Figure 1: Random sample of images with questions and ground truth answers taken from the VQA dataset.

tures of the current problem as context. In this paper, we use SWAF to more effectively combine several VQA models. Traditional stacking (Wolpert, 1992) trains a supervised meta-classifier to appropriately combine multiple system outputs. SWAF further enables the stacker to exploit additional relevant knowledge of both the component systems and the problem by providing *auxiliary features* to the meta-classifier. Our approach extracts features from the IQ pair under consideration, as well as the component models and provides this information to the classifier. The meta-classifier then learns to predict whether a specific generated answer is correct or not.

Explanations attempt to justify a system's predicted output and provide context for their decision that may also help SWAF. We extract *visual* explanations from various deep learning models and use those as auxiliary features for SWAF. Our contributions can be summarized as follows: (a) developing novel auxiliary features that can be inferred from VQA questions and images; (b) extracting visual explanations from several component models for each IQ pair and using those to also generate auxiliary features; and (c) using SWAF to ensemble various VQA models and evaluating ablations of features while comparing our approach extensively to several individual as well as ensemble systems. By effectively ensembling three leading VQA systems with SWAF, we demonstrate state-of-the-art performance.

2 Background and Related Work

VQA is the task of answering a natural language question about the content of an image by returning an appropriate word or phrase. Figure 1 shows a sample of images and questions from the VQA 2016 challenge. The dataset consists of images taken from the MS COCO dataset (Lin et al., 2014) with three questions and answers per image obtained through Mechanical Turk (Antol et al., 2015). Table 1 summarizes the splits in the VQA dataset. Several deep learning models have been developed that combine a computer vision component with a linguistic component in order to solve the VQA challenge. Some of these models also use data-augmentation for pre-training. We discuss the VQA models we use in Section 5.

	Images	Questions
Training	82,783	248,349
Validation	40,504	121,512
Test	81,434	244,302

Table 1: VQA dataset splits.

Stacking With Auxiliary Features (SWAF) is an ensembling technique that combines outputs from multiple systems using their confidence scores and task-relevant features. It has previously been applied effectively to information extraction (Viswanathan et al., 2015), entity linking (Rajani and Mooney, 2016) and ImageNet object detection (Rajani and Mooney, 2017). To the best of our knowledge, there has been no prior work on stacking for VQA, and we are the first to show how model-specific explanations can serve as an auxiliary feature. The auxiliary features that we use are motivated by an analysis of the VQA dataset and also inspired by related work, such as using a Bayesian framework to predict the form of the answer from the question (Kafle and Kanan, 2016).

Deep learning models have been used widely on several vision and language problems. However, they frequently lack transparency and are unable to explain their decisions (Selvaraju et al., 2017). On the other hand, humans can justify their decisions with natural language as well as point to the visual evidence that supports their decision. There are several advantages of having AI systems that can generate explanations that support their predictions (Johns et al., 2015; Agrawal et al., 2016). These advantages have motivated recent work on explainable AI systems, particularly in computer vision (Antol et al., 2015; Goyal et al., 2016; Hendricks et al., 2016; Park et al., 2016). However, there has been no prior work on using explanations for ensembling multiple models or improving performance on a challenging task. In this

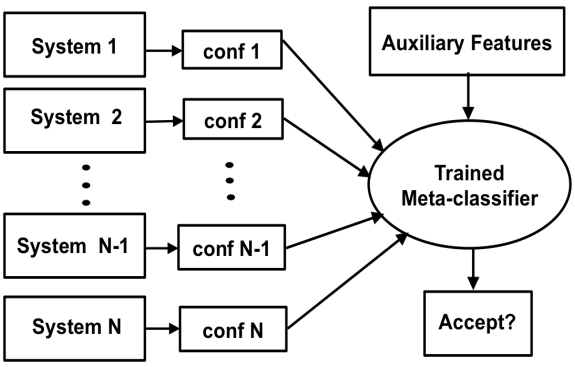


Figure 2: Ensemble Architecture using Stacking with Auxiliary Features. Given an input, the ensemble judges every possible question-answer pair produced by the component systems and determines the final output answer.

paper, we generate visual explanations for three different VQA models and use these explanations to develop auxiliary features that aid in effectively ensembling VQA systems.

3 Stacking With Auxiliary Features (SWAF) for VQA

In stacking, a meta-classifier is learned to combine the outputs of multiple underlying systems (Wolpert, 1992). The stacker learns a classification boundary based on the confidence scores provided by individual systems for each possible output. However, many times the scores produced by systems are not probabilities or not well calibrated and cannot be meaningfully compared. In such circumstances, it is beneficial to also have other reliable auxiliary features, as in the SWAF approach. SWAF provides the meta-classifier additional information, such as features of the current problem and provenance or explanation information about the output from individual systems. This allows SWAF to *learn* which systems do well on which types of problems and when to trust agreements between specific systems. The learned meta-classifier makes a binary decision whether or not to accept a particular output. Figure 2 gives an overview of the SWAF approach.

For stacking VQA systems, we first form unique question-answer pairs across all of the systems' outputs before passing them through the stacker. If a system generates a given output, then we use its probability estimate for that output, otherwise, the confidence is considered zero. If a question-answer pair is classified as correct by the stacker, and if there are other answers that are also classified as correct for the same question, the output with the highest meta-classifier confidence is chosen. For questions that do not have any answer classified as correct by the stacker, we choose the answer with lowest classifier confidence, which means it is least likely to be incorrect. The reason we do this is that the online VQA scorer expects an answer for each question in the test set and penalizes the model for every unanswered question.

The confidence scores along with other auxiliary features form the complete set of features used by the stacker. The auxiliary features are the backbone of the SWAF approach, enabling the stacker to intelligently learn to rely on systems' outputs conditioned on the supporting evidence. We use a total of four different categories of auxiliary features for VQA. Three of these types can be inferred directly from the image-question (IQ) pair and do not require querying the individual models. For the fourth category of auxiliary features, we generate *visual explanations* for the component models and use these to create the explanation auxiliary features. The first three categories of features are discussed below and the fourth category is discussed in the next section.

3.1 Question and Answer Types

Antol et al. (2015) analyzed the VQA data and found that most questions fall into several types based on the first few words (e.g. questions beginning with "What is...", "Is there...", "How many...", or "Does the..."). Using the validation data, we discover such lexical patterns to define a set of question types. The questions were tokenized and a question type was formed by adding one token at a time, up to a maximum of five, to the current substring. The question "What is the color of the vase?" has the following types: "What", "What is", "What is the", "What is the color", "What is the color of". The prefixes that contain at least 500 questions were then retained as types. We added a final type "other" for questions that do not fall into any of the predefined types, resulting in a total of 70 question types. A 70-bit vector is used to encode the question type as a set of auxiliary features.

The original analysis of VQA answers found that they are 38% "yes/no" type and 12% numbers.

There is clearly a pattern in the VQA answers as well and we use the questions to infer some of these patterns. We considered three answer types – "yes/no", "number", and "other". The answer-type auxiliary features are encoded using a one-hot vector. We classify all questions beginning with "Does","Is","Was","Are", and "Has" as "yes/no". Ones beginning with "How many", "What time", "What number" are assigned "number" type. These inferred answer types are not exhaustive but have good coverage. The intuition behind using the question and answer types as auxiliary features is that some VQA models are better than others at handling certain types of questions and/or answers. Making this information available at the time of classification aids the stacker in making a better decision.

3.2 Question Features

We also use a bag-of-words (BOW) representation of the question as auxiliary features. Words that occur at least five times in the validation set were included. The final sparse vector representing a question was normalized by the number of unique words in the question. In this way, we are able to embed the question into a single vector. Goyal et al. (2016) showed that attending to specific words in the question is important in VQA. Including a BOW for the question as auxiliary features equip the stacker to efficiently learn which words are important and can aid in classifying answers.

3.3 Image Features

We also used "deep visual features" of the image as additional auxiliary features. Specifically, we use the $4,096$ features from VGGNet's (Simonyan and Zisserman, 2015) $fc7$ layer . This creates an embedding of the image in a single vector which is then used by the stacker. Using such image features enables the stacker to learn to rely on systems that are good at identifying answers for particular types of images. Recall that the individual VQA models fuse an embedding of the image along with an embedding of the question. By using the question and image embeddings at the meta-classifier level, the stacker learns to discriminate between the component models based on a deeper representation of the IQ pair.

4 Using Explanations

Recently, there has been work on analyzing regions of an image that deep-learning models focus on when making decisions (Goyal et al., 2016; Hendricks et al., 2016; Park et al., 2016). This work shows that deep-learning models attend to relevant parts of the image when making a decision. For VQA, the parts of images that the models focus on can be thought of as *visual* explanations for answering the question. We use these visual explanations to construct auxiliary features for SWAF. The idea behind using explanation features is that they enable the stacker to learn to trust the agreement between systems when they also agree on the heat-map explanation by "looking" at the right region of the image when generating an answer.

4.1 Generating Explanations

We use the GradCAM algorithm (Selvaraju et al., 2017) to generate model-specific explanatory heat-maps for each IQ pair. This approach generates a class-discriminative localization-map for a given model based on its respective predicted output class in the following way. First, the gradient of the score y^c for the predicted class c is computed before the softmax layer with respect to the feature maps A^k of a convolutional layer. Then, the gradients flowing back are global average pooled to obtain the neuron importance weights.

$$w_k^c = \overbrace{\frac{1}{Z}\sum_i\sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{backprop gradients}}$$

The above weights capture the importance of a convolutional feature map k for the output class c, where Z is the total number of pixels in the feature map. A ReLU over the weighted combination of the feature maps results in the required localization-map for the output class as follows:

$$H^c = ReLU(\sum_k w_k^c A^k)$$

For each of the component VQA models, we generate the localization-map to be used as auxiliary features for ensembling. Figure 3 shows a sample of IQ pairs from the VQA dataset and their respective heat-maps generated for three VQA models.

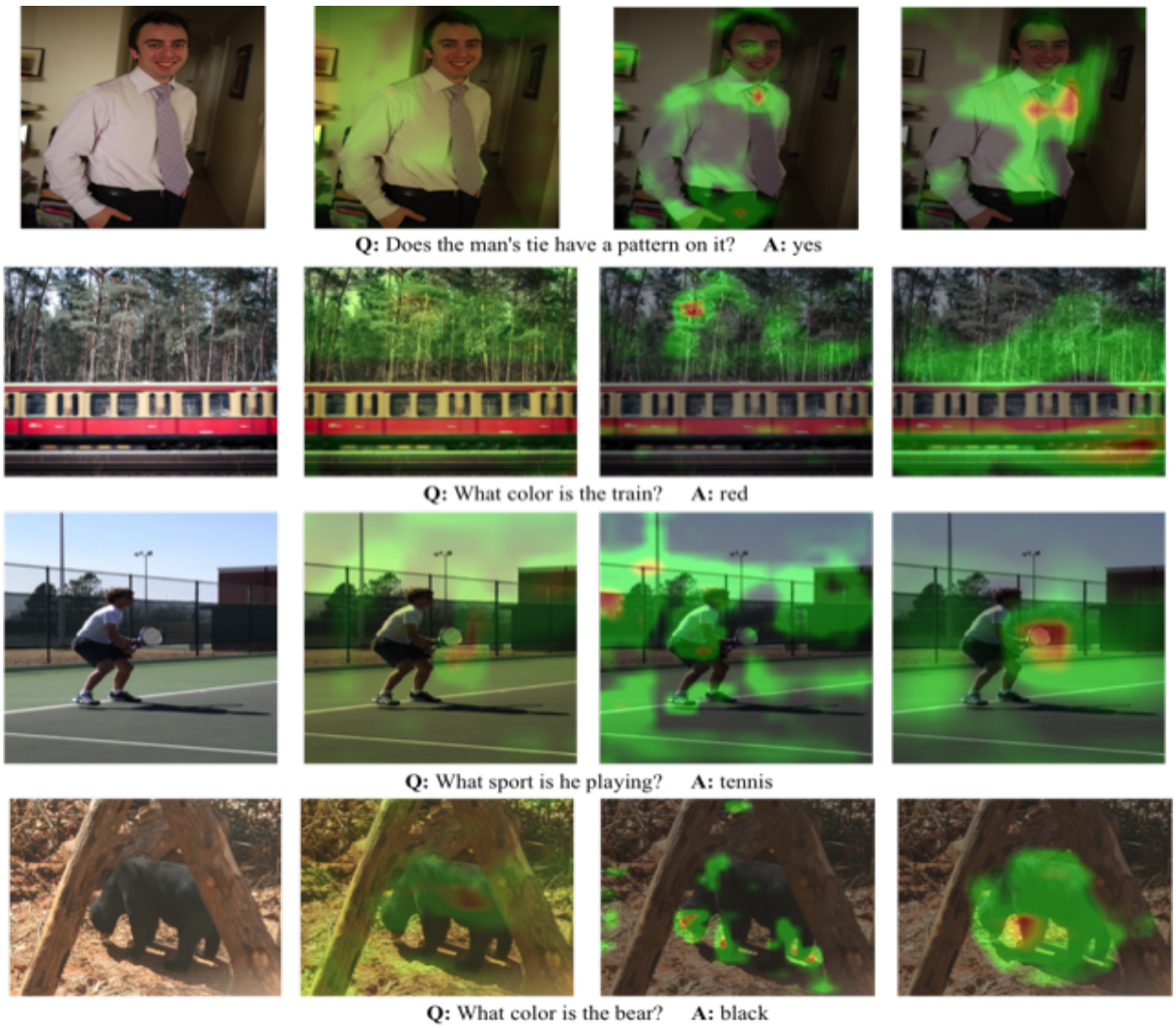


Figure 3: Each row from left to right shows an image-question pair from the VQA dataset along with localization-maps overlaid on the image generated by the LSTM, HieCoAtt and MCB models respectively. The answers shown are those predicted by our ensemble.

4.2 Explanation as Auxiliary Features

The localization-map generated by each VQA model serves as a visual explanation for the predicted output of that model. We compare agreement between the localization-maps of the individual models to generate auxiliary features for SWAF. We take the absolute gray-scale value of the localization-maps in of each model and compute their mean rank-correlation with the localization-map of every other model. We rank the pixels according to their spatial attention and then compute the correlation between the two ranked lists. The rank correlation protocol has been used in the past to compare machine-generated and human attention-maps as described by Das et al. (2016). We also experimented with using the Earth Mover's Distance (EMD) in place of the rank-order correlation metric, as discussed in Section 6. We compare the localization-maps of each pair of VQA models, generating $\binom{n}{2}$ "explanation agreement" auxiliary features for SWAF, where n is the total number of models.

5 Component VQA Systems

We use SWAF to combine three diverse VQA systems such that the final ensemble performs better than any individual component model even on questions with a low agreement. The three component models are trained on the VQA training set. Each of the three models is described below.

5.1 Long Short-Term Memory (LSTM)

The LSTM model (Antol et al., 2015) is one of the original baseline models used to establish a

benchmark for the VQA dataset. A VGGNet (Simonyan and Zisserman, 2015) is used to obtain embeddings for the image which is combined with an LSTM (Hochreiter and Schmidhuber, 1997) embedding of each question. An LSTM with two hidden layers is used to obtain a $2,048$-dimensional embedding of the question, followed by a fully-connected layer with *tanh* non-linearity to transform the embedding to $1,024$ dimensions. The l_2 normalized activations from the last hidden layer of VGGNet are used as a $4,096$ dimensional image embedding. The image embedding is first transformed to $1,024$ dimensions by a fully-connected layer with *tanh* nonlinearity to match the dimensionality of the LSTM embedding of the question. The transformed image and LSTM embeddings are then fused via element-wise multiplication.

5.2 Hierarchical Question-Image Co-Attention (HieCoAtt)

The idea behind the HieCoAtt model is that in addition to using visual attention to focus on where to look, it is equally important to model what words to attend to in the question (question-attention) (Lu et al., 2016). This model jointly reasons about the visual and language components using "co-attention". Question attention is modeled using a hierarchical architecture at word, phrase, and question levels.

HieCoAtt uses two types of co-attention – parallel and alternating. Parallel co-attention attends to the image and question simultaneously by calculating the similarity between image and question features at all pairs of image-locations and question-locations. Alternating co-attention sequentially alternates between generating image and question attention by attending to the image based on the question summary vector and then attending to the question based on the attended image features.

5.3 Multimodal Compact Bilinear pooling (MCB)

The MCB model combines the vision and language vector representations using an outer product instead of the traditional approach of using concatenation or element-wise product or sum of the two vectors (Fukui et al., 2016). Bilinear pooling computes the outer product between two vectors which, in contrast to the element-wise product, allows a multiplicative interaction between all elements of both vectors. To overcome the challenge of high dimensionality due to the outer product, the authors adopt the idea of using Multimodal Compact Bilinear pooling (MCB) (Gao et al., 2016) to efficiently and expressively combine multimodal features.

The MCB model extracts representations for the image using the 152-layer Residual Network (He et al., 2016) and an LSTM (Hochreiter and Schmidhuber, 1997) embedding of the question. The two vector are pooled using MCB and the answer is obtained by treating the problem as a multi-class classification problem with $3,000$ possible classes. The best MCB model is an ensemble of seven attention models and uses data-augmentation for pre-training along with pre-trained GloVe word embeddings. The best MCB model won the VQA 2016 challenge by obtaining the best performance on the test set.

6 Experimental Results and Discussion

We present experimental results on the VQA challenge using the SWAF approach and compare it to various baselines, individual and ensemble VQA models, as well as ablations of our SWAF algorithm on the standard VQA test set. In addition to the three data splits given in Table 1, the VQA challenge divides the test set into *test-dev* and *test-standard*. Evaluation on either split requires submitting the output to the competition's online server.[1] However, there are fewer restrictions on the number of submissions that can be made to the *test-dev* compared to the *test-standard*. The *test-dev* is a subset of the standard test set consisting of randomly selected $60,864$ (25%) questions. We use the *test-dev* set to tune the parameters of the meta-classifier. All the individual VQA models that we ensemble are trained only on the VQA training set and the SWAF meta-classifier is trained on the VQA validation set.

For the meta-classifier, we use a $L1$-regularized SVM classifier for generic stacking and stacking with only question/answer types as auxiliary features. For the question, image, and explanation features, we found that a neural network with two hidden layers works best. The first hidden layer is fully connected and the second has approximately half the number of neurons as the first layer. The question and image features are high-dimensional and therefore a neural network classifier worked

[1] www.visualqa.org/challenge.html

Method	All	Yes/No	Number	Other
DPPNet (Noh et al., 2016)	57.36	80.28	36.92	42.24
iBOWIMG (Zhou et al., 2015)	55.72	76.55	35.03	42.62
NMNs (Andreas et al., 2016b)	58.70	81.20	37.70	44.00
LSTM (Antol et al., 2015)	58.20	80.60	36.50	43.70
HieCoAtt (Lu et al., 2016)	61.80	79.70	38.70	51.70
MCB (Single system) (Fukui et al., 2016)	62.56	80.68	35.59	52.93
MCB (Ensemble) (Fukui et al., 2016)	66.50	**83.20**	39.50	58.00
Voting (MCB + HieCoAtt + LSTM)	60.31	80.22	34.92	48.83
Stacking	63.12	81.61	36.07	53.77
+ Q/A type features	65.25	82.01	36.50	57.15
+ Question features	65.50	82.26	38.21	57.35
+ Image features	65.54	82.28	38.63	57.32
+ Explanation features	**67.26**	82.62	**39.50**	**58.34**

Table 2: Accuracy results on the VQA *test-standard* set. The first block shows performance of a VQA model that use external data for pre-training, the second block shows single system VQA models, the third block shows an ensemble VQA model that also uses external data for pre-training, and the fourth block shows ensemble VQA models.

well. We found that using late fusion (Karpathy et al., 2014) to combine the auxiliary features for the neural network classifier worked slightly better. We used Keras with Tensorflow back-end (Chollet, 2015) for implementing the network. We compare our approach to a voting baseline that returns the answer with maximum agreement, with ties broken in the favor of systems with higher confidence scores. We also compare against other state-of-the-art VQA systems not used in our ensemble: iBowIMG (Zhou et al., 2015), DPPNet (Noh et al., 2016) and the Neural Module Networks (NMNs) (Andreas et al., 2016b).

The iBowIMG concatenates the image features with the bag-of-word question embedding and feeds them into a softmax classifier to predict the answer, resulting in performance comparable to other models that use deep or recursive neural networks. The iBowIMG beats most VQA models considered in their paper. The DPPNet, on the other hand, learns a CNN with some parameters predicted from a separate parameter prediction network. Their parameter prediction network uses a Gated Recurrent Unit (GRU) to generate a question representation and maps the predicted weights to a CNN via hashing. The DPPNet uses external data (data-augmentation) in addition to the VQA dataset to pre-train the GRU. Another well-known VQA model is the Neural Module Network (NMN) that generates a neural network on the fly for each individual image and question. This is done through choosing from various sub-modules based on the question and composing these to generate the neural network, *e.g.*, the `find[x]` module outputs an attention map for detecting `x`. To arrange the modules, the question is first parsed into a symbolic expression and using these expressions, modules are composed into a sequence to answer the query. The whole system is trained end-to-end through backpropagation.

The VQA evaluation server, along with reporting accuracies on the full question set, also reports a break-down of accuracy across three answer categories. The image-question (IQ) pairs that have answer type as "yes/no", those that have "number" as their answer type and finally those that do not belong to either of the first two categories are classified as "other". Table 2 shows the full and category-wise accuracies. All scores for the stacking models were obtained using the VQA *test-standard* server. The table shows results for both single system and ensemble MCB models. We used the single system MCB model as a component in our ensemble. The ensemble MCB system, however, was the top-ranked system in the VQA 2016 challenge and it is pre-trained on the Visual Genome dataset (Krishna et al., 2017) as well as uses pre-trained GloVe vectors (Pennington et al., 2014). On the other hand, our ensemble system does not use any external data and consists

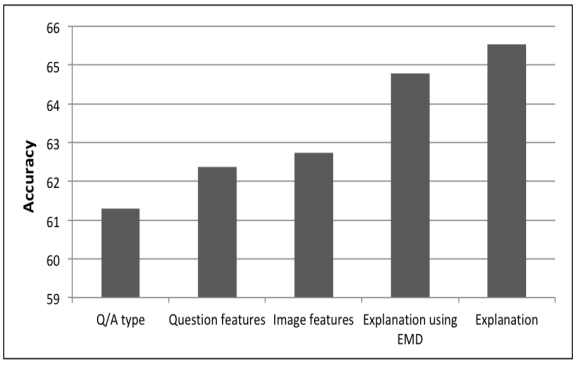


Figure 4: Results for auxiliary feature ablations on the VQA *test-dev* set. The `x-axis` indicates the feature set that was ablated from the final ensemble.

of only three component models.

The SWAF approach obtains a new state-of-the-art result on the VQA task. The vanilla stacking approach itself beats the best *individual* model and adding the auxiliary features further boosts the performance. Our SWAF model that uses all three sets of auxiliary features related to IQ pairs does particularly well on the more difficult "other" answer category, indicating that the auxiliary features provide crucial information at classification time. To further analyze the SWAF results, we performed experiments with ablations of the auxiliary features. Figure 4 shows the results on the *test-dev* set obtained when ablating each of the auxiliary feature sets. We observe that deleting the Q/A type decreased performance the most and deleting the explanation features decreased performance the least. This indicates that the Q/A type features are the most informative and the explanation features are the least informative for deciding the correct answer.

The voting baseline does not perform very well even though it is able to beat one of the component models. The SWAF ablation results clearly indicate that there is an advantage to using each type of auxiliary feature. Each of the auxiliary feature sets contributes to the final ensemble's performance, which is clear from Table 2. The voting and the "vanilla stacking" ensembles do not perform as well as SWAF. This leads us to conclude that the performance gain is actually obtained from using the auxiliary features.

In particular, using explanations generated by various deep learning models as auxiliary features improved performance. We observed that the localization-maps generated were fairly noisy, as is evident from Figure 3. Although the individual component systems agreed on an answer for many of the IQ pairs, the regions of the image they attend to varied significantly. However, the rank correlation metric in the auxiliary features made the localization-maps useful for ensembling. This is because, when training on the validation set, the stacker *learns* how to weight the auxiliary features, including those obtained using localization-maps. In this way, it learns to trust only the localization-maps that are actually useful. We also observed that there was a high positive correlation between the localization-maps generated by the HieCoAtt and MCB models, followed by the LSTM and MCB models, and then the LSTM and HieCoAtt models with several of the maps even negatively correlated between the last two models.

We also experimented with using Earth Mover's Distance (EMD) to compare heat-maps and found that it worked even better than rank-order correlation; however, it came at a cost of high computational complexity ($\mathcal{O}(n^3)$ vs. $\mathcal{O}(n)$). Figure 4 shows the difference in performance obtained when explanation features calculated using either EMD or rank-order correlation are ablated from the final ensemble. Clearly, using EMD to compare explanation maps has more impact on the system's accuracy. Consistent with previous findings (Bylinskii et al., 2018), our results confirm that EMD provides a finer-grained comparison between localization maps. Overall, our work shows that the utility of explanations is not limited to just developing human trust and making models more transparent. Explanations can also be used to improve performance on a challenging task.

7 Conclusions and Future Work

We have presented results for using stacking with auxiliary features (SWAF) to ensemble VQA systems. We proposed four different categories of auxiliary features, three of which can be inferred from an image-question pair. We showed that our model trained on these auxiliary features outperforms the individual component systems as well as other baselines to obtain a new state-of-the-art for VQA. For the fourth category of features, we have proposed and evaluated the novel idea of using explanations to improve ensembling of multiple systems. We demonstrated how visual explanations for VQA (represented as localization-maps) can be used to aid stacking with auxiliary

features. This approach effectively utilizes information on the degree to which systems agree on the *explanation* of their answers. We showed that the combination of all of these categories of auxiliary features, including explanation, gives the best results.

We believe that integrating explanation with ensembling has a two-fold advantage. First, as discussed in this paper, explanations can be used to improve the accuracy of an ensemble. Second, explanations from the component systems could be used to build an explanation for the overall ensemble. That is, by combining multiple component explanations, SWAF could also produce more comprehensible results. Therefore, in the future, we would like to focus on explaining the results of an ensemble. Another issue we plan to explore is using textual explanations (Park et al., 2016) for VQA. We believe that the words in the question to which a system attends can also be used to improve ensembling. Finally, we hope to apply our approach to additional problems beyond VQA.

Acknowledgement

This research was supported by the DARPA XAI program under the AFRL grant.

References

Aishwarya Agrawal, Dhruv Batra, and Devi Parikh. 2016. Analyzing the Behavior of Visual Question Answering Models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP2016)*.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016a. Learning to compose neural networks for question answering. In *Proceedings of the Conference on Natural language learning (NAACL2016)*. pages 1545–1554.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016b. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR2016)*. pages 39–48.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual Question Answering. In *The IEEE International Conference on Computer Vision (ICCV2015)*.

Zoya Bylinskii, Tilke Judd, Aude Oliva, Antonio Torralba, and Frédo Durand. 2018. What do different evaluation metrics tell us about saliency models? *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI2018)* .

Kan Chen, Jiang Wang, Liang-Chieh Chen, Haoyuan Gao, Wei Xu, and Ram Nevatia. 2015. ABC-CNN: An attention based convolutional neural network for Visual Question Answering. *arXiv preprint arXiv:1511.05960* .

Franois Chollet. 2015. Keras. https://github.com/fchollet/keras.

Abhishek Das, Harsh Agrawal, C. Lawrence Zitnick, Devi Parikh, and Dhruv Batra. 2016. Human Attention in Visual Question Answering: Do Humans and Deep Networks Look at the Same Regions? In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP2016)*.

T. Dietterich. 2000. Ensemble methods in machine learning. In J. Kittler and F. Roli, editors, *First International Workshop on Multiple Classifier Systems, Lecture Notes in Computer Science*. Springer-Verlag, pages 1–15.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal Compact Bilinear Pooling for Visual Question Answering and Visual Grounding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP2016)*.

Yang Gao, Oscar Beijbom, Ning Zhang, and Trevor Darrell. 2016. Compact bilinear pooling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR2016)*. pages 317–326.

Yash Goyal, Akrit Mohapatra, Devi Parikh, and Dhruv Batra. 2016. Towards Transparent AI Systems: Interpreting Visual Question Answering Models. In *International Conference on Machine Learning (ICML) Workshop on Visualization for Deep Learning, 2016*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR2016)*. pages 770–778.

Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. 2016. Generating visual explanations. In *Proceedings of the European Conference on Computer Vision (ECCV2016)*. Springer, pages 3–19.

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

Edward Johns, Oisin Mac Aodha, and Gabriel J Brostow. 2015. Becoming the expert-interactive multi-class machine teaching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR2015)*. pages 2616–2624.

Kushal Kafle and Christopher Kanan. 2016. Answer-type prediction for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR2016)*.

Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. 2014. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR2014)*. pages 1725–1732.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. 2017. Visual Genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision (IJCV)* 123(1):32–73.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV2014)*. Springer, pages 740–755.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical Question-Image Co-Attention for Visual Question Answering. In *Advances in Neural Information Processing Systems (NIPS2016)*. pages 289–297.

Mateusz Malinowski and Mario Fritz. 2014. A multi-world approach to question answering about real-world scenes based on uncertain input. In *Advances in Neural Information Processing Systems (NIPS2014)*. pages 1682–1690.

Hyeonwoo Noh, Paul Hongsuck Seo, and Bohyung Han. 2016. Image question answering using convolutional neural network with dynamic parameter prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR2016)*. pages 30–38.

Dong Huk Park, Lisa Anne Hendricks, Zeynep Akata, Bernt Schiele, Trevor Darrell, and Marcus Rohrbach. 2016. Attentive explanations: Justifying decisions and pointing to the evidence. *arXiv preprint arXiv:1612.04757* .

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP2014)*. pages 1532–1543.

Nazneen Fatema Rajani and Raymond J. Mooney. 2016. Combining Supervised and Unsupervised Ensembles for Knowledge Base Population. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP2016)*.

Nazneen Fatema Rajani and Raymond J. Mooney. 2017. Stacking With Auxiliary Features. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI2017)*. Melbourne, Australia.

Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *The IEEE International Conference on Computer Vision (ICCV2017)*.

Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations (ICLR2015)*.

Damien Teney, Peter Anderson, Xiaodong He, and Anton van den Hengel. 2017. Tips and tricks for visual question answering: Learnings from the 2017 challenge. *arXiv preprint arXiv:1708.02711* .

Vidhoon Viswanathan, Nazneen Fatema Rajani, Yinon Bentor, and Raymond J. Mooney. 2015. Stacked Ensembles of Information Extractors for Knowledge-Base Population. In *Association for Computational Linguistics (ACL2015)*. Beijing, China, pages 177–187.

David H. Wolpert. 1992. Stacked Generalization. *Neural Networks* 5:241–259.

Huijuan Xu and Kate Saenko. 2016. Ask, Attend and Answer: Exploring question-guided spatial attention for Visual Question Answering. In *European Conference on Computer Vision (ECCV2016)*. Springer, pages 451–466.

Bolei Zhou, Yuandong Tian, Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. 2015. Simple baseline for Visual Question Answering. *arXiv preprint arXiv:1512.02167* .

Deep contextualized word representations

Matthew E. Peters[†], **Mark Neumann**[†], **Mohit Iyyer**[†], **Matt Gardner**[†],
{matthewp,markn,mohiti,mattg}@allenai.org

Christopher Clark[*], **Kenton Lee**[*], **Luke Zettlemoyer**[†*]
{csquared,kentonl,lsz}@cs.washington.edu

[†]Allen Institute for Artificial Intelligence
[*]Paul G. Allen School of Computer Science & Engineering, University of Washington

Abstract

We introduce a new type of *deep contextualized* word representation that models both (1) complex characteristics of word use (e.g., syntax and semantics), and (2) how these uses vary across linguistic contexts (i.e., to model polysemy). Our word vectors are learned functions of the internal states of a deep bidirectional language model (biLM), which is pre-trained on a large text corpus. We show that these representations can be easily added to existing models and significantly improve the state of the art across six challenging NLP problems, including question answering, textual entailment and sentiment analysis. We also present an analysis showing that exposing the deep internals of the pre-trained network is crucial, allowing downstream models to mix different types of semi-supervision signals.

1 Introduction

Pre-trained word representations (Mikolov et al., 2013; Pennington et al., 2014) are a key component in many neural language understanding models. However, learning high quality representations can be challenging. They should ideally model both (1) complex characteristics of word use (e.g., syntax and semantics), and (2) how these uses vary across linguistic contexts (i.e., to model polysemy). In this paper, we introduce a new type of *deep contextualized* word representation that directly addresses both challenges, can be easily integrated into existing models, and significantly improves the state of the art in every considered case across a range of challenging language understanding problems.

Our representations differ from traditional word type embeddings in that each token is assigned a representation that is a function of the entire input sentence. We use vectors derived from a bidirectional LSTM that is trained with a coupled language model (LM) objective on a large text corpus. For this reason, we call them ELMo (Embeddings from Language Models) representations. Unlike previous approaches for learning contextualized word vectors (Peters et al., 2017; McCann et al., 2017), ELMo representations are deep, in the sense that they are a function of all of the internal layers of the biLM. More specifically, we learn a linear combination of the vectors stacked above each input word for each end task, which markedly improves performance over just using the top LSTM layer.

Combining the internal states in this manner allows for very rich word representations. Using intrinsic evaluations, we show that the higher-level LSTM states capture context-dependent aspects of word meaning (e.g., they can be used without modification to perform well on supervised word sense disambiguation tasks) while lower-level states model aspects of syntax (e.g., they can be used to do part-of-speech tagging). Simultaneously exposing all of these signals is highly beneficial, allowing the learned models select the types of semi-supervision that are most useful for each end task.

Extensive experiments demonstrate that ELMo representations work extremely well in practice. We first show that they can be easily added to existing models for six diverse and challenging language understanding problems, including textual entailment, question answering and sentiment analysis. The addition of ELMo representations alone significantly improves the state of the art in every case, including up to 20% relative error reductions. For tasks where direct comparisons are possible, ELMo outperforms CoVe (McCann et al., 2017), which computes contextualized representations using a neural machine translation encoder. Finally, an analysis of both ELMo and CoVe reveals that deep representations outperform

Proceedings of NAACL-HLT 2018, pages 2227–2237
New Orleans, Louisiana, June 1 - 6, 2018.

those derived from just the top layer of an LSTM. Our trained models and code are publicly available, and we expect that ELMo will provide similar gains for many other NLP problems.[1]

2 Related work

Due to their ability to capture syntactic and semantic information of words from large scale unlabeled text, pretrained word vectors (Turian et al., 2010; Mikolov et al., 2013; Pennington et al., 2014) are a standard component of most state-of-the-art NLP architectures, including for question answering (Liu et al., 2017), textual entailment (Chen et al., 2017) and semantic role labeling (He et al., 2017). However, these approaches for learning word vectors only allow a single context-independent representation for each word.

Previously proposed methods overcome some of the shortcomings of traditional word vectors by either enriching them with subword information (e.g., Wieting et al., 2016; Bojanowski et al., 2017) or learning separate vectors for each word sense (e.g., Neelakantan et al., 2014). Our approach also benefits from subword units through the use of character convolutions, and we seamlessly incorporate multi-sense information into downstream tasks without explicitly training to predict predefined sense classes.

Other recent work has also focused on learning context-dependent representations. `context2vec` (Melamud et al., 2016) uses a bidirectional Long Short Term Memory (LSTM; Hochreiter and Schmidhuber, 1997) to encode the context around a pivot word. Other approaches for learning contextual embeddings include the pivot word itself in the representation and are computed with the encoder of either a supervised neural machine translation (MT) system (CoVe; McCann et al., 2017) or an unsupervised language model (Peters et al., 2017). Both of these approaches benefit from large datasets, although the MT approach is limited by the size of parallel corpora. In this paper, we take full advantage of access to plentiful monolingual data, and train our biLM on a corpus with approximately 30 million sentences (Chelba et al., 2014). We also generalize these approaches to deep contextual representations, which we show work well across a broad range of diverse NLP tasks.

[1] `http://allennlp.org/elmo`

Previous work has also shown that different layers of deep biRNNs encode different types of information. For example, introducing multi-task syntactic supervision (e.g., part-of-speech tags) at the lower levels of a deep LSTM can improve overall performance of higher level tasks such as dependency parsing (Hashimoto et al., 2017) or CCG super tagging (Søgaard and Goldberg, 2016). In an RNN-based encoder-decoder machine translation system, Belinkov et al. (2017) showed that the representations learned at the first layer in a 2-layer LSTM encoder are better at predicting POS tags then second layer. Finally, the top layer of an LSTM for encoding word context (Melamud et al., 2016) has been shown to learn representations of word sense. We show that similar signals are also induced by the modified language model objective of our ELMo representations, and it can be very beneficial to learn models for downstream tasks that mix these different types of semi-supervision.

Dai and Le (2015) and Ramachandran et al. (2017) pretrain encoder-decoder pairs using language models and sequence autoencoders and then fine tune with task specific supervision. In contrast, after pretraining the biLM with unlabeled data, we fix the weights and add additional task-specific model capacity, allowing us to leverage large, rich and universal biLM representations for cases where downstream training data size dictates a smaller supervised model.

3 ELMo: Embeddings from Language Models

Unlike most widely used word embeddings (Pennington et al., 2014), ELMo word representations are functions of the entire input sentence, as described in this section. They are computed on top of two-layer biLMs with character convolutions (Sec. 3.1), as a linear function of the internal network states (Sec. 3.2). This setup allows us to do semi-supervised learning, where the biLM is pretrained at a large scale (Sec. 3.4) and easily incorporated into a wide range of existing neural NLP architectures (Sec. 3.3).

3.1 Bidirectional language models

Given a sequence of N tokens, $(t_1, t_2, ..., t_N)$, a forward language model computes the probability of the sequence by modeling the probability of to-

ken t_k given the history $(t_1, ..., t_{k-1})$:

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k \mid t_1, t_2, \ldots, t_{k-1}).$$

Recent state-of-the-art neural language models (Józefowicz et al., 2016; Melis et al., 2017; Merity et al., 2017) compute a context-independent token representation $\mathbf{x}_k^{LM}$ (via token embeddings or a CNN over characters) then pass it through L layers of forward LSTMs. At each position k, each LSTM layer outputs a context-dependent representation $\overrightarrow{\mathbf{h}}_{k,j}^{LM}$ where $j = 1, \ldots, L$. The top layer LSTM output, $\overrightarrow{\mathbf{h}}_{k,L}^{LM}$, is used to predict the next token t_{k+1} with a Softmax layer.

A backward LM is similar to a forward LM, except it runs over the sequence in reverse, predicting the previous token given the future context:

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k \mid t_{k+1}, t_{k+2}, \ldots, t_N).$$

It can be implemented in an analogous way to a forward LM, with each backward LSTM layer j in a L layer deep model producing representations $\overleftarrow{\mathbf{h}}_{k,j}^{LM}$ of t_k given $(t_{k+1}, \ldots, t_N)$.

A biLM combines both a forward and backward LM. Our formulation jointly maximizes the log likelihood of the forward and backward directions:

$$\sum_{k=1}^{N} \Big(\log p(t_k \mid t_1, \ldots, t_{k-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) + \log p(t_k \mid t_{k+1}, \ldots, t_N; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s) \Big).$$

We tie the parameters for both the token representation (Θ_x) and Softmax layer (Θ_s) in the forward and backward direction while maintaining separate parameters for the LSTMs in each direction. Overall, this formulation is similar to the approach of Peters et al. (2017), with the exception that we share some weights between directions instead of using completely independent parameters. In the next section, we depart from previous work by introducing a new approach for learning word representations that are a linear combination of the biLM layers.

3.2 ELMo

ELMo is a task specific combination of the intermediate layer representations in the biLM. For each token t_k, a L-layer biLM computes a set of $2L + 1$ representations

$$\begin{aligned} R_k &= \{\mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} \mid j = 1, \ldots, L\} \\ &= \{\mathbf{h}_{k,j}^{LM} \mid j = 0, \ldots, L\}, \end{aligned}$$

where $\mathbf{h}_{k,0}^{LM}$ is the token layer and $\mathbf{h}_{k,j}^{LM} = [\overrightarrow{\mathbf{h}}_{k,j}^{LM}; \overleftarrow{\mathbf{h}}_{k,j}^{LM}]$, for each biLSTM layer.

For inclusion in a downstream model, ELMo collapses all layers in R into a single vector, $\mathbf{ELMo}_k = E(R_k; \mathbf{\Theta}_e)$. In the simplest case, ELMo just selects the top layer, $E(R_k) = \mathbf{h}_{k,L}^{LM}$, as in TagLM (Peters et al., 2017) and CoVe (McCann et al., 2017). More generally, we compute a task specific weighting of all biLM layers:

$$\mathbf{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}. \quad (1)$$

In (1), $\mathbf{s}^{task}$ are softmax-normalized weights and the scalar parameter γ^{task} allows the task model to scale the entire ELMo vector. γ is of practical importance to aid the optimization process (see supplemental material for details). Considering that the activations of each biLM layer have a different distribution, in some cases it also helped to apply layer normalization (Ba et al., 2016) to each biLM layer before weighting.

3.3 Using biLMs for supervised NLP tasks

Given a pre-trained biLM and a supervised architecture for a target NLP task, it is a simple process to use the biLM to improve the task model. We simply run the biLM and record all of the layer representations for each word. Then, we let the end task model learn a linear combination of these representations, as described below.

First consider the lowest layers of the supervised model without the biLM. Most supervised NLP models share a common architecture at the lowest layers, allowing us to add ELMo in a consistent, unified manner. Given a sequence of tokens $(t_1, \ldots, t_N)$, it is standard to form a context-independent token representation $\mathbf{x}_k$ for each token position using pre-trained word embeddings and optionally character-based representations. Then, the model forms a context-sensitive representation $\mathbf{h}_k$, typically using either bidirectional RNNs, CNNs, or feed forward networks.

To add ELMo to the supervised model, we first freeze the weights of the biLM and then

concatenate the ELMo vector $\mathbf{ELMo}_k^{task}$ with $\mathbf{x}_k$ and pass the ELMo enhanced representation $[\mathbf{x}_k; \mathbf{ELMo}_k^{task}]$ into the task RNN. For some tasks (e.g., SNLI, SQuAD), we observe further improvements by also including ELMo at the output of the task RNN by introducing another set of output specific linear weights and replacing $\mathbf{h}_k$ with $[\mathbf{h}_k; \mathbf{ELMo}_k^{task}]$. As the remainder of the supervised model remains unchanged, these additions can happen within the context of more complex neural models. For example, see the SNLI experiments in Sec. 4 where a bi-attention layer follows the biLSTMs, or the coreference resolution experiments where a clustering model is layered on top of the biLSTMs.

Finally, we found it beneficial to add a moderate amount of dropout to ELMo (Srivastava et al., 2014) and in some cases to regularize the ELMo weights by adding $\lambda\|\mathbf{w}\|_2^2$ to the loss. This imposes an inductive bias on the ELMo weights to stay close to an average of all biLM layers.

3.4 Pre-trained bidirectional language model architecture

The pre-trained biLMs in this paper are similar to the architectures in Józefowicz et al. (2016) and Kim et al. (2015), but modified to support joint training of both directions and add a residual connection between LSTM layers. We focus on large scale biLMs in this work, as Peters et al. (2017) highlighted the importance of using biLMs over forward-only LMs and large scale training.

To balance overall language model perplexity with model size and computational requirements for downstream tasks while maintaining a purely character-based input representation, we halved all embedding and hidden dimensions from the single best model `CNN-BIG-LSTM` in Józefowicz et al. (2016). The final model uses $L = 2$ biLSTM layers with 4096 units and 512 dimension projections and a residual connection from the first to second layer. The context insensitive type representation uses 2048 character n-gram convolutional filters followed by two highway layers (Srivastava et al., 2015) and a linear projection down to a 512 representation. As a result, the biLM provides three layers of representations for each input token, including those outside the training set due to the purely character input. In contrast, traditional word embedding methods only provide one layer of representation for tokens in a fixed vocabulary.

After training for 10 epochs on the 1B Word Benchmark (Chelba et al., 2014), the average forward and backward perplexities is 39.7, compared to 30.0 for the forward `CNN-BIG-LSTM`. Generally, we found the forward and backward perplexities to be approximately equal, with the backward value slightly lower.

Once pretrained, the biLM can compute representations for any task. In some cases, fine tuning the biLM on domain specific data leads to significant drops in perplexity and an increase in downstream task performance. This can be seen as a type of domain transfer for the biLM. As a result, in most cases we used a fine-tuned biLM in the downstream task. See supplemental material for details.

4 Evaluation

Table 1 shows the performance of ELMo across a diverse set of six benchmark NLP tasks. In every task considered, simply adding ELMo establishes a new state-of-the-art result, with relative error reductions ranging from 6 - 20% over strong base models. This is a very general result across a diverse set model architectures and language understanding tasks. In the remainder of this section we provide high-level sketches of the individual task results; see the supplemental material for full experimental details.

Question answering The Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016) contains 100K+ crowd sourced question-answer pairs where the answer is a span in a given Wikipedia paragraph. Our baseline model (Clark and Gardner, 2017) is an improved version of the Bidirectional Attention Flow model in Seo et al. (BiDAF; 2017). It adds a self-attention layer after the bidirectional attention component, simplifies some of the pooling operations and substitutes the LSTMs for gated recurrent units (GRUs; Cho et al., 2014). After adding ELMo to the baseline model, test set F_1 improved by 4.7% from 81.1% to 85.8%, a 24.9% relative error reduction over the baseline, and improving the overall single model state-of-the-art by 1.4%. A 11 member ensemble pushes F_1 to 87.4, the overall state-of-the-art at time of submission to the leaderboard.[2] The increase of 4.7% with ELMo is also significantly larger then the 1.8% improvement from adding CoVe to a baseline model (McCann et al., 2017).

[2] As of November 17, 2017.

TASK	PREVIOUS SOTA		OUR BASELINE	ELMO + BASELINE	INCREASE (ABSOLUTE/ RELATIVE)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

Table 1: Test set comparison of ELMo enhanced neural models with state-of-the-art single model baselines across six benchmark NLP tasks. The performance metric varies across tasks – accuracy for SNLI and SST-5; F_1 for SQuAD, SRL and NER; average F_1 for Coref. Due to the small test sizes for NER and SST-5, we report the mean and standard deviation across five runs with different random seeds. The "increase" column lists both the absolute and relative improvements over our baseline.

Textual entailment Textual entailment is the task of determining whether a "hypothesis" is true, given a "premise". The Stanford Natural Language Inference (SNLI) corpus (Bowman et al., 2015) provides approximately 550K hypothesis/premise pairs. Our baseline, the ESIM sequence model from Chen et al. (2017), uses a biLSTM to encode the premise and hypothesis, followed by a matrix attention layer, a local inference layer, another biLSTM inference composition layer, and finally a pooling operation before the output layer. Overall, adding ELMo to the ESIM model improves accuracy by an average of 0.7% across five random seeds. A five member ensemble pushes the overall accuracy to 89.3%, exceeding the previous ensemble best of 88.9% (Gong et al., 2018).

Semantic role labeling A semantic role labeling (SRL) system models the predicate-argument structure of a sentence, and is often described as answering "Who did what to whom". He et al. (2017) modeled SRL as a BIO tagging problem and used an 8-layer deep biLSTM with forward and backward directions interleaved, following Zhou and Xu (2015). As shown in Table 1, when adding ELMo to a re-implementation of He et al. (2017) the single model test set F_1 jumped 3.2% from 81.4% to 84.6% – a new state-of-the-art on the OntoNotes benchmark (Pradhan et al., 2013), even improving over the previous best ensemble result by 1.2%.

Coreference resolution Coreference resolution is the task of clustering mentions in text that refer to the same underlying real world entities. Our baseline model is the end-to-end span-based neural model of Lee et al. (2017). It uses a biLSTM and attention mechanism to first compute span representations and then applies a softmax mention ranking model to find coreference chains. In our experiments with the OntoNotes coreference annotations from the CoNLL 2012 shared task (Pradhan et al., 2012), adding ELMo improved the average F_1 by 3.2% from 67.2 to 70.4, establishing a new state of the art, again improving over the previous best ensemble result by 1.6% F_1.

Named entity extraction The CoNLL 2003 NER task (Sang and Meulder, 2003) consists of newswire from the Reuters RCV1 corpus tagged with four different entity types (`PER`, `LOC`, `ORG`, `MISC`). Following recent state-of-the-art systems (Lample et al., 2016; Peters et al., 2017), the baseline model uses pre-trained word embeddings, a character-based CNN representation, two biLSTM layers and a conditional random field (CRF) loss (Lafferty et al., 2001), similar to Collobert et al. (2011). As shown in Table 1, our ELMo enhanced biLSTM-CRF achieves 92.22% F_1 averaged over five runs. The key difference between our system and the previous state of the art from Peters et al. (2017) is that we allowed the task model to learn a weighted average of all biLM layers, whereas Peters et al. (2017) only use the top biLM layer. As shown in Sec. 5.1, using all layers instead of just the last layer improves performance across multiple tasks.

Sentiment analysis The fine-grained sentiment classification task in the Stanford Sentiment Treebank (SST-5; Socher et al., 2013) involves selecting one of five labels (from very negative to very positive) to describe a sentence from a movie review. The sentences contain diverse linguistic phenomena such as idioms and complex syntac-

Task	Baseline	Last Only	All layers	
			λ=1	λ=0.001
SQuAD	80.8	84.7	85.0	**85.2**
SNLI	88.1	89.1	89.3	**89.5**
SRL	81.6	84.1	84.6	**84.8**

Table 2: Development set performance for SQuAD, SNLI and SRL comparing using all layers of the biLM (with different choices of regularization strength λ) to just the top layer.

Task	Input Only	Input & Output	Output Only
SQuAD	85.1	**85.6**	84.8
SNLI	88.9	**89.5**	88.7
SRL	**84.7**	84.3	80.9

Table 3: Development set performance for SQuAD, SNLI and SRL when including ELMo at different locations in the supervised model.

tic constructions such as negations that are difficult for models to learn. Our baseline model is the biattentive classification network (BCN) from McCann et al. (2017), which also held the prior state-of-the-art result when augmented with CoVe embeddings. Replacing CoVe with ELMo in the BCN model results in a 1.0% absolute accuracy improvement over the state of the art.

5 Analysis

This section provides an ablation analysis to validate our chief claims and to elucidate some interesting aspects of ELMo representations. Sec. 5.1 shows that using deep contextual representations in downstream tasks improves performance over previous work that uses just the top layer, regardless of whether they are produced from a biLM or MT encoder, and that ELMo representations provide the best overall performance. Sec. 5.3 explores the different types of contextual information captured in biLMs and uses two intrinsic evaluations to show that syntactic information is better represented at lower layers while semantic information is captured a higher layers, consistent with MT encoders. It also shows that our biLM consistently provides richer representations then CoVe. Additionally, we analyze the sensitivity to where ELMo is included in the task model (Sec. 5.2), training set size (Sec. 5.4), and visualize the ELMo learned weights across the tasks (Sec. 5.5).

5.1 Alternate layer weighting schemes

There are many alternatives to Equation 1 for combining the biLM layers. Previous work on contextual representations used only the last layer, whether it be from a biLM (Peters et al., 2017) or an MT encoder (CoVe; McCann et al., 2017). The choice of the regularization parameter λ is also important, as large values such as $\lambda = 1$ effectively reduce the weighting function to a simple average over the layers, while smaller values (e.g., $\lambda = 0.001$) allow the layer weights to vary.

Table 2 compares these alternatives for SQuAD, SNLI and SRL. Including representations from all layers improves overall performance over just using the last layer, and including contextual representations from the last layer improves performance over the baseline. For example, in the case of SQuAD, using just the last biLM layer improves development F_1 by 3.9% over the baseline. Averaging all biLM layers instead of using just the last layer improves F_1 another 0.3% (comparing "Last Only" to λ=1 columns), and allowing the task model to learn individual layer weights improves F_1 another 0.2% (λ=1 vs. λ=0.001). A small λ is preferred in most cases with ELMo, although for NER, a task with a smaller training set, the results are insensitive to λ (not shown).

The overall trend is similar with CoVe but with smaller increases over the baseline. For SNLI, averaging all layers with λ=1 improves development accuracy from 88.2 to 88.7% over using just the last layer. SRL F_1 increased a marginal 0.1% to 82.2 for the λ=1 case compared to using the last layer only.

5.2 Where to include ELMo?

All of the task architectures in this paper include word embeddings only as input to the lowest layer biRNN. However, we find that including ELMo at the output of the biRNN in task-specific architectures improves overall results for some tasks. As shown in Table 3, including ELMo at both the input and output layers for SNLI and SQuAD improves over just the input layer, but for SRL (and coreference resolution, not shown) performance is highest when it is included at just the input layer. One possible explanation for this result is that both the SNLI and SQuAD architectures use attention layers after the biRNN, so introducing ELMo at this layer allows the model to attend directly to the biLM's internal representations. In the SRL case,

	Source	Nearest Neighbors
GloVe	play	playing, game, games, played, players, plays, player, Play, football, multiplayer
biLM	Chico Ruiz made a spectacular play on Alusik 's grounder {...}	Kieffer , the only junior in the group , was commended for his ability to hit in the clutch , as well as his all-round excellent play .
	Olivia De Havilland signed to do a Broadway play for Garson {...}	{...} they were actors who had been handed fat roles in a successful play , and had talent enough to fill the roles competently , with nice understatement .

Table 4: Nearest neighbors to "play" using GloVe and the context embeddings from a biLM.

Model	**F_1**
WordNet 1st Sense Baseline	65.9
Raganato et al. (2017a)	69.9
Iacobacci et al. (2016)	**70.1**
CoVe, First Layer	59.4
CoVe, Second Layer	*64.7*
biLM, First layer	67.4
biLM, Second layer	*69.0*

Table 5: All-words fine grained WSD F_1. For CoVe and the biLM, we report scores for both the first and second layer biLSTMs.

Model	**Acc.**
Collobert et al. (2011)	97.3
Ma and Hovy (2016)	97.6
Ling et al. (2015)	**97.8**
CoVe, First Layer	*93.3*
CoVe, Second Layer	92.8
biLM, First Layer	*97.3*
biLM, Second Layer	96.8

Table 6: Test set POS tagging accuracies for PTB. For CoVe and the biLM, we report scores for both the first and second layer biLSTMs.

the task-specific context representations are likely more important than those from the biLM.

5.3 What information is captured by the biLM's representations?

Since adding ELMo improves task performance over word vectors alone, the biLM's contextual representations must encode information generally useful for NLP tasks that is not captured in word vectors. Intuitively, the biLM must be disambiguating the meaning of words using their context. Consider "play", a highly polysemous word. The top of Table 4 lists nearest neighbors to "play" using GloVe vectors. They are spread across several parts of speech (e.g., "played", "playing" as verbs, and "player", "game" as nouns) but concentrated in the sports-related senses of "play". In contrast, the bottom two rows show nearest neighbor sentences from the SemCor dataset (see below) using the biLM's context representation of "play" in the source sentence. In these cases, the biLM is able to disambiguate both the part of speech and word sense in the source sentence.

These observations can be quantified using an intrinsic evaluation of the contextual representations similar to Belinkov et al. (2017). To isolate the information encoded by the biLM, the representations are used to directly make predictions for a fine grained word sense disambiguation (WSD) task and a POS tagging task. Using this approach, it is also possible to compare to CoVe, and across each of the individual layers.

Word sense disambiguation Given a sentence, we can use the biLM representations to predict the sense of a target word using a simple 1-nearest neighbor approach, similar to Melamud et al. (2016). To do so, we first use the biLM to compute representations for all words in SemCor 3.0, our training corpus (Miller et al., 1994), and then take the average representation for each sense. At test time, we again use the biLM to compute representations for a given target word and take the nearest neighbor sense from the training set, falling back to the first sense from WordNet for lemmas not observed during training.

Table 5 compares WSD results using the evaluation framework from Raganato et al. (2017b) across the same suite of four test sets in Raganato et al. (2017a). Overall, the biLM top layer rep-

resentations have F_1 of 69.0 and are better at WSD then the first layer. This is competitive with a state-of-the-art WSD-specific supervised model using hand crafted features (Iacobacci et al., 2016) and a task specific biLSTM that is also trained with auxiliary coarse-grained semantic labels and POS tags (Raganato et al., 2017a). The CoVe biLSTM layers follow a similar pattern to those from the biLM (higher overall performance at the second layer compared to the first); however, our biLM outperforms the CoVe biLSTM, which trails the WordNet first sense baseline.

POS tagging To examine whether the biLM captures basic syntax, we used the context representations as input to a linear classifier that predicts POS tags with the Wall Street Journal portion of the Penn Treebank (PTB) (Marcus et al., 1993). As the linear classifier adds only a small amount of model capacity, this is direct test of the biLM's representations. Similar to WSD, the biLM representations are competitive with carefully tuned, task specific biLSTMs (Ling et al., 2015; Ma and Hovy, 2016). However, unlike WSD, accuracies using the first biLM layer are higher than the top layer, consistent with results from deep biLSTMs in multi-task training (Søgaard and Goldberg, 2016; Hashimoto et al., 2017) and MT (Belinkov et al., 2017). CoVe POS tagging accuracies follow the same pattern as those from the biLM, and just like for WSD, the biLM achieves higher accuracies than the CoVe encoder.

Implications for supervised tasks Taken together, these experiments confirm different layers in the biLM represent different types of information and explain why including all biLM layers is important for the highest performance in downstream tasks. In addition, the biLM's representations are more transferable to WSD and POS tagging than those in CoVe, helping to illustrate why ELMo outperforms CoVe in downstream tasks.

5.4 Sample efficiency

Adding ELMo to a model increases the sample efficiency considerably, both in terms of number of parameter updates to reach state-of-the-art performance and the overall training set size. For example, the SRL model reaches a maximum development F_1 after 486 epochs of training without ELMo. After adding ELMo, the model exceeds the baseline maximum at epoch 10, a 98% relative decrease in the number of updates needed to reach the same level of performance.

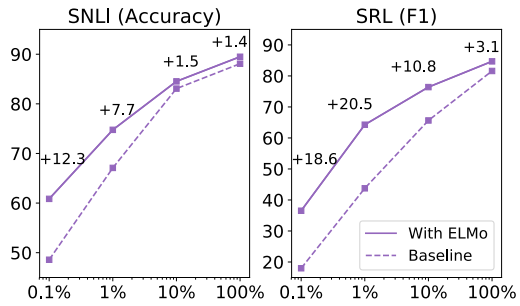


Figure 1: Comparison of baseline vs. ELMo performance for SNLI and SRL as the training set size is varied from 0.1% to 100%.

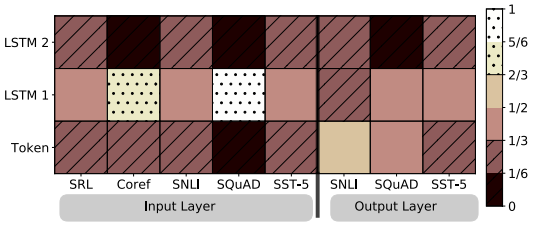


Figure 2: Visualization of softmax normalized biLM layer weights across tasks and ELMo locations. Normalized weights less then $1/3$ are hatched with horizontal lines and those greater then $2/3$ are speckled.

In addition, ELMo-enhanced models use smaller training sets more efficiently than models without ELMo. Figure 1 compares the performance of baselines models with and without ELMo as the percentage of the full training set is varied from 0.1% to 100%. Improvements with ELMo are largest for smaller training sets and significantly reduce the amount of training data needed to reach a given level of performance. In the SRL case, the ELMo model with 1% of the training set has about the same F_1 as the baseline model with 10% of the training set.

5.5 Visualization of learned weights

Figure 2 visualizes the softmax-normalized learned layer weights. At the input layer, the task model favors the first biLSTM layer. For coreference and SQuAD, the this is strongly favored, but the distribution is less peaked for the other tasks. The output layer weights are relatively balanced, with a slight preference for the lower layers.

Task	GloVe	ELMo type	ELMo	ELMo + GloVe
SQuAD	80.8	81.4	85.3	85.6
SNLI	88.1	88.5	89.1	89.5
SRL	81.6	81.7	84.5	84.7

Table 7: Development set ablation analysis for SQuAD, SNLI and SRL comparing different choices for the context-independent type representation and contextual representation. From left to right, the table compares systems with only GloVe vectors; only the ELMo context-independent type representation without the ELMo biLSTM layers; full ELMo representations without GloVe; both GloVe and ELMo.

5.6 Contextual vs. sub-word information

In addition to the contextual information captured in the biLM's biLSTM layers, ELMo representations also contain sub-word information in the fully character based context insensitive type layer, x_k^{LM}. To analyze the relative contribution of the contextual information compared to the sub-word information, we ran an additional ablation that replaced the GloVe vectors with just the biLM character based x_k^{LM} layer without the biLM biLSTM layers. Table 7 summarizes the results for SQuAD, SNLI and SNLI. Replacing the GloVe vectors with the biLM character layer gives a slight improvement for all tasks (e.g. from 80.8 to 81.4 F_1 for SQuAD), but overall the improvements are small compared to the full ELMo model. From this, we conclude that most of the gains in the downstream tasks are due to the contextual information and not the sub-word information.

5.7 Are pre-trained vectors necessary with ELMo?

All of the results presented in Sec.4 include pre-trained word vectors in addition to ELMo representations. However, it is natural to ask whether pre-trained vectors are still necessary with high quality contextualized representations. As shown in the two right hand columns of Table 7, adding GloVe to models with ELMo generally provides a marginal improvement over ELMo only models (e.g. 0.2% F_1 improvement for SRL from 84.5 to 84.7).

6 Conclusion

We have introduced a general approach for learning high-quality deep context-dependent representations from biLMs, and shown large improvements when applying ELMo to a broad range of NLP tasks. Through ablations and other controlled experiments, we have also confirmed that the biLM layers efficiently encode different types of syntactic and semantic information about words-in-context, and that using all layers improves overall task performance.

References

Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR* abs/1607.06450.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James R. Glass. 2017. What do neural machine translation models learn about morphology? In *ACL*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL* 5:135–146.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *INTERSPEECH*.

Qian Chen, Xiao-Dan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced lstm for natural language inference. In *ACL*.

Jason Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. In *TACL*.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *SSST@EMNLP*.

Christopher Clark and Matthew Gardner. 2017. Simple and effective multi-paragraph reading comprehension. *CoRR* abs/1710.10723.

Kevin Clark and Christopher D. Manning. 2016. Deep reinforcement learning for mention-ranking coreference models. In *EMNLP*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. In *JMLR*.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. In *NIPS*.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *EMNLP*.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *NIPS*.

Yichen Gong, Heng Luo, and Jian Zhang. 2018. Natural language inference over interaction space. In *ICLR*.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *EMNLP 2017*.

Luheng He, Kenton Lee, Mike Lewis, and Luke S. Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *ACL*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Embeddings for word sense disambiguation: An evaluation study. In *ACL*.

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *CoRR* abs/1602.02410.

Rafal Józefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *ICML*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2015. Character-aware neural language models. In *AAAI 2016*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, Ishaan Gulrajani James Bradbury, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *ICML*.

John D. Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL-HLT*.

Kenton Lee, Luheng He, Mike Lewis, and Luke S. Zettlemoyer. 2017. End-to-end neural coreference resolution. In *EMNLP*.

Wang Ling, Chris Dyer, Alan W. Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luís Marujo, and Tiago Luís. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *EMNLP*.

Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. 2017. Stochastic answer networks for machine reading comprehension. *arXiv preprint arXiv:1712.03556* .

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *ACL*.

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

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *NIPS 2017*.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional lstm. In *CoNLL*.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2017. On the state of the art of evaluation in neural language models. *CoRR* abs/1707.05589.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and optimizing lstm language models. *CoRR* abs/1708.02182.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

George A. Miller, Martin Chodorow, Shari Landes, Claudia Leacock, and Robert G. Thomas. 1994. Using a semantic concordance for sense identification. In *HLT*.

Tsendsuren Munkhdalai and Hong Yu. 2017. Neural tree indexers for text understanding. In *EACL*.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *EMNLP*.

Martha Palmer, Paul Kingsbury, and Daniel Gildea. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics* 31:71–106.

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

Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *ACL*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using ontonotes. In *CoNLL*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *EMNLP-CoNLL Shared Task*.

Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2017a. Neural sequence learning models for word sense disambiguation. In *EMNLP*.

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017b. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *EACL*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*.

Prajit Ramachandran, Peter Liu, and Quoc Le. 2017. Improving sequence to sequence learning with unlabeled data. In *EMNLP*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *CoNLL*.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *ICLR*.

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

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *ACL 2016*.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15:1929–1958.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *NIPS*.

Joseph P. Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *ACL*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *ACL*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. In *EMNLP*.

Sam Wiseman, Alexander M. Rush, and Stuart M. Shieber. 2016. Learning global features for coreference resolution. In *HLT-NAACL*.

Matthew D. Zeiler. 2012. Adadelta: An adaptive learning rate method. *CoRR* abs/1212.5701.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *ACL*.

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. In *COLING*.

Learning to Map Context-Dependent Sentences to Executable Formal Queries

Alane Suhr[†], **Srinivasan Iyer**[‡], **and Yoav Artzi**[†]

[†] Dept. of Computer Science and Cornell Tech, Cornell University, New York, NY
{suhr, yoav}@cs.cornell.edu
[‡] Paul G. Allen School of Computer Science & Engineering, Univ. of Washington, Seattle, WA
sviyer@cs.washington.edu

Abstract

We propose a context-dependent model to map utterances within an interaction to executable formal queries. To incorporate interaction history, the model maintains an interaction-level encoder that updates after each turn, and can copy sub-sequences of previously predicted queries during generation. Our approach combines implicit and explicit modeling of references between utterances. We evaluate our model on the ATIS flight planning interactions, and demonstrate the benefits of modeling context and explicit references.

1 Introduction

The meaning of conversational utterances depends strongly on the history of the interaction. Consider a user querying a flight database using natural language (Figure 1). Given a user utterance, the system must generate a query, execute it, and display results to the user, who then provides the next request. Key to correctly mapping utterances to executable queries is resolving references. For example, the second utterance implicitly depends on the first, and the reference *ones* in the third utterance explicitly refers to the response to the second utterance. Within an interactive system, this information needs to be composed with mentions of database entries (e.g., *Seattle*, *next Monday*) to generate a formal executable representation. In this paper, we propose encoder-decoder models that directly map user utterances to executable queries, while considering the history of the interaction, including both previous utterances and their generated queries.

Reasoning about how the meaning of an utterance depends on the history of the interaction is critical to correctly respond to user requests. As interactions progress, users may omit previously-mentioned constraints and entities, and an increasing portion of the utterance meaning must be derived from the interaction history. Figure 2 shows SQL queries for the utterances in Figure 1. As the interaction progresses, the majority of the generated query is derived from the interaction history (underlined), rather than from the current utterance. A key challenge is resolving what past information is incorporated and how. For example, in the figure, the second utterance depends on the set of flights defined by the first, while adding a new constraint. The third utterance further refines this set by adding a constraint to the constraints from both previous utterances. In contrast, the fourth utterance refers only to the first one, and skips the two utterances in between.[1] Correctly generating the fourth query requires understanding that the time constraint (*at 7pm*) can be ignored as it follows an airline constraint that has been replaced.

show me flights from seattle to boston next monday
[Table with 31 flights]
on american airlines
[Table with 5 flights]
which ones arrive at 7pm
[No flights returned]
show me delta flights
[Table with 5 flights]
...

Figure 1: An excerpt of an interaction from the ATIS flight planning system (Hemphill et al., 1990; Dahl et al., 1994). Each request is followed by a description of the system response.

We study complementary methods to enable this type of reasoning. The first set of methods implicitly reason about references by modifying the encoder-decoder architecture to encode information from previous utterances for generation decisions. We experiment with attending over previous utterances and using an interaction-level recurrent encoder. We also study explicitly maintaining a set of referents using segments from pre-

[1] An alternative explanation is that utterance four refers to utterance three, and deletes the time and airline constraints.

Proceedings of NAACL-HLT 2018, pages 2238–2249
New Orleans, Louisiana, June 1 - 6, 2018.

$\bar{x}_1$: *show me flights from seattle to boston next monday*

$\bar{y}_1$:
```
(SELECT DISTINCT flight.flight_id FROM flight WHERE (flight.from_airport IN (SELECT
airport_service.airport_code FROM airport_service WHERE airport_service.city_code IN (SELECT
city.city_code FROM city WHERE city.city_name = 'SEATTLE'))) AND (flight.to_airport IN (SELECT
airport_service.airport_code FROM airport_service WHERE airport_service.city_code IN (SELECT
city.city_code FROM city WHERE city.city_name = 'BOSTON'))) AND (flight.flight_days IN (SELECT
days.days_code FROM days WHERE days.day_name IN (SELECT date_day.day_name FROM date_day WHERE
date_day.year = 1993 AND date_day.month_number = 2 AND date_day.day_number = 8))));
```

$\bar{x}_2$: *on american airlines*

$\bar{y}_2$:
```
(SELECT DISTINCT flight.flight_id FROM flight WHERE (flight.airline_code = 'AA') AND (flight.from_airport
IN (SELECT airport_service.airport_code FROM airport_service WHERE airport_service.city_code IN (SELECT
city.city_code FROM city WHERE city.city_name = 'SEATTLE'))) AND (flight.to_airport IN (SELECT air
port_service.airport_code FROM airport_service WHERE airport_service.city_code IN (SELECT city.city_code
FROM city WHERE city.city_name = 'BOSTON'))) AND (flight.flight_days IN (SELECT days.days_code FROM
days WHERE days.day_name IN (SELECT date_day.day_name FROM date_day WHERE date_day.year = 1993 AND
date_day.month_number = 2 AND date_day.day_number = 8))));
```

$\bar{x}_3$: *which ones arrive at 7pm*

$\bar{y}_3$:
```
(SELECT DISTINCT flight.flight_id FROM flight WHERE (flight.airline_code = 'AA') AND (flight.from_airport
IN (SELECT airport_service.airport_code FROM airport_service WHERE airport_service.city_code IN (SELECT
city.city_code FROM city WHERE city.city_name = 'SEATTLE'))) AND (flight.to_airport IN (SELECT air
port_service.airport_code FROM airport_service WHERE airport_service.city_code IN (SELECT city.city_code
FROM city WHERE city.city_name = 'BOSTON'))) AND (flight.flight_days IN (SELECT days.days_code FROM
days WHERE days.day_name IN (SELECT date_day.day_name FROM date_day WHERE date_day.year = 1993 AND
date_day.month_number = 2 AND date_day.day_number = 8))) AND (flight.arrival_time = 1900));
```

$\bar{x}_4$: *show me delta flights*

$\bar{y}_4$:
```
(SELECT DISTINCT flight.flight_id FROM flight WHERE (flight.airline_code = 'DL') AND (flight.from_airport
IN (SELECT airport_service.airport_code FROM airport_service WHERE airport_service.city_code IN (SELECT
city.city_code FROM city WHERE city.city_name = 'SEATTLE'))) AND (flight.to_airport IN (SELECT air
port_service.airport_code FROM airport_service WHERE airport_service.city_code IN (SELECT city.city_code
FROM city WHERE city.city_name = 'BOSTON'))) AND (flight.flight_days IN (SELECT days.days_code FROM
days WHERE days.day_name IN (SELECT date_day.day_name FROM date_day WHERE date_day.year = 1993 AND
date_day.month_number = 2 AND date_day.day_number = 8))));
```

Figure 2: Annotated SQL queries ($\bar{y}_1$,...,$\bar{y}_4$) in the ATIS (Hemphill et al., 1990) domain for utterances ($\bar{x}_1$,...,$\bar{x}_4$) from Figure 1. Underlining (not part of the annotation) indicates segments originating from the interaction context.

vious queries. At each step, the decoder chooses whether to output a token or select a segment from the set, which is appended to the output in a single decoding step. In addition to enabling references to previously mentioned entities, sets, and constraints, this method also reduces the number of generation steps required, illustrated by the underlined segments in Figure 2. For example, the query $\bar{y}_2$ will require 17 steps instead of 94.

We evaluate our approach using the ATIS (Hemphill et al., 1990; Dahl et al., 1994) task, where a user interacts with a SQL flight database using natural language requests, and almost all queries require joins across multiple tables. In addition to reasoning about contextual phenomena, we design our system to effectively resolve database values, including resolution of time expressions (e.g., *next monday* in Figure 1) using an existing semantic parser. Our evaluation shows that reasoning about the history of the interaction is necessary, relatively increasing performance by 28.6% over a baseline with no access to this information, and that combining the implicit and explicit methods provides the best performance. Furthermore, our analysis shows that our full approach maintains its performance as interaction length increases, while the performance of systems without explicit modeling deteriorates. Our code is available at https://github.com/clic-lab/atis.

2 Technical Overview

Our goal is to map utterances in interactions to formal executable queries. We evaluate our approach with the ATIS corpus (Hemphill et al., 1990; Dahl et al., 1994), where users query a realistic flight planning system using natural language. The system responds by displaying tables and database entries. User utterances are mapped to SQL to query a complex database with 27 tables and 162K entries. 96.6% of the queries require joins of different tables. Section 7 describes ATIS.

Task Notation Let $\mathcal{I}$ be the set of all interactions, $\mathcal{X}$ the set of all utterances, and $\mathcal{Y}$ the set of all formal queries. A user utterance $\bar{x} \in \mathcal{X}$ of length $|\bar{x}|$ is a sequence $\langle x_1, \ldots, x_{|\bar{x}|} \rangle$, where each x_i is a natural language token. A formal query $\bar{y} \in \mathcal{Y}$ of length $|\bar{y}|$ is a sequence $\langle y_1, \ldots, y_{|\bar{y}|} \rangle$, where each y_i is a formal query token. An interaction $\bar{I} \in \mathcal{I}$ is a sequence of n utterance-query pairs $\langle (\bar{x}_1, \bar{y}_1), \ldots, (\bar{x}_n, \bar{y}_n) \rangle$ representing an interaction with n turns. To refer to indexed interactions and their content, we mark $\bar{I}^{(l)}$ as an interaction with index l, the i-th utterance and query in $\bar{I}^{(l)}$ as $\bar{x}_i^{(l)}$ and $\bar{y}_i^{(l)}$, and the j-th tokens in $\bar{x}_i^{(l)}$ and $\bar{y}_i^{(l)}$ as $x_{i,j}^{(l)}$ and $y_{i,j}^{(l)}$. At turn i, we denote the interaction history of length $i-1$ as $\bar{I}[:i-1] = \langle (\bar{x}_1, \bar{y}_1), \ldots, (\bar{x}_{i-1}, \bar{y}_{i-1}) \rangle$. Given $\bar{I}[:i-1]$ and utterance $\bar{x}_i$ our goal is to generate $\bar{y}_i$, while considering both $\bar{x}_i$ and $\bar{I}[:i-1]$. Following the ex-

ecution of $\bar{y}_i$, the interaction history at turn $i+1$ becomes $\bar{I}[:i] = \langle (\bar{x}_1, \bar{y}_1), \ldots, (\bar{x}_i, \bar{y}_i) \rangle$.

Model Our model is based on the recurrent neural network (RNN; Elman, 1990) encoder-decoder framework with attention (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015; Luong et al., 2015). We modify the model in three ways to reason about context from the interaction history by attending over previous utterances (Section 4.2), adding a turn-level recurrent encoder that updates after each turn (Section 4.3), and adding a mechanism to copy segments of queries from previous utterances (Section 4.4). We also design a scoring function to score values that are abstracted during pre-processing, including entities and times (Section 6). The full model selects between generating query tokens and copying complete segments from previous queries.

Learning We assume access to a training set that contains N interactions $\{\bar{I}^{(l)}\}_{l=1}^{N}$. We train using a token-level cross-entropy objective (Section 5). For models that use the turn-level encoder, we construct computational graphs for the entire interaction and back-propagate the loss for all queries together. Without the turn-level encoder, each utterance is processed separately.

Evaluation We evaluate using a test set $\{\bar{I}^{(l)}\}_{l=1}^{M}$ of M interactions. We measure the accuracy of each utterance for each test interaction against the annotated query and its execution result. For models that copy segments from previous queries, we evaluate using both predicted and gold previous queries.

3 Related Work

Mapping sentences to formal representations, commonly known as semantic parsing, has been studied extensively with linguistically-motivated compositional representations, including variable-free logic (e.g., Zelle and Mooney, 1996; Clarke et al., 2010), lambda calculus (e.g., Zettlemoyer and Collins, 2005; Artzi and Zettlemoyer, 2011; Kushman and Barzilay, 2013), and dependency-based compositional semantics (e.g., Liang et al., 2011; Berant et al., 2013). Recovering lambda-calculus representations was also studied with ATIS with focus on context-independent meaning using grammar-based approaches (Zettlemoyer and Collins, 2007; Kwiatkowski et al., 2011; Wang et al., 2014) and neural networks (Dong and Lapata, 2016; Jia and Liang, 2016).

Recovering context-independent executable representations has been receiving increasing attention. Mapping sentence in isolation to SQL queries has been studied with ATIS using statistical parsing (Popescu et al., 2004; Poon, 2013) and sequence-to-sequence models (Iyer et al., 2017). Generating executable programs was studied with other domains and formal languages (Giordani and Moschitti, 2012; Ling et al., 2016; Zhong et al., 2017; Xu et al., 2017). Recently, various approaches were proposed to use the formal language syntax to constrain the search space (Yin and Neubig, 2017; Rabinovich et al., 2017; Krishnamurthy et al., 2017; Cheng et al., 2017) making all outputs valid programs. These contributions are orthogonal to ours, and can be directly integrated into our decoder.

Generating context-dependent formal representations has received less attention. Miller et al. (1996) used ATIS and mapped utterances to semantic frames, which were then mapped to SQL queries. For learning, they required full supervision, including annotated parse trees and contextual dependencies.[2] Zettlemoyer and Collins (2009) addressed the problem with lambda calculus, using a semantic parser trained separately with context-independent data. In contrast, we generate executable formal queries and require only interaction query annotations for training.

Recovering context-dependent meaning was also studied with the SCONE (Long et al., 2016) and SequentialQA (Iyyer et al., 2017) corpora. We compare ATIS to these corpora in Section 7. Resolving explicit references, a part of our problem, has been studied as co-reference resolution (Ng, 2010). Context-dependent language understanding was also studied for dialogue systems, including with ATIS, as surveyed by Tür et al. (2010). More recently, encoder-decoder methods were applied to dialogue systems (Peng et al., 2017; Li et al., 2017), including using hierarchical RNNs (Serban et al., 2016, 2017), an architecture related to our turn-level encoder. These approaches use slot-filling frames with limited expressivity, while we focus on the original representation of unconstrained SQL queries.

[2] Miller et al. (1996) provide limited details about their evaluation. Later work notes that they evaluate SQL query correctness (Zettlemoyer and Collins, 2009) with an accuracy of 78.4%, higher than our results. However, the lack of details (e.g., if the metric is strict or relaxed) makes comparison difficult. In addition, we use significantly less supervision, and re-split the data to avoid scenario bias (Section 7).

4 Context-dependent Model

We base our model on an encoder-decoder architecture with attention (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015; Luong et al., 2015). At each interaction turn i, given the current utterance $\bar{x}_i$ and the interaction history $\bar{I}[:i-1]$, the model generates the formal query $\bar{y}_i$. Figure 3 illustrates our architecture. We describe the base architecture, and gradually add components.

4.1 Base Encoder-Decoder Architecture

Our base architecture uses an encoder to process the user utterance $\bar{x}_i = \langle x_{i,1}, \ldots, x_{i,|\bar{x}_i|} \rangle$ and a decoder to generate the output query $\bar{y}_i$ token-by-token. This architecture does not observe the interaction history $\bar{I}[:i-1]$.

The encoder computes a hidden state $\mathbf{h}_j^E = [\mathbf{h}_j^{\overrightarrow{E}}; \mathbf{h}_j^{\overleftarrow{E}}]$ for each token $x_{i,j}$ using a bi-directional RNN. The forward RNN is defined by: [3]

$$\mathbf{h}_j^{\overrightarrow{E}} = \text{LSTM}^{\overrightarrow{E}}\left(\phi^x(x_{i,j}); \mathbf{h}_{j-1}^{\overrightarrow{E}}\right) \quad , \tag{1}$$

where $\text{LSTM}^{\overrightarrow{E}}$ is a long short-term memory recurrence (LSTM; Hochreiter and Schmidhuber, 1997) and ϕ^x is a learned embedding function for input tokens. The backward RNN recurs in the opposite direction with separate parameters.

We generate the query with an RNN decoder. The decoder state at step k is:

$$\mathbf{h}_k^D = \text{LSTM}^D\left([\phi^y(y_{i,k-1}); \mathbf{c}_{k-1}]; \mathbf{h}_{k-1}^D\right) \quad ,$$

where LSTM^D is a two-layer LSTM recurrence, ϕ^y is a learned embedding function for query tokens, and $\mathbf{c}_k$ is an attention vector computed from the encoder states. $y_{i,0}$ is a special start token, and $\mathbf{c}_0$ is a zero-vector. The initial hidden state and cell memory of each layer are initialized as $\mathbf{h}_{|\bar{x}_i|}^E$ and $\mathbf{c}_{|\bar{x}_i|}^E$. The attention vector $\mathbf{c}_k$ is a weighted sum of the encoder hidden states:

$$s_k(j) = \mathbf{h}_j^E \mathbf{W}^{\mathbf{A}} \mathbf{h}_k^D \tag{2}$$

$$\alpha_k = \text{softmax}(\mathbf{s}_k) \tag{3}$$

$$\mathbf{c}_k = \sum_{j=1}^{|\bar{x}_i|} \mathbf{h}_j^E \alpha_k(j) \quad , \tag{4}$$

where $\mathbf{W}^{\mathbf{A}}$ is a learned matrix. The probabilities of output query tokens are computed as:

$$\mathbf{m}_k = \tanh\left([\mathbf{h}_k^D; \mathbf{c}_k]\mathbf{W}^m\right) \tag{5}$$

$$P(y_{i,k} = w \mid \bar{x}_i, \bar{y}_{i,1:k-1}) \propto \exp(\mathbf{m}_k \mathbf{W}_w^o + \mathbf{b}_w^o) \tag{6}$$

where $\mathbf{W}^m$, $\mathbf{W}^o$, and $\mathbf{b}^o$ are learned.

[3] We omit the memory cell (often denoted as $\mathbf{c}_j$) from all LSTM descriptions. We use only the LSTM hidden state $\mathbf{h}_j$ in other parts of the architecture unless explicitly noted.

4.2 Incorporating Recent History

We provide the model with the most recent interaction history by concatenating the previous h utterances $\langle \bar{x}_{i-h}, ..., \bar{x}_{i-1} \rangle$ with the current utterance in order, adding a special delimiter token between each utterance. The concatenated input provides the model access to previous utterances, but not to previously generated queries, or utterances that are more than h turns in the past. The architecture remains the same, except that the encoder and attention are computed over the concatenated sequence of tokens. The probability of an output query token is computed the same, but is now conditioned on the interaction history:

$$P(y_{i,k} = w \mid \bar{x}_i, \bar{y}_{i,1:k-1}, \bar{I}[:i-1]) \propto \exp(\mathbf{m}_k \mathbf{W}_w^o + \mathbf{b}_w^o) \quad . \tag{7}$$

4.3 Turn-level Encoder

Concatenating recent utterances to provide access to recent history has computational drawbacks. The encoding of the utterance depends on its location in the concatenated string. This requires encoding all recent history for each new utterance, and does not allow re-use of computation between utterances during encoding. It also introduces a tradeoff between computation cost and expressivity: attending over the h previous utterances allows the decoder access to the information in these utterances when generating a query, but is computationally more expensive as h increases. We address this by encoding each utterance once. To account for the influence of the interaction history on utterance encoding, we maintain a discourse state encoding $\mathbf{h}_i^I$ computed with a turn-level recurrence, and use it during utterance encoding. The state is maintained and updated over the entire interaction. At turn i, this model has access to the complete prefix of the interaction $\bar{I}[:i-1]$ and the current request $\bar{x}_i$. In contrast, the concatenation-based encoder (Section 4.2) has access only to information from the previous h utterances. We also use positional encoding in the attention computation to account for the position of each utterance relative to the current utterance.

Formally, we modify Equation 1 to encode $\bar{x}_i$:

$$\mathbf{h}_{i,j}^{\overrightarrow{E}} = \text{LSTM}^{\overrightarrow{E}}\left(\left[\phi^x(x_{i,j}); \mathbf{h}_{i-1}^I\right]; \mathbf{h}_{i,j-1}^{\overrightarrow{E}}\right) \quad ,$$

where $\mathbf{h}_{i-1}^I$ is the discourse state following utterance $\bar{x}_{i-1}$. $\text{LSTM}^{\overleftarrow{E}}$ is modified analogously. In contrast to the concatenation-based model, the recurrence processes a single utterance. The dis-

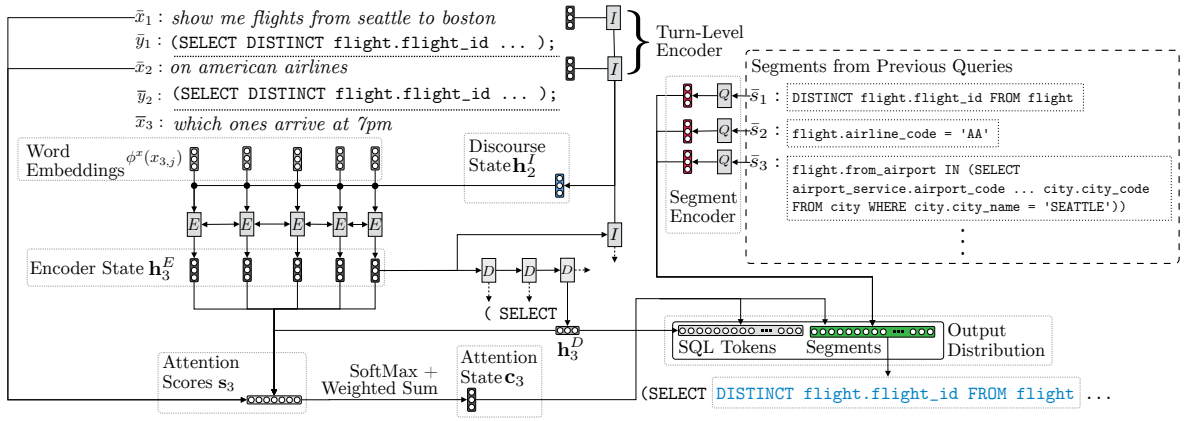


Figure 3: Illustration of the model architecture during the third decoding step while processing the instruction *which ones arrive at 7pm* from the interaction in Figure 2. The current discourse state $\mathbf{h}_2^I$ is used to encode the current utterance $\bar{x}_3$ (Section 4.3). Query segments from previous queries are encoded into vector representations (Section 4.4). In each generation step, the decoder attends over the previous and current utterances, and a probability distribution is computed over SQL tokens and query segments. Here, segment $\bar{s}_1$ is selected.

course state $\mathbf{h}_i^I$ is computed as

$$\mathbf{h}_i^I = \text{LSTM}^I\left(\mathbf{h}_{i,|\bar{x}_i|}^E; \mathbf{h}_{i-1}^I\right) \quad .$$

Similar to the concatenation-based model, we attend over the current utterance and the h previous utterances. We add relative position embeddings ϕ^I to each hidden state. These embeddings are learned for each possible distance $0, \ldots, h-1$ from the current utterance. We modify Equation 2 to index over both utterances and tokens:

$$s_k(t,j) = \left[\mathbf{h}_{t,j}^E; \phi^I(i-t)\right] \mathbf{W}^{\mathbf{A}} \mathbf{h}_k^D \quad . \tag{8}$$

In contrast to the concatenation model, without position embeddings, the attention computation has no indication of the utterance position, as our ablation shows in Section 8. The attention distribution is computed as in Equation 3, and normalized across all utterances. The position embedding is also used to compute the context vector $\mathbf{c}_k$:

$$\mathbf{c}_k = \sum_{t=i-h}^{i} \sum_{j=1}^{|\bar{x}_t|} \left[\mathbf{h}_{t,j}^E; \phi^I(i-t)\right] \alpha_k(t,j) \quad .$$

4.4 Copying Query Segments

The discourse state and attention over previous utterances allow the model to consider the interaction history when generating queries. However, we observe that context-dependent reasoning often requires generating sequences that were generated in previous turns. Figure 2 shows how segments (underlined) extracted from previous utterances are predominant in later queries. To take advantage of what was previously generated, we add copying of complete segments from previous queries by expanding the set of outputs at each generation step. This mechanism explicitly models references, reduces the number of steps required to generate queries, and provides an interpretable view of what parts of a query originate in context. Figure 3 illustrates this architecture.

Extracting Segments Given the interaction history $\bar{I}[:\ i-1]$, we construct the set of segments S_{i-1} by deterministically extracting subtrees from previously generated queries.[4] In our data, we extract 13 ± 5.9 ($\mu \pm \sigma$) segments for each annotated query. Each segment $\bar{s} \in S_{i-1}$ is a tuple $\langle a, b, l, r \rangle$, where a and b are the indices of the first and most recent queries, $\bar{y}_a$ and $\bar{y}_b$, in the interaction that contain the segment. l and r are the start and end indices of the segment in $\bar{y}_b$.

Encoding Segments We represent a segment $\bar{s} = \langle a, b, l, r \rangle$ using the hidden states of an RNN encoding of the query $\bar{y}_b$. The hidden states $\langle \mathbf{h}^Q{}_1, ..., \mathbf{h}^Q{}_{|\bar{y}_b|} \rangle$ are computed using a bi-directional LSTM RNN similar to the utterance encoder (Equation 1), except using separate LSTM parameters and ϕ^y to embed the query tokens. The embedded representation of a segment is a concatenation of the hidden states at the segment endpoints and an embedding of the relative position of the utterance where it appears first:

$$\mathbf{h}^S = \left[\mathbf{h}_l^Q; \mathbf{h}_r^Q; \phi^g(\min(g, i-a))\right] \quad ,$$

where ϕ^g is a learned embedding function of the position of the initial query $\bar{y}_a$ relative to the current turn index i. We learn an embedding for each relative position that is smaller than g, and use the same embedding for all other positions.

Generation with Segments At each generation step, the decoder selects between a single query token or a segment. When a segment is selected, it

[4] The process of extracting sub-trees is described in the supplementary material.

is appended to the generated query, an embedded segment representation for the next step is computed, and generation continues. The probability of a segment $\bar{s} = \langle a, b, l, r \rangle$ at decoding step k is:

$$P(y_{i,k} = \bar{s} \mid \bar{x}_i, \bar{y}_{i,1:k-1}, \bar{I}[:i-1]) \propto \exp\left(\mathbf{m}_k \mathbf{W}^S \mathbf{h}^S\right) \quad (9)$$

where $\mathbf{m}_k$ is computed in Equation 5 and $\mathbf{W}^S$ is a learned matrix. To simplify the notation, we assign the segment to a single output token. The output probabilities (Equations 7 and 9) are normalized together to a single probability distribution. When a segment is selected, the embedding used as input for the next generation step is a bag-of-words encoding of the segment. We extend the output token function ϕ^y to take segments:

$$\phi^y(\bar{s} = \langle a, b, l, r \rangle) = \frac{1}{r-l} \sum_{k=l}^{r} \phi^y(y_{b,k}) \quad .$$

The recursion in ϕ^y is limited to depth one because segments do not contain other segments.

4.5 Inference with Full Model

Given an utterance $\bar{x}_i$ and the history of interaction $\bar{I}[:i-1]$, we generate the query $\bar{y}_i$. An interaction starts with the user providing the first utterance $\bar{x}_1$. The utterance is encoded using the initial discourse state $\mathbf{h}_0^I$, the discourse state $\mathbf{h}_1^I$ is computed, the query $\bar{y}_1$ is generated, and the set of segments S_1 is created. The initial discourse state $\mathbf{h}_0^I$ is learned, and the set of segments S_0 used when generating $\bar{y}_1$ is the empty set. The attention is computed only over the first utterance because no previous utterances exist. The user then provides the next utterance or concludes the interaction. At turn i, the utterance $\bar{x}_i$ is encoded using the discourse state $\mathbf{h}_{i-1}^I$, the discourse state $\mathbf{h}_i^I$ is computed, and the query $\bar{y}_i$ is generated using the set of segments S_{i-1}. The model has no access to future utterances. We use greedy inference for generation. Figure 3 illustrates a single decoding step.

5 Learning

We assume access to a training set of N interactions $\{\bar{I}^{(l)}\}_{l=1}^N$. Given an interaction $\bar{I}^{(l)}$, each utterance $\bar{x}_i^{(l)}$ where $1 \leq i \leq |\bar{I}^{(l)}|$, is paired with an annotated query $\bar{y}_i^{(l)}$. The set of segments from previous utterances is deterministically extracted from the annotated queries during learning. However, the data does not indicate what parts of each query originate in segments copied from previous utterances. We adopt a simple approach and heuristically identify context-dependent segments based on entities that appear in the utterance and the query.[5] Once we identify a segment in the annotated query, we replace it with a unique placeholder token, and it appears to the learning algorithm as a single generation decision. Treating this decision as latent is an important direction for future work. Given the segment copy decisions, we minimize the token cross-entropy loss:

$$\mathcal{L}(y_{i,k}^{(l)}) = -\log P\left(y_{i,k}^{(l)} \mid \bar{x}_i^{(l)}, \bar{y}_{i,1:k-1}^{(l)}, \bar{I}^{(l)}[:i-1]\right) \quad ,$$

where k is the index of the output token. The base and recent-history encoders (Sections 4.1 and 4.2) can be trained by processing each utterance separately. For these models, given a mini-batch $\mathcal{B}$ of utterances, each identified by an interaction-utterance index pair, the loss is the mean token loss

$$\mathcal{L} = \frac{1}{\sum_{(i,j)\in\mathcal{B}} |\bar{y}_i^{(j)}|} \sum_{(i,j)\in\mathcal{B}} \sum_{k=1}^{|\bar{y}_i^{(j)}|} \mathcal{L}(y_{i,k}^{(j)}) \quad .$$

The turn-level encoder (Section 4.3) requires building a computation graph for the entire interaction. We update the model parameters for each interaction. The interaction loss is

$$\mathcal{L} = \frac{n}{B} \frac{1}{\sum_{i=1}^{n} |\bar{y}_i^{(j)}|} \sum_{i=1}^{n} \sum_{k=1}^{|\bar{y}_i^{(j)}|} \mathcal{L}(y_{i,k}^{(j)}) \quad ,$$

where B is the batch size, and $\frac{n}{B}$ re-normalizes the loss so the gradient magnitude is not dependent on the number of utterances in the interaction. Our ablations ($-$batch re-weight in Table 2) shows the importance of this term. For both cases, we use teacher forcing (Williams and Zipser, 1989).

6 Reasoning with Anonymized Tokens

An important practical consideration for generation in ATIS and other database domains is reasoning about database values, such as entities, times, and dates. For example, the first utterance in Figure 2 includes two entities and a date reference. With limited data, learning to both reason about a large number of entities and to resolve dates are challenging for neural network models. Following previous work (Dong and Lapata, 2016; Iyer et al., 2017), we address this with anonymization, where the data is pre- and post-processed to abstract over tokens that can be heuristically resolved to tokens in the query language. In contrast to previous work, we design a special scoring function to anonymized tokens to reflect how they are used in the input utterances. Figure 4 illustrates preprocessing in ATIS. For example, we use a temporal semantic parser to resolve dates (e.g., *next*

[5] The alignment is detailed in the supplementary material.

Original utterance and query:

$\bar{x}_1$: *show me flights from seattle to boston next monday*

```
ȳ1: ( SELECT DISTINCT flight.flight_id ...
    city.city_name = 'SEATTLE' ...  city.city_name
    = 'BOSTON' ...  date_day.year = 1993 AND
    date_day.month_number = 2 AND date_day.day_number
    = 8 ...
```

Anonymized utterance and query:

$\bar{x}'_1$: *show me flights from CITY#1 to CITY#2 DAY#1 MONTH#1 YEAR#1*

```
ȳ'1: ( SELECT DISTINCT flight.flight_id ...
    city.city_name = CITY#1 ...  city.city_name
    = CITY#2 ...  date_day.year = YEAR#1
    AND date_day.month_number = MONTH#1 AND
    date_day.day_number = DAY#1 ...
```

Anonymization mapping:

CITY#1	'SEATTLE'	MONTH#1	2
CITY#2	'BOSTON'	YEAR#1	1993
DAY#1	8		

Figure 4: An example of date and entity anonymization pre-processing for $\bar{x}_1$ and $\bar{y}_1$ in Figure 2.

Monday) and replace them with day, month, and year placeholders. To anonymize database entries, we use a dictionary compiled from the database (e.g., to map *Seattle* to `SEATTLE`). The full details of the anonymization procedure are provided in the supplementary material. Following preprocessing, the model reasons about encoding and generation of anonymized tokens (e.g., `CITY#1`) in addition to regular output tokens and query segments from the interaction history. Anonymized tokens are typed (e.g., `CITY`), map to a token in the query language (e.g., `'BOSTON'`), and appear both in input utterances and generated queries.

We modify our encoder and decoder embedding functions (ϕ^x and ϕ^y) to map anonymized tokens to the embeddings of their types (e.g., `CITY`). The type embeddings in ϕ^x and ϕ^y are separate. Using the types only, while ignoring the indices, avoids learning biases that arise from the arbitrary ordering of the tokens in the training data. However, it does not allow distinguishing between entries with the same type for generation decisions; for example, the common case where multiple cities are mentioned in an interaction. We address this by scoring anonymized token based on the magnitude of attention assigned to them at generation step k. The attention magnitude is computed from the encoder hidden states. This computation considers both the decoder state and the location of the anonymized tokens in the input utterances to account for how they are used in the interaction. The probability of an anonymized token w at generation step k is

$$P(y_{i,k} = w \mid \bar{x}_i, \bar{y}_{i,1:k-1}, \bar{I}[:i-1]) \propto \sum_{t=i-h}^{i} \sum_{j=1}^{|\bar{x}_t|} (\exp(s_k(t,j)))$$

where $s_k(t,j)$ is the attention score computed in Equation 8. This probability is normalized together with the probabilities in Equations 7 and 9 to form the complete output probability.

Mean/max utterances per interaction	7.0 / 64
Mean/max tokens per utterance	10.2 / 47
Mean/max token per SQL query	102.9 / 1286
Input vocabulary size	1582
Output vocabulary size	982

Table 1: ATIS data statistics.

7 Experimental Setup

Hyperparameters, architecture details, and other experimental choices are detailed in the supplementary material.

Data We use ATIS (Hemphill et al., 1990; Dahl et al., 1994) to evaluate our approach. The data was originally collected using wizard-of-oz experiments, and annotated with SQL queries. Each interaction was based on a scenario given to a user. We observed that the original data split shares scenarios between the train, development, and test splits. This introduces biases, where travel patterns that appeared during training repeat in testing. For example, a model trained on the original data split often correctly resolves the exact referenced by *on Saturday* with no pre-processing or access to the document date. We evaluate this overfitting empirically in the supplementary material. We re-split the data to avoid this bias. We evenly distribute scenarios across splits so that each split contains both scenarios with many and few representative interactions. The new split follows the original split sizes with 1148/380/130 train/dev/test interactions. Table 1 shows data statistics. The system uses a SQL database of 27 tables and 162K entries. 96.6% of the queries require at least one join, and 93% at least two joins. The most related work on ATIS to ours is Miller et al. (1996), which we discuss in Section 3.

The most related corpora to ATIS are SCONE (Long et al., 2016) and SequentialQA (Iyyer et al., 2017). SCONE (Long et al., 2016) contains micro-domains consisting of stack- or list-like elements. The formal representation is linguistically-motivated and the majority of queries include a single binary predicate. All interactions include five turns. SequentialQA (Iyyer et al., 2017) contains sequences of questions on a single Wikipedia table. Interactions are on average 2.9 turns long, and were created by re-phrasing a question from a context-independent corpus (Pasupat and Liang, 2015). In contrast, ATIS uses a significantly larger

database, requires generating complex queries with multiple joins, includes longer interactions, and was collected through interaction with users. The supplementary material contains analysis of the contextual phenomena observed in ATIS.

Pre-processing We pre-process the data to identify and anonymize entities (e.g., cities), numbers, times, and dates. We use string matching heuristics to identify entities and numbers, and identify and resolve times and dates using UWTime (Lee et al., 2014). When resolving dates we use the original interaction date as the document time. The supplementary material details this process.

Metrics We evaluate using query accuracy, strict denotation accuracy, and relaxed denotation accuracy. Query accuracy is the percentage of predicted queries that match the reference query. Strict denotation accuracy is the percentage of predicted queries that execute to exactly the same table as the reference query. In contrast to strict, relaxed gives credit to a prediction query that fails to execute if the reference table is empty. In cases when the utterance is ambiguous and there are multiple gold queries, we consider the query or table correct if they match any of the gold labels.

Systems We evaluate four systems: (a) SEQ2SEQ-0: the baseline encoder-decoder model (Section 4.1); (b) SEQ2SEQ-H: encoder-decoder with attention on current and previous utterances (Section 4.2); (c) S2S+ANON: encoder-decoder with attention on previous utterances and anonymization scoring (Section 6); and (d) FULL: the complete approach including segment copying (Section 4.4). For FULL, we evaluate with predicted and gold (FULL-GOLD) previous queries, and without attention on previous utterances (FULL-0). All models except SEQ2SEQ-0 and FULL-0 use $h = 3$ previous utterances. We limit segment copying to segments that appear in the most recent query only.[6] Unless specifically ablated, all experiments use pre-processing.

8 Results

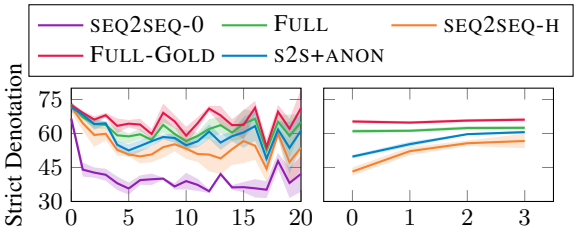
Table 2 shows development and test results. We run each experiment five times and report mean and standard deviation. The main metric we focus on is strict denotation accuracy. The relatively low performance of SEQ2SEQ-0 demonstrates the need for context in this task. Attending on recent history significantly increases performance. Both SEQ2SEQ models score anonymized tokens as regular vocabulary tokens. Adding anonymized token scoring further increases performance (S2S+ANON). FULL-0 and FULL add segment copying and the turn-level encoder. The relatively high performance of FULL-0 shows that substituting segment copying with attention maintains and even improves the system effectiveness. However, the best performance is provided with FULL, which combines both. This shows the benefit of redundancy in accessing contextual information. Unlike the other systems, both FULL and FULL-0 suffer from cascading errors due to selecting query segments from previously incorrect predictions. The higher FULL-GOLD performance illustrates the influence of error propagation. While part of this error can be mitigated by having both attention and segment copying, this behavior is unlikely to be learned from supervised learning, where errors are never observed.

Model	Query	Denotation	
		Relaxed	Strict
Development Results			
SEQ2SEQ-0	28.7±1.7	48.8±1.4	43.2±1.8
SEQ2SEQ-H	35.1±2.2	59.4±2.4	56.7±3.2
S2S+ANON	37.6±0.7	61.6±0.7	60.6±0.7
FULL-0	36.3±0.5	61.5±0.8	61.0±0.9
FULL	37.5±0.9	63.0±0.7	**62.5**±0.9
– turn-level enc.	37.4±1.5	62.1±2.5	61.4±2.7
– batch re-weight	36.4±0.6	61.8±0.3	61.5±0.4
– input pos. embs.	33.3±0.2	57.9±0.8	57.4±0.8
– query segments	36.0±0.9	59.5±1.3	58.3±1.4
– anon. scoring	35.7±0.5	60.8±1.1	60.0±1.0
– pre-processing	26.4±6.1	53.3±8.6	53.0±8.5
FULL-GOLD	42.1±0.8	66.6±0.7	66.1±0.7
Test Results			
SEQ2SEQ-0	35.7±1.5	56.4±1.1	53.8±1.0
SEQ2SEQ-H	42.2±2.0	66.6±3.2	65.8±3.4
S2S+ANON	44.0±1.2	69.3±1.0	68.6±1.1
FULL-0	43.1±1.3	67.8±1.6	67.2±1.6
FULL	43.6±1.0	69.3±0.8	**69.2**±0.8
FULL-GOLD	47.4±1.3	72.3±0.5	72.0±0.5

Table 2: Mean and standard deviation development and test results, including ablations on the FULL model.

Ablations show that all components contribute to the system performance. Performance drops when using a concatenation-based encoder instead of the turn-level encoder (−turn-level enc.; Section 4.3). Using batch-reweighting (−batch-reweight; Section 5) and input position embeddings (−input pos. embs.; Section 4.3) are critical to the performance of the turn-level encoder. Removing copying of query segments

[6] While we only use segments from the most recent query, they often appear for the first time much earlier in the interaction, which influences their absolute position value a.



(a) Interaction turns (b) History length h

Figure 5: Mean development strict denotation accuracy as function of turns and h.

Model	Query	Denotation	
		Relaxed	Strict
Development Results			
S2S+ANON	44.4±1.2	69.9±0.3	68.9±0.3
FULL	42.8±0.2	68.8±0.2	68.4±0.2
FULL-GOLD	47.5±0.2	71.5±0.4	70.7±0.6
Test Results			
S2S+ANON	43.9±0.3	67.4±1.0	67.2±1.0
FULL	44.3±0.2	66.3±0.3	66.3±0.4
FULL-GOLD	47.2±0.3	68.2±0.5	67.9±0.4

Table 3: Results on the original split of the data.

from the interaction history lowers performance (−query segments; Section 4.4). Treating indexed anonymized tokens as regular tokens, rather than using attention-based scoring and type embeddings, lowers performance (−anon. scoring; Section 6). Finally, pre-processing, which includes anonymization, is critical (−pre-processing).

Figure 5(a) shows the performance as interactions progress. All systems show a drop in performance after the first utterance, which is always context-independent. As expected, SEQ2SEQ-0 shows the biggest drop. The FULL approach is the most stable as the interaction progresses.

Figure 5(b) shows the performance as we decrease the number of previous utterances used for attention h. Without the turn-level encoder and segment copying (SEQ2SEQ-H and S2S+ANON), performance decreases significantly as h decreases. In contrast, the FULL model shows a smaller decrease (1.5%). The supplementary material includes attention analysis demonstrating the importance of previous-utterance attention. However, attending on fewer utterances improves inference speed: FULL-0 is 30% faster than FULL.

Finally, while we re-split the data due to scenario sharing between train and test early in development and used this split only for development, we also evaluate on the original split (Table 3). We report mean and standard deviation over three trials. The high performance of S2S+ANON potentially indicates it benefits more from the differences between the splitting procedures.

9 Analysis

We analyze errors made by the full model on thirty development interactions. When analyzing the output of FULL, we focus on error propagation and analyze predictions that resulted in an incorrect table when using FULL, but a correct table when using FULL-GOLD. 56.7% are due to selection of a segment that contained an incorrect constraint. 43.4% of the errors are caused by a necessary segment missing during generation. 93.0% of all predictions are valid SQL and follow the database schema. We also analyze the errors of FULL-GOLD. We observe that 30.0% of errors are due to generating constraints that were not mentioned by the user. Other common errors include generating relevant constraints with incorrect values (23.3%) and missing constraints (23.3%).

We also evaluate our model's ability to recover long-distance references while constraints are added, changed, or removed, and when target attributes change. The supplementary material includes the analysis details. In general, the model resolves references well. However, it fails to recover constraints mentioned in the past following a user focus state change (Grosz and Sidner, 1986).

10 Discussion

We study models that recover context-dependent executable representations from user utterances by reasoning about interaction history. We observe that our segment-copying models suffer from error propagation when extracting segments from previously-generated queries. This could be mitigated by training a model to ignore erroneous segments, and recover by relying on attention for generation. However, because supervised learning does not expose the model to erroneous states, a different learning approach is required. Our analysis demonstrates that our model is relatively insensitive to interaction length, and is able to recover both explicit and implicit references to previously-mentioned entities and constraints. Further study of user focus change is required, an important phenomenon that is relatively rare in ATIS.

Acknowledgements

This research was supported by the NSF (CRII-1656998), Schmidt Sciences, a gift from Google, and cloud computing credits from Amazon. We thank Valts Blukis, Luke Zettlemoyer, and the anonymous reviewers for their helpful comments.

References

Yoav Artzi and Luke Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. http://www.aclweb.org/anthology/D11-1039.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. http://www.aclweb.org/anthology/D13-1160.

Jianpeng Cheng, Siva Reddy, Vijay Saraswat, and Mirella Lapata. 2017. Learning structured natural language representations for semantic parsing. In *Proceedings Association for Computational Linguistics*. https://doi.org/10.18653/v1/P17-1005.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. https://doi.org/10.3115/v1/D14-1179.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of the Conference on Computational Natural Language Learning*. http://www.aclweb.org/anthology/W10-2903.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Proceedings of the Workshop on Human Language Technology*. http://www.aclweb.org/anthology/H94-1010.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the Association for Computational Linguistics*. https://doi.org/10.18653/v1/P16-1004.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science* 14:179–211.

Alessandra Giordani and Alessandro Moschitti. 2012. Translating questions to SQL queries with generative parsers discriminatively reranked. In *Conference on Computational Linguistics*. http://www.aclweb.org/anthology/C12-2040.

Barbara J. Grosz and Candace L. Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics* 12(3). http://www.aclweb.org/anthology/J86-3001.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The ATIS spoken language systems pilot corpus. In *Proceedings of the DARPA Speech and Natural Language Workshop*. http://www.aclweb.org/anthology/H90-1021.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proceedings of the the Association for Computational Linguistics*. https://doi.org/10.18653/v1/P17-1089.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the Association for Computational Linguistics*. https://doi.org/10.18653/v1/P17-1167.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the Association for Computational Linguistics*. https://doi.org/10.18653/v1/P16-1002.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of theConference on Empirical Methods in Natural Language Processing*. https://www.aclweb.org/anthology/D17-1160.

Nate Kushman and Regina Barzilay. 2013. Using semantic unification to generate regular expressions from natural language. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. http://www.aclweb.org/anthology/N13-1103.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*. http://www.aclweb.org/anthology/D11-1140.

Kenton Lee, Yoav Artzi, Jesse Dodge, and Luke Zettlemoyer. 2014. Context-dependent semantic parsing for time expressions. In *Proceedings of the Association for Computational Linguistics*. https://doi.org/10.3115/v1/P14-1135.

Xiujun Li, Yun-Nung Chen, Lihong Li, and Jianfeng Gao. 2017. End-to-end task-completion neural dialogue systems. *CoRR* abs/1703.01008.

Percy Liang, Michael Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the Association for Computational Linguistics: Human Language Technologies*. http://www.aclweb.org/anthology/P11-1060.

Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, Fumin Wang, and Andrew Senior. 2016. Latent predictor networks for code generation. *CoRR* abs/1603.06744.

Reginald Long, Panupong Pasupat, and Percy Liang. 2016. Simpler context-dependent logical forms via model projections. In *Proceedings of the Association for Computational Linguistics*. https://doi.org/10.18653/v1/P16-1138.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. https://doi.org/10.18653/v1/D15-1166.

Scott Miller, David Stallard, Robert Bobrow, and Richard Schwartz. 1996. A fully statistical approach to natural language interfaces. In *Proceedings of the Association for Computational Linguistics*. http://www.aclweb.org/anthology/P96-1008.

Vincent Ng. 2010. Supervised noun phrase coreference research: The first fifteen years. In *Proceedings of the Association for Computational Linguistics*. http://www.aclweb.org/anthology/P10-1142.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the Association for Computational Linguistics*. https://doi.org/10.3115/v1/P15-1142.

Baolin Peng, Xiujun Li, Lihong Li, Jianfeng Gao, Asli Celikyilmaz, Sungjin Lee, and Kam-Fai Wong. 2017. Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. https://aclanthology.info/pdf/D/D17/D17-1236.pdf.

Hoifung Poon. 2013. Grounded unsupervised semantic parsing. In *Proceedings of the Association for Computational Linguistics*. http://www.aclweb.org/anthology/P13-1092.

Ana-Maria Popescu, Alex Armanasu, Oren Etzioni, David Ko, and Alexander Yates. 2004. Modern natural language interfaces to databases: Composing statistical parsing with semantic tractability. In *International Conference on Computational Linguistics*. http://www.aclweb.org/anthology/C04-1021.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *Proceedings of the Association for Computational Linguistics*. https://doi.org/10.18653/v1/P17-1105.

Iulian Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Association for the Advancement of Artificial Intelligence*.

Iulian Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Association for the Advancement of Artificial Intelligence*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Neural Information Processing Systems*.

Gökhan Tür, Dilek Z. Hakkani-Tür, and Larry P. Heck. 2010. What is left to be understood in ATIS? In *Proceedings of the Spoken Language Technology Workshop*.

Adrienne Wang, Tom Kwiatkowski, and Luke Zettlemoyer. 2014. Morpho-syntactic lexical generalization for CCG semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. https://doi.org/10.3115/v1/D14-1135.

Ronald J. Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation* 1:270–280.

Xiaojun Xu, Chang Liu, and Dawn Xiaodong Song. 2017. SQLNet: Generating structured queries from natural language without reinforcement learning. *CoRR* abs/1711.04436.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of the Association for Computational Linguistics*. https://doi.org/10.18653/v1/P17-1041.

John M. Zelle and Raymond J. Mooney. 1996. Comparative results on using inductive logic programming for corpus-based parser construction. In *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing*.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

Luke S. Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. http://www.aclweb.org/anthology/D07-1071.

Luke S. Zettlemoyer and Michael Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *Proceedings of the Joint Conference of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing of the AFNLP*. http://www.aclweb.org/anthology/P09-1110.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating structured queries from natural language using reinforcement learning. *CoRR* abs/1709.00103.

Neural Text Generation in Stories Using Entity Representations as Context

Elizabeth Clark Yangfeng Ji Noah A. Smith
Paul G. Allen School of Computer Science & Engineering
University of Washington
Seattle, WA, USA
{eaclark7,yangfeng,nasmith}@cs.washington.edu

Abstract

We introduce an approach to neural text generation that explicitly represents entities mentioned in the text. Entity representations are vectors that are updated as the text proceeds; they are designed specifically for narrative text like fiction or news stories. Our experiments demonstrate that modeling entities offers a benefit in two automatic evaluations: mention generation (in which a model chooses which entity to mention next and which words to use in the mention) and selection between a correct next sentence and a distractor from later in the same story. We also conduct a human evaluation on automatically generated text in story contexts; this study supports our emphasis on entities and suggests directions for further research.

1 Introduction

We consider the problem of automatically generating narrative text, a challenging problem at the junction of computational creativity and language technologies (Gervás, 2009). We are motivated in particular by potential applications in personalized education and assistive tools for human authors, though we believe narrative might also play a role in social conversational agents (Sordoni et al., 2015). In this work, the term "narrative text" refers primarily to fiction but might also include news and other kinds of stories.

A notable difference between longstanding work in natural language generation and recent "neural" models is in the treatment of *entities* and the words used to refer to them. Particularly in the generation of narrative text, character-centered generation has been shown important in character dialogue generation (Walker et al., 2011; Cavazza and Charles, 2005) and story planning (Cavazza et al., 2002). Neural models, on the other hand, treat mentions as just more words, relying on representation learning to relate the people in a story through the words alone.

Context	All of a sudden, [*Emily*]$_1$ walked towards [*the dragon*]$_2$.
Current Sentence	[*Seth*]$_3$ yelled at [*her*]$_1$ to get back but _____

Figure 1: An example of entity-labeled story data. The brackets indicate which words are part of entity mentions. Mentions marked with the same number refer to the same entity. The goal is to continue the story in a coherent way. The actual story reads, "*Seth yelled at her to get back but she ignored him.*"

Entities are an important element of narrative text. Centering Theory places entities at the center of explaining what makes text coherent (Grosz et al., 1995). In this work, we incorporate entities into neural text generation models; each entity in a story is given its own vector representation, which is updated as the story unfolds. These representations are learned specifically to predict words—both mentions of the entity itself and also the following context. At a given moment in the story, the current representations of the entities help to predict what happens next.

Consider the example in Figure 1. Given the context, the reader expects the subsequent words and sentences of the passage to track the results of Emily approaching the dragon. Future text should include references to Emily's character and the dragon and the result of their interaction. The choice of entity generated next in the sentence will change what language should follow that mention and will shape and drive the direction of the story. For this reason, we propose using entity representations as context for generation.

Of course, entities are not the only context needed for coherent language generation; previously generated content remains an important source of information. We use a simple, parameter-free method for combining preceding context with entity context within an end-to-end–

Proceedings of NAACL-HLT 2018, pages 2250–2260
New Orleans, Louisiana, June 1 - 6, 2018.

trainable neural language generator.

We evaluate our model's performance through two automatic evaluation tasks. The first is a new mention generation task inspired by earlier work in referring expression generation (Dale and Reiter, 1995). The second is a sentence selection task inspired by coherence tests from Barzilay and Lapata (2008). Our model outperforms strong baselines on both tasks.

We further conduct a human evaluation in which our model's generated sentences are compared to a strong baseline model. This evaluation elucidates strengths and weaknesses of our model and offers guidance for future work on narrative text generation.

2 Model Description

We propose an entity-based generation model (ENGEN)[1] that combines three different sources of contextual information for text generation:

1. The content that has already been generated within the current sentence
2. The content that was generated in the previous sentence
3. The current state of the entities mentioned in the document so far

Each of these types of information is encoded in vector form, following extensive past work on recurrent neural network (RNN) language models. The first source of context is the familiar hidden state vector of the RNN; more precisely, our starting point is a sequence-to-sequence model (Sutskever et al., 2014). Representations of the second and third forms of context are discussed in §2.1 and §2.2, respectively. The combination of all three context representations is described in §2.3.

2.1 Context from Previous Sentence

As noted, our starting point is a sequence-to-sequence model (Sutskever et al., 2014); the last hidden state from the previous sentence offers a representation of the preceding context. We add an attention mechanism (Bahdanau et al., 2015). Let $\boldsymbol{h}_{t,i}$ and $\boldsymbol{h}_{t-1,j}$ be the LSTM hidden states of sentence t at timestep i and the previous sentence $t-1$ at timestep j, where j ranges over the number of words in the previous sentence. To summarize the contextual information from the previous sentence for predicting the next word at timestep $i+1$ in sentence t, we have

$$\boldsymbol{p}_{t-1,i} = \sum_j \alpha_{i,j} \boldsymbol{h}_{t-1,j}, \text{where} \quad (1)$$

$$\alpha_{i,j} = \frac{\exp(\boldsymbol{h}_{t-1,j} \mathbf{W}_a \boldsymbol{h}_{t,i})}{\sum_{j'} \exp(\boldsymbol{h}_{t-1,j'} \mathbf{W}_a \boldsymbol{h}_{t,i})} \quad (2)$$

is the attention weight for $\boldsymbol{h}_{t-1,j}$. Unlike the definition of attention in Bahdanau et al. (2015), here we use the bilinear product in Equation 2 to encourage correlation between $\boldsymbol{h}_{t,i}$ and $\boldsymbol{h}_{t-1,j}$ for coherence in text generation. In §2.3, we will combine this with $\boldsymbol{h}_{t,i}$ for predicting the next word; we refer to that model as S2SA, and it serves as an entity-unaware baseline in our experiments.

2.2 Context from Entities

In S2SA, the context of a sentence is (at best) represented by compressing information about the words that have appeared in the previous sentence. Past research has suggested several approaches to capturing other contextual information. For example, Lau et al. (2017) and Ghosh et al. (2016) have sought to capture longer contexts by modeling topics. Recently, Ji et al. (2017) introduced a language model, ENTITYNLM, that adds explicit tracking of entities, which have their own representations that are updated as the document progresses.[2] That model was introduced for analysis tasks, such as language modeling and coreference resolution, where the texts (and their coreference information) are given, and the model is used to score the texts to help resolve coreference relationships.[3] ENTITYNLM's strong performance on language modeling suggests the potential of distributed entity representations as another source of contextual information for text generation. Inspired by that work, we maintain the dynamic representation of entities and use them as contextual information when generating text.

In general, every entity in a document (e.g., EMILY in Figure 1) is assigned a vector representation; this vector is updated every time the entity is mentioned. This is entirely appropriate for generating narrative stories in which characters develop and change over long contexts. When we

[1] Code available at github.com/eaclark07/engen.

[2] Because space does not permit a full exposition of all the details of ENTITYNLM, we refer the interested reader to Ji et al. (2017).

[3] The entity prediction task used in their work is relevant to our mention generation task, which will be discussed in §5.

generate text, the model will have access to the current representation of every participant (i.e., every entity) in the story at that time (denoted by $\boldsymbol{e}_{i,t}$ for entity i at timestep t).

When choosing which entity is referred to at timestep t, there are $m+1$ options, where m is the number of entities tracked in the document so far—the $(m+1)$th is for a new, previously unmentioned entity. Given that a word is part of an entity mention and given the previous hidden state, the probability that the word is referring to a given entity $i \in \{1, \ldots, m+1\}$ is proportional to:

$$\exp(\boldsymbol{h}_{t-1}^{\top} \mathbf{W}_{entity} \boldsymbol{e}_{i,t-1} + \boldsymbol{w}_{dist}^{\top} \boldsymbol{f}(i)), \quad (3)$$

where $\mathbf{W}_{entity}$ is a weight matrix for predicting the entities and $\boldsymbol{w}_{dist}^{\top} \boldsymbol{f}(i)$ is a term that takes into account distance features between the current and past entity mentions.

Once an entity is selected, its vector is assigned to $\boldsymbol{e}_{current}$, which is used to generate the word w_t. If the model decided the current word should not refer to an entity, then $\boldsymbol{e}_{current}$ is still used and will be the representation of the most recently mentioned entity. If the choice is a new, previously unmentioned entity, then $\boldsymbol{e}_{current}$ is initialized with a new embedding randomly generated from a normal distribution:

$$\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{r}, \sigma^2 \mathbf{I}), \quad (4)$$

where $\sigma = 0.01$ and $\boldsymbol{r}$ is a parameter vector that is used to determine whether the next word should refer to an entity.

Once the word w_t has been generated, the entity representation is updated based on the new hidden state information, $\boldsymbol{h}_t$.

2.3 Combining Contexts

Our new model merges S2SA and ENTITYNLM. Both provide a representation of context: respectively, the previous sentence's representation ($\boldsymbol{p}_t$) and the most salient entity's representation ($\boldsymbol{e}_{current}$). The hidden state $\boldsymbol{h}_{t-1}$ is, of course, also available, and is intended to capture local contextual effects. The challenge is how to combine these representations effectively for text generation.

In this work, for simplicity, we choose a combination function without any extra parameters, and leave the detailed investigation of paramaterized composition functions as future work. We use a max-pooling function to form a context vector $\boldsymbol{c}_t$ with the same dimensionality as $\boldsymbol{h}_{t-1}$ (and, of course, $\boldsymbol{p}_t$ and $\boldsymbol{e}_{current}$). Specifically, at time step t, each element of the combined context vector $\boldsymbol{c}_t$ is calculated as follows. For $k \in \{1, ..., |\boldsymbol{c}_t|\}$,

$$\boldsymbol{c}_t[k] = \max(\boldsymbol{h}_{t-1}[k], \boldsymbol{p}_t[k], \boldsymbol{e}_{current}[k]). \quad (5)$$

The max pooling technique originates from the design of convolutional neural networks and has been found useful elsewhere in NLP (Kalchbrenner et al., 2014). Other alternatives, including average pooling, min pooling, and element-wise multiplication on all three vectors, were considered in informal preliminary experiments on development data and found less effective than max pooling.

This combined context vector $\boldsymbol{c}_t$ is used to generate word w_t by calculating the probability of each word type in the vocabulary. We use a class-factored softmax function (Goodman, 2001; Baltescu and Blunsom, 2015). This choice greatly reduces the runtime of word prediction. In practice, we often find it gives better performance than standard softmax.

2.4 Learning

The training objective is to maximize the log-probability of $\boldsymbol{X}$:

$$\ell(\boldsymbol{\theta}) = \log P(\boldsymbol{X}; \boldsymbol{\theta}) = \sum_t \log P(X_t; \boldsymbol{\theta}) \quad (6)$$

$\boldsymbol{\theta}$ denotes all of the model's parameters. X_t represents all decisions at timestep t about the word (whether it is part of a entity mention, and if so, the entity the mention refers to, the length of the mention, and the word itself).

These decisions are made by calculating probabilities for each available option using the current state of the neural network (a vector) and the current vector representations of the entities. Given the probabilities, the next word is assumed to have been randomly generated by sampling.

While we might consider training the model to maximize the probability of the generated words directly, treating the entity-related variables as *latent*, this would create a mismatch between how we train and use the model. For generation, the model explicitly predicts not just the word, but also the entity information associated with that word. Training with latent variables is also expensive. For these reasons, we use the same training method used for ENTITYNLM, which requires

training data annotated with mention and coreference information (entity clusters).

2.5 Variants

In our experiments, we consider the combined model (ENGEN) and two ablations: S2SA and a model similar to ENTITYNLM. Note that, unlike past work with previous-sentence context, S2SA uses max pooling for $\boldsymbol{h}_{t-1}$ and $\boldsymbol{p}_t$ and class-factored softmax; our version of ENTITYNLM also uses max pooling and class-factored softmax. All of these models are trained in a similar way.

3 Implementation Details

The models are implemented using DyNet (Neubig et al., 2017) with GPU support. We optimize with SGD, with a learning rate of $\lambda = 0.1$. The dimensions of input layer, hidden layer, and entity representation are fixed at 512 (hyperparameter optimization might lead to better solutions). The input word embeddings are randomly initialized with the default method in DyNet and updated during training jointly with other parameters. For class-factored softmax, we use 160 Brown clusters (Brown et al., 1992; Liang, 2005) estimated from the training data.

4 Data

We trained all models on 312 adventure books from the Toronto Book Corpus (Zhu et al., 2015), with development and test sets of an additional 39 books each. We divided the books into smaller segments, where each segment includes up to 50 sentences. There are 33,279 segments in the training set, 4,577 in the dev. set, and 4,037 in the test set. This helps with memory efficiency, allowing us to train the model without building a recurrent neural network on the entire book.

All the tokens in the data were downcased, and numbers were replaced with a special NUM token. The vocabulary was selected by replacing the lowest frequency (less than 10) word types with a special UNK token. There are 43 million tokens, and the vocabulary size is 35,443.

To obtain entity annotations, we used the Stanford CoreNLP system (Clark and Manning, 2016a,b), version 3.8.0. From the coreference resolution results, we noticed that some entity mentions include more than 70 tokens, which is likely in error. To simplify the problem, we only kept the mentions consisting of three words or fewer, which covers more than 95% of the mentions in the training data. For mentions of more than three words, we replaced them with their head word, as determined by the Stanford CoreNLP system. While truncating these mentions sacrifices some information, we believe this preprocessing step is justifed as it retains most character names and pronouns, an especially important entity type for stories.

Of course, the use of automatic annotations from a coreference system will introduce noise and risks "confusing" the entity-aware models. The benefit is that we were able to train on a much larger corpus than any existing coreference dataset (e.g., the CoNLL 2012 English shared task training set has only 1.3 million tokens; Pradhan et al., 2012). Further, a corpus of books offers language that is much closer to our intended narrative text generation applications. Our experiments aim to measure some aspects of our models' intrinsic correctness, though we emphasize that even if entity information is incorrect at training time, it may still be helpful.

For all experiments, the same preprocessed dataset and trained models were used. The best models were selected based on development set log likelihood (Equation 6).

5 Experiment: Mention Generation

The goal of our first experiment is to investigate each model's capacity to mention an entity in context. For example, in Figure 1, *Emily* and *her* are both possible mentions of EMILY's character, but the two cannot be used interchangeably. Inspired by early work on referring expression generation (Dale and Reiter, 1995) and recent work on entity prediction (Modi et al., 2017), we propose a new task we call *mention generation*. Given a text and a slot to be filled with an entity mention, a model must choose among all preceding entity mentions and the correct mention. So if the model was choosing the next entity mention to be generated in Figure 1, it would select between all the previous entity mentions (*Emily*, *the dragon*, *Seth*, and *her*) and the correct mention (*she*).

In our model, each candidate mention is augmented with the index of its entity. Therefore, performing well on this task requires choosing both the entity and the words used to refer to it; this notion of quality is our most stringent evaluation measure. It requires the greatest precision, as it is

model	cluster and mention	cluster only	mention only
1. Reverse order	0.12	0.38	0.15
2. S2SA	—	—	0.44
3. EntityNLM	0.52	0.46	0.54
4. EnGen	0.53	0.46	0.55

Table 1: MAP on the mention generation task. Note that these results can only be compared between models, not between tasks, as there are a different number of candidates for each of the tasks.

cluster and mention	cluster only	mention only
[*Emily*]$_1$	*Emily	Emily
[*the dragon*]$_2$	The Dragon	the dragon
[*Seth*]$_3$	Seth	Seth
[*her*]$_1$		her
*[*she*]$_1$		*she

Figure 2: Candidate lists for each of the mention generation tasks for completing the blank in Figure 1. The asterisk (*) indicates the correct choice.

possible to select the correct mention but not the correct cluster and vice versa. Since S2SA does not model entities, we also compare systems on quality of mentions alone (without entity clusters). For completeness, we include cluster quality for the entity-aware models. Candidate lists for each task to generate the next mention in the example in Figure 1 are shown in Figure 2.

The experiment setup does not require manual creation of candidate lists. However, it makes the mention generation task even more challenging, because the size of a candidate list can exceed 100 mention candidates.

We note that the difficulty of this task increases as we consider mention slots later and later in the document. The first mention generation choice is a trivial one, with a single candidate that is by definition correct. As more entity mentions are observed, the number of options will increase.[4] To enable aggregation across contexts of all lengths, we report the mean average precision (MAP) of the correct candidates, where the language model scores are used to rank candidates.

Baselines Along with the two ablated models (S2SA and EntityNLM), we include a "reverse order" baseline, which ranks mentions by recency (the first element in the ranking is the most recent mention, then the second-most-recent, and so on).

Results The ranking results of EnGen and other systems are reported in Table 1. A higher MAP score implies a better system. We measure the overall performance of all the systems, along with their performance on selecting the *mention only* and *entity cluster only*. Across all the evaluation measures, EnGen gives the highest MAP numbers. Recall that S2SA does not have a component for entity prediction, therefore we only compare it with EnGen in the *mention only* case. The difference between line 4 and line 2 on the *mention only* column shows the benefit of adding entity representations for text generation. The difference between lines 3 and 4 shows that local context also gives a small boost. Although the distance between the current slot and previous entity mention has been shown as a useful feature in coreference resolution (Clark and Manning, 2016b), line 1 shows distance alone is not an effective heuristic for mention generation.

6 Experiment: Pairwise Sentence Selection

The sentence selection task is inspired by tests of coherence used to assess text generation components automatically, without human evaluation (Barzilay and Lapata, 2008). It serves as a sanity check, as it was conducted prior to full generation and human evaluations (§7). Since the models under consideration are generative, they can be used to assign scores to candidate sentences, given a context.

In our version of this task, we provide a model with $n - 1 = 49$ sentences of preceding context, and offer two choices for the nth (50th) sentence: the actual 50th sentence or a distractor sentence randomly chosen from the next 50 sentences. A random baseline would achieve 50% accuracy.

Because the distractor comes from the same

[4] Note that the list of candidates may include duplicate entries with the same mention words and cluster. These are collapsed since they will have the same score under a language model.

Context	All of a sudden, [*Emily*]$_1$ walked towards [*the dragon*]$_2$.
1.	[*Seth*]$_3$ yelled at [*her*]$_1$ to get back but [*she*]$_1$ ignored [*him*]$_3$.
2.	[*She*]$_1$ patted [*its head*]$_4$ and [*it*]$_2$ curled up outside [*the cave*]$_5$.
3.	"[*Emily*]$_1$, how did [*you*]$_1$ keep [*that dragon*]$_2$ from attacking [*us*]$_6$?"

Figure 3: A passage's last sentence of context, and 3 sentences from various points in the next passage.

model	mean accuracy	s.d.
1. S2SA	0.546	0.01
2. ENTITYNLM	0.534	0.006
3. ENGEN	*0.566	0.008

* signficantly better than lines 1 and 2 with $p < 0.05$.

Table 2: Accuracy in choosing the actual next sentence, given 49 sentences of context, with a distractor from slightly later in the story. The mean accuracies and standard deviation are calculated across the five rounds of pairwise sentence selection.

story (with similar language, characters, and topics) and relatively nearby (in 2% cases, the very next sentence), this is not a trivial task. Consider the example in Figure 3. All of the sentences share lexical and entity information with the last line of the context. However, the first sentence immediately follows the context, while the second and third sentences are 10 lines and 48 lines away from the context, respectively. These entity and lexical similarities make distinguishing the actual sentence from the random sentence a challenging problem for the model.

To select the sentence, the model scores each of the two candidate sentences based on its probability on words and all entity-related information as defined in Equation 6. (Both candidate sentences come from the preprocessed data and have the entity annotations described in §4.)

The sentence that receives the higher probability is chosen. For each of the 4,037 segments of context in the test set, we calculated the accuracy of each model at distinguishing the gold sentence from a distractor sentence. We ran this pairwise decision 5 times, each time with a different set of randomly selected distractor sentences and averaged their performance across all 5 rounds.

Results The accuracy of each of the models is reported in Table 2. The best performance is obtained by ENGEN, which is significantly better than the other two models ($p < 0.05$, binomial test). Unlike the mention generation task, S2SA beats ENTITYNLM at this task; this difference in performance shows the importance of local context. Although we performed five different rounds of random sampling to choose a sentence from the following segment as the distractor sentence, the standard deviations in Table 2 show the results are generally consistent across rounds, regardless of the distractor's distance from the gold sentence.

7 Human Evaluation: Sentence Generation

The task motivating the work in this paper is narrative text generation. As such, evaluation by human judges of the quality of generated text is the best measure of our methods' quality. This study simplifies that evaluation by distilling the judgment down to a forced choice between contextually generated sentences generated by two different models. We use this task to investigate the strengths and weaknesses of our model in a downstream application. By asking humans to decide which sentences they prefer (in a given context) and to explain why, we can analyze where our model is helping and where text generation for stories still needs to improve, both with respect to entities and to other aspects of language. Here we control for training data and assess the benefit of including entity information for generating sentences to continue a story.

We presented Amazon Mechanical Turkers[5] with a short excerpt from a story and two generated sentences, one generated by ENGEN and one generated by the entity-unaware S2SA. We asked them to *"choose a sentence to continue the story"* and to briefly explain why they made the choice they did, an approach similar to that in other story-based work such as Lukin et al. (2015).

Note that we did not prime Turkers to focus on entities. Rather, the purpose of this experiment was to examine the performance of the model in a story generation setting and to get feedback on what people generally notice in generated text, not only with regard to entities. By keeping the task

[5]We selected workers who had completed over 1,000 tasks, had over a 95% task acceptance rate, and were from the United States.

open-ended, we can better analyze what people value in generated text for stories, and where our model supports that and where it doesn't.

We used a subset of 50 randomly selected text segments from the test set described in §4. However, for the human evaluation, we only used the final 60 words[6] of the story segments to keep the amount of reading and context manageable for Turkers. The models had access to the same subset of the context that the evaluator saw, not all 50 sentences from the original segment as in earlier experiments. For each context, we randomly sampled a sentence to continue the document, using each of two models: ENGEN and S2SA. These two models allowed us to see if adding the entity information noticeably improved the quality of the generation to evaluators.

Initial experiments showed that fluency remains a problem for neural text generation. To reduce the effect of fluency on Turkers' judgments, we generated 100 samples for each context/model pair and then reranked them with a 5-gram language model (Heafield, 2011) that was trained on the same training data. The two top ranked sentences (one for ENGEN and one for S2SA) were presented in random order and without reference to the models that generated them.

For each of the 50 contexts, we had 11 Turkers pick a candidate sentence to continue the story passage. Turkers were paid $0.10 for each evaluation they completed. In total, 93 Turkers completed the task. The number of passages Turkers completed ranged from 1 to all 50 story segments (with an average of 6.1). While the quantitative portion of this task would be easy to scale, the qualitative portion is not; we kept the human evaluation small, running it until reaching saturation.

Results Each pair of sentences was evaluated by 11 Turkers, so each of the passages could receive up to 11 votes for ENGEN. For 27 of the passages, the majority of Turkers (6 or more) chose the sentence from ENGEN, versus 23 passages that went to the baseline model, S2SA. The scores were close in many cases, and for several passages, Turkers noted in their explanations that while they were required to choose one sentence, both would have worked. Examples of the context and sentence pairs that were strongly in favor of ENGEN, strongly in favor of S2SA, and that received mixed reviews are shown in Table 3.

When asked to explain why they selected the sentence they did, a few Turkers attributed their choices to connections between pronouns in ENGEN's suggestions to characters mentioned in the story excerpt. However, a more frequent occurrence was Turkers citing a mismatch in entities as their reason for rejecting an option. For example, one Turker said they chose ENGEN's sentence because the S2SA sentence began with "she," and there were no female characters in the context.

Interestingly, while pronouns not mentioned in the context were cited as a reason for rejecting candidate sentences, new proper noun entity mentions were seen as an asset by some. One Turker chose a S2SA sentence that referenced "Richard," a character not present in the context, saying, *"I believe including Richard as a name gives some context of the characters of the story."* This demonstrates the importance of the ability to generate new entities, in addition to referring back to exisiting entities.

However, due to the open-ended nature of the task, the reasons Turkers cited for selecting sentences extended far beyond characters and entity mentions. In fact, most of the responses credited other aspects of stories and language for their choice. Some chose sentences based on their potential to move the plot forward or because they fit better with *"the theme"* or *"the tone"* of the context. Others made decisions based on whether they thought a sentence of dialogue or a descriptive sentence was more appropriate, or a statement versus a question. Many made their decisions using deeper knowledge about the story's context. For example, in the second story listed in Table 3, one Turker used social knowledge to choose the S2SA sentence because *"the introduction makes the man sound like he is a stranger, so 'I'm proud of you' seems out of place."* In this case, even though the sentence from ENGEN correctly generated pronouns that refer to entities in the context, the mismatch in the social aspects of the context and ENGEN's sentence contributed to 7 out of 11 Turkers choosing the vaguer S2SA sentence.

While neither S2SA nor ENGEN explicitly encodes these types of information, these qualities are important to human evaluators of generated text and should influence future work on narrative text generation.

[6]We included the whole sentence that contained the 60th word, so most documents were slightly over 60 words.

Context	ENGEN	S2SA	#
he says that it was supposed to look random , but he feels it was planned . i was the target . he 's not sure , but he feels that you might have something to do with this , " cassey said sadly . " he ca n't do that ! " manny yelled . " he ca n't accuse me with no justification .	it 's not me . "	he has nothing to do with my life	10
he was wearing brown slacks and a tan button-down shirt , with wool slippers . he looked about sixty , a little paunchy , with balding brown hair and a bushy mustache . ice blue eyes observed alejo keenly , then drifted over to wara ." welcome to my home . " the man 's voice was deep and calm .	" i 'm proud of you , " he said .	" what 's going on ? '	4
bearl looked on the scene , and gasped . this was the white rock of legend , the rock that had lured him to this land . then he stopped . " look , geron . the white rock we saw from the sea . " the struggle was taking place on the white rock . the monster had his back to bearl .	" oh my god ! "	he could not believe his eyes	1

Table 3: Example generated sentences, for three different contexts. The last column indicates the number of Turkers who voted for ENGEN's sentence (out of 11). While entity mentions appear in most of the generated texts, correct entity mentions are not sufficient to guarantee a win, as seen in the second example.

8 Related Work

Beyond past work already discussed, we note a few additional important areas of research relevant to our work.

Neural models for text generation Natural language generation is a classic problem in artificial intelligence. Recent use of RNNs (Sutskever et al., 2011) has reignited interest in this area. Our work provides an additional way to address the well-known drawback of RNNs: they use only limited context. This has been noted as a serious problem in conversational modeling (Sordoni et al., 2015) and text generation with multiple sentences (Lau et al., 2017). Recent work on context-aware text generation (or the related task, language modeling) has studied the possibilities of using different granularity of context. For example, in the scenario of response generation, Sordoni et al. (2015) showed a consistent gain by including one more utterance from context. Similar effects are also observed by adding topical information for language modeling and generation (Lau et al., 2017).

Entity-related generation Choosing an appropriate entity and its mention has a big influence on the coherence of a text, as studied in Centering Theory (Grosz et al., 1995). Recently, the ENTITYNLM proposed by Ji et al. (2017) shows that adding entity related information can improve the performance of language modeling, which potentially provides a method for entity related text generation. We build on ENTITYNLM, combining entity context with previous-sentence context, and demonstrate the importance of the latter in a coherence test (§6). The max pooling combination we propose is simple but effective.

Another line of related work on recipe generation included special treatment of entities as candidates in generating sentences, but not as context (Kiddon et al., 2016). Bosselut et al. (2018) also generated recipes, using neural process networks to track and update entity representations with the goal of modeling actions and their causal effects on entities. However, the entity representations are frozen during generation, rather than dynamically updated.

Mention generation Our novel mention generation task is inspired by both referring expression generation (Dale and Reiter, 1995) and entity prediction (Modi et al., 2017). The major difference is that, unlike referring expression generation, our

task includes all the mentions used for entities, including pronouns; we believe it is a more realistic test of a model's handling of entities. Krahmer and Van Deemter (2012) give a comprehensive survey on early work of referring expression generation.

The mention only version of the mention generation task is related to cloze tests like the Children's Book Test (Hill et al., 2016), the "Who-did-What" Test (Onishi et al., 2016), and the CNN and Daily Mail test described by Hermann et al. (2015). However, unlike these tests, we predict all entity mentions in the text and from a dynamically expanding candidate list, typically much larger than those in other cloze tests.

Story generation Work in story generation has incorporated structure and context through event representations (Martin et al., 2017) or semantic representations, like story graphs (Rishes et al., 2013; Elson and McKeown, 2009). In this work, we provide evidence for the value of entity representations as an additional form of structure, following work by Walker et al. (2011), Cavazza and Charles (2005), and Cavazza et al. (2002).

9 Conclusion

Inspired by Centering Theory and the importance of characters in stories, we propose a neural model for text generation that incorporates context via entities. We found that combining entity representations with representations of the previous sentence and the hidden state (from a neural language model) improves performance on three tasks: mention generation, sentence selection, and sentence generation. By collecting human evaluations of sentences generated with entity information, we find that while coherently referring back to entities in the context was cited by several Turkers as a factor in their decision, the introduction of new entities and moving the narrative forward were also valued.

Therefore, while entities are a useful structure to incorporate in story generation, other structures may also prove useful, including other aspects of discourse (e.g., discourse relations or planning) or story-related structures (e.g., narrative structure).

Acknowledgments

This research was supported in part by a NSF graduate research fellowship, the DARPA CwC program through ARO (W911NF-15-1-0543), and the NVIDIA Corporation with the donation of the Tesla GPU used for this research. The authors also thank Maarten Sap for his feedback and helpful suggestions, the anonymous reviewers for their useful comments, and the participants who took part in our study.

References

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

Paul Baltescu and Phil Blunsom. 2015. Pragmatic neural language modelling in machine translation. In *NAACL*. https://doi.org/10.3115/v1/n15-1083.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics* 34(1):1–34. https://doi.org/10.1162/coli.2008.34.1.1.

Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. 2018. Simulating action dynamics with neural process networks. In *ICLR*.

Peter F. Brown, Vincent J. Della Pietra, Peter V. de Souza, Jennifer C. Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics* 18:467–479.

Marc Cavazza and Fred Charles. 2005. Dialogue generation in character-based interactive storytelling. In *AIIDE*.

Marc Cavazza, Fred Charles, and Steven J. Mead. 2002. Character-based interactive storytelling. *IEEE Intelligent Systems* 17:17–24. https://doi.org/10.1109/mis.2002.1024747.

Kevin Clark and Christopher D. Manning. 2016a. Deep reinforcement learning for mention-ranking coreference models. In *EMNLP*. https://doi.org/10.18653/v1/d16-1245.

Kevin Clark and Christopher D. Manning. 2016b. Improving coreference resolution by learning entity-level distributed representations. In *ACL*. https://doi.org/10.18653/v1/p16-1061.

Robert Dale and Ehud Reiter. 1995. Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive Science* 19(2):233–263. https://doi.org/10.1207/s15516709cog1902_3.

David K. Elson and Kathleen McKeown. 2009. A tool for deep semantic encoding of narrative texts. In *ACL-IJCNLP*. https://doi.org/10.3115/1667872.1667875.

Pablo Gervás. 2009. Computational approaches to storytelling and creativity. *AI Magazine* 30:49–62. https://doi.org/10.1609/aimag.v30i3.2250.

Shalini Ghosh, Oriol Vinyals, Brian Strope, Scott Roy, Tom Dean, and Larry Heck. 2016. Contextual LSTM (CLSTM) models for large scale NLP tasks. arXiv:1602.06291.

Joshua Goodman. 2001. Classes for fast maximum entropy training. In *ICASSP*. https://doi.org/10.1109/icassp.2001.940893.

Barbara J. Grosz, Scott Weinstein, and Aravind K. Joshi. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics* 21(2):203–225. https://doi.org/10.21236/ada324949.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The Goldilocks principle: Reading children's books with explicit memory representations. In *ICLR*.

Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A. Smith. 2017. Dynamic entity representations in neural language models. In *EMNLP*. https://doi.org/10.18653/v1/d17-1195.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *ACL*. https://doi.org/10.3115/v1/p14-1062.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *EMNLP*. https://doi.org/10.18653/v1/d16-1032.

Emiel Krahmer and Kees Van Deemter. 2012. Computational generation of referring expressions: A survey. *Computational Linguistics* 38(1):173–218. https://doi.org/10.1162/coli_a_00088.

Jey Han Lau, Timothy Baldwin, and Trevor Cohn. 2017. Topically driven neural language model. In *ACL*. https://doi.org/10.18653/v1/p17-1033.

Percy Liang. 2005. *Semi-supervised learning for natural language*. Ph.D. thesis, Massachusetts Institute of Technology.

Stephanie M. Lukin, Lena Reed, and Marilyn A. Walker. 2015. Generating sentence planning variations for story telling. In *SIGDIAL*. https://doi.org/10.18653/v1/w15-4627.

Lara J. Martin, Prithviraj Ammanabrolu, William Hancock, Shruti Singh, Brent Harrison, and Mark O. Riedl. 2017. Event representations for automated story generation with deep neural nets. arXiv:1706.01331.

Ashutosh Modi, Ivan Titov, Vera Demberg, Asad B. Sayeed, and Manfred Pinkal. 2017. Modeling semantic expectation: Using script knowledge for referent prediction. *TACL* 5:31–44.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. DyNet: The dynamic neural network toolkit. arXiv:1701.03980.

Takeshi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel, and David A. McAllester. 2016. Who did what: A large-scale person-centered cloze dataset. In *EMNLP*. https://doi.org/10.18653/v1/d16-1241.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *EMNLP-CoNLL*.

Elena Rishes, Stephanie M. Lukin, David K. Elson, and Marilyn A. Walker. 2013. Generating different story tellings from semantic representations of narrative. In *ICIDS*.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Meg Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *NAACL*. https://doi.org/10.3115/v1/n15-1020.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *ICML*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Marilyn A. Walker, Ricky Grant, Jennifer Sawyer, Grace I. Lin, Noah Wardrip-Fruin, and Michael Buell. 2011. Perceived or not perceived: Film character models for expressive NLG. In *ICIDS*. https://doi.org/10.1007/978-3-642-25289-1_12.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *ICCV*. https://doi.org/10.1109/iccv.2015.11.

Recurrent Neural Networks as Weighted Language Recognizers

Yining Chen
Dartmouth College
yining.chen.18@dartmouth.edu

Sorcha Gilroy
ILCC
University of Edinburgh
s.gilroy@sms.ed.ac.uk

Andreas Maletti
Institute of Computer Science
Universität Leipzig
andreas.maletti@uni-leipzig.de

Jonathan May
Information Sciences Institute
University of Southern California
jonmay@isi.edu

Kevin Knight
Information Sciences Institute
University of Southern California
knight@isi.edu

Abstract

We investigate the computational complexity of various problems for simple recurrent neural networks (RNNs) as formal models for recognizing weighted languages. We focus on the single-layer, ReLU-activation, rational-weight RNNs with softmax, which are commonly used in natural language processing applications. We show that most problems for such RNNs are undecidable, including consistency, equivalence, minimization, and the determination of the highest-weighted string. However, for consistent RNNs the last problem becomes decidable, although the solution length can surpass all computable bounds. If additionally the string is limited to polynomial length, the problem becomes NP-complete. In summary, this shows that approximations and heuristic algorithms are necessary in practical applications of those RNNs.

1 Introduction

Recurrent neural networks (RNNs) are an attractive apparatus for probabilistic language modeling (Mikolov and Zweig, 2012). Recent experiments show that RNNs significantly outperform other methods in assigning high probability to held-out English text (Jozefowicz et al., 2016).

Roughly speaking, an RNN works as follows. At each time step, it consumes one input token, updates its hidden state vector, and predicts the next token by generating a probability distribution over all permissible tokens. The probability of an input string is simply obtained as the product of the predictions of the tokens constituting the string followed by a terminating token. In this manner, each RNN defines a *weighted language*; i.e. a total function from strings to weights. Siegelmann and Sontag (1995) showed that single-layer rational-weight RNNs with saturated linear activation can compute any computable function. To this end, a specific architecture with 886 hidden units can simulate any Turing machine in real-time (i.e., each Turing machine step is simulated in a single time step). However, their RNN encodes the whole input in its internal state, performs the actual computation of the Turing machine when reading the terminating token, and then encodes the output (provided an output is produced) in a particular hidden unit. In this way, their RNN allows "thinking" time (equivalent to the computation time of the Turing machine) after the input has been encoded.

We consider a different variant of RNNs that is commonly used in natural language processing applications. It uses ReLU activations, consumes an input token at each time step, and produces softmax predictions for the next token. It thus immediately halts after reading the last input token and the weight assigned to the input is simply the product of the input token predictions in each step.

Other formal models that are currently used to implement probabilistic language models such as finite-state automata and context-free grammars are by now well-understood. A fair share of their utility directly derives from their nice algorithmic properties. For example, the weighted languages computed by weighted finite-state automata are closed under intersection (pointwise product) and union (pointwise sum), and the corresponding unweighted languages are closed under intersection, union, difference, and complementation (Droste et al., 2013). Moreover, toolkits like OpenFST (Allauzen et al., 2007) and Carmel[1] implement efficient algorithms on automata like minimization, intersection, finding the highest-weighted path and the highest-weighted string.

RNN practitioners naturally face many of these same problems. For example, an RNN-

[1] https://www.isi.edu/licensed-sw/carmel/

Proceedings of NAACL-HLT 2018, pages 2261–2271
New Orleans, Louisiana, June 1 - 6, 2018.

based machine translation system should extract the highest-weighted output string (i.e., the most likely translation) generated by an RNN, (Sutskever et al., 2014; Bahdanau et al., 2014). Currently this task is solved by approximation techniques like heuristic greedy and beam searches. To facilitate the deployment of large RNNs onto limited memory devices (like mobile phones) minimization techniques would be beneficial. Again currently only heuristic approaches like knowledge distillation (Kim and Rush, 2016) are available. Meanwhile, it is unclear whether we can determine if the computed weighted language is consistent; i.e., if it is a probability distribution on the set of all strings. Without a determination of the overall probability mass assigned to all finite strings, a fair comparison of language models with regard to perplexity is simply impossible.

The goal of this paper is to study the above problems for the mentioned ReLU-variant of RNNs. More specifically, we ask and answer the following questions:

- Consistency: Do RNNs compute consistent weighted languages? Is the consistency of the computed weighted language decidable?
- Highest-weighted string: Can we (efficiently) determine the highest-weighted string in a computed weighted language?
- Equivalence: Can we decide whether two given RNNs compute the same weighted language?
- Minimization: Can we minimize the number of neurons for a given RNN?

2 Definitions and notations

Before we introduce our RNN model formally, we recall some basic notions and notation. An *alphabet* Σ is a finite set of symbols, and we write $|\Sigma|$ for the number of symbols in Σ. A *string* s over the alphabet Σ is a finite sequence of zero or more symbols drawn from Σ, and we write Σ^* for the set of all strings over Σ, of which ϵ is the empty string. The length of the string $s \in \Sigma^*$ is denoted $|s|$ and coincides with the number of symbols constituting the string. As usual, we write A^B for the set of functions $\{f \mid f\colon B \to A\}$. A *weighted language* L is a total function $L\colon \Sigma^* \to \mathbb{R}$ from strings to real-valued weights. For example, $L(a^n) = e^{-n}$ for all $n \geq 0$ is such a weighted language.

We restrict the weights in our RNNs to the rational numbers $\mathbb{Q}$. In addition, we reserve the use of a special symbol \$ to mark the start and end of an input string. To this end, we assume that $\$ \notin \Sigma$ for all considered alphabets, and we let $\Sigma_\$ = \Sigma \cup \{\$\}$.

Definition 1. *A single-layer RNN R is a 7-tuple $\langle \Sigma, N, h_{-1}, W, W', E, E' \rangle$, in which*

- *Σ is an* input alphabet,
- *N is a finite set of* neurons,
- *$h_{-1} \in \mathbb{Q}^N$ is an* initial activation vector,
- *$W \in \mathbb{Q}^{N \times N}$ is a* transition matrix,
- *$W' = (W'_a)_{a \in \Sigma_\$}$ is a $\Sigma_\$$-indexed family of* bias vectors *$W'_a \in \mathbb{Q}^N$,*
- *$E \in \mathbb{Q}^{\Sigma_\$ \times N}$ is a* prediction matrix, *and*
- *$E' \in \mathbb{Q}^{\Sigma_\$}$ is a* prediction bias vector.

Next, let us define how such an RNN works. We first prepare our input encoding and the effect of our activation function. For an input string $s = s_1 s_2 \cdots s_n \in \Sigma^*$ with $s_1, \ldots, s_n \in \Sigma$, we encode this input as $\$s\$$ and thus assume that $s_0 = \$$ and $s_{n+1} = \$$. Our RNNs use ReLUs (Rectified Linear Units), so for every $v \in \mathbb{Q}^N$ we let $\sigma\langle v \rangle$ (the ReLU activation) be the vector $\sigma\langle v \rangle \in \mathbb{Q}^N$ such that

$$\sigma\langle v \rangle(n) = \max\big(0, v(n)\big) \qquad \text{for every } n \in N.$$

In other words, the ReLUs act like identities on nonnegative inputs, but clip negative inputs to 0. We use softmax-predictions, so for every vector $p \in \mathbb{Q}^{\Sigma_\$}$ and $a \in \Sigma_\$$ we let

$$\operatorname{softmax}\langle p \rangle(a) = \frac{e^{p(a)}}{\sum_{a' \in \Sigma_\$} e^{p(a')}} \ .$$

RNNs act in discrete time steps reading a single letter at each step. We now define the semantics of our RNNs.

Definition 2. *Let $R = \langle \Sigma, N, h_{-1}, W, W', E, E' \rangle$ be an RNN, s an input string of length n and $0 \leq t \leq n$ a time step. We define*

- *the* hidden state vector $h_{s,t} \in \mathbb{Q}^N$ *given by*

$$h_{s,t} = \sigma\langle W \cdot h_{s,t-1} + W'_{s_t} \rangle \ ,$$

  where $h_{s,-1} = h_{-1}$ and we use standard matrix product and point-wise vector addition,
- *the* next-token prediction vector $E_{s,t} \in \mathbb{Q}^{\Sigma_\$}$

$$E_{s,t} = E \cdot h_{s,t} + E'$$

- *the* next-token distribution $E'_{s,t} \in \mathbb{R}^{\Sigma_\$}$

$$E'_{s,t} = \operatorname{softmax}\langle E_{s,t} \rangle \ .$$

Finally, the RNN R computes the weighted language $R\colon \Sigma^* \to \mathbb{R}$, which is given for every input $s = s_1 \cdots s_n$ as above by

$$R(s) = \prod_{t=0}^{n} E'_{s,t}(s_{t+1}) \ .$$

In other words, each component $h_{s,t}(n)$ of the hidden state vector is the ReLU activation applied to a linear combination of all the components of the previous hidden state vector $h_{s,t-1}$ together with a summand W'_{s_t} that depends on the t-th input letter s_t. Thus, we often specify $h_{s,t}(n)$ as linear combination instead of specifying the matrix W and the vectors W'_a. The semantics is then obtained by predicting the letters $s_1, \ldots, s_n$ of the input s and the final terminator \$ and multiplying the probabilities of the individual predictions.

Let us illustrate these notions on an example. We consider the RNN $\langle \Sigma, N, h_{-1}, W, W', E, E' \rangle$ with $\gamma \in \mathbb{Q}$ and

- $\Sigma = \{a\}$ and $N = \{1, 2\}$,
- $h_{-1} = (-1, 0)^T$ and

$$W = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \quad \text{and} \quad W'_{\$} = W'_a = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

- $E(\$, \cdot) = (M + 1, -(M + 1))$ and $E(a, \cdot) = (1, -1)$ and
- $E'(\$) = -M$ and $E'(a) = 0$.

In this case, we obtain the linear combinations

$$h_{s,t} = \sigma \left\langle \begin{matrix} h_{s,t-1}(1) + 1 \\ h_{s,t-1}(1) \end{matrix} \right\rangle$$

computing the next hidden state components. Given the initial activation, we thus obtain $h_{s,t} = \sigma\langle t, t-1 \rangle$. Using this information, we obtain

$$\begin{aligned} E_{s,t}(\$) &= (M+1) \cdot (t - \sigma\langle t-1 \rangle) - M \\ E_{s,t}(a) &= t - \sigma\langle t-1 \rangle \ . \end{aligned}$$

Consequently, we assign weight $\frac{e^{-M}}{1+e^{-M}}$ to input ε, weight $\frac{1}{1+e^{-M}} \cdot \frac{e^1}{e^1+e^1}$ to a, and, more generally, weight $\frac{1}{1+e^{-M}} \cdot \frac{1}{2^n}$ to a^n.

Clearly the weight assigned by an RNN is always in the interval $(0, 1)$, which enables a probabilistic view. Similar to weighted finite-state automata or weighted context-free grammars, each RNN is a compact, finite representation of a weighted language. The softmax-operation enforces that the probability 0 is impossible as assigned weight, so each input string is principally possible. In practical language modeling, smoothing methods are used to change distributions such that impossibility (probability 0) is removed. Our RNNs avoid impossibility outright, so this can be considered a feature instead of a disadvantage.

The hidden state $h_{s,t}$ of an RNN can be used as scratch space for computation. For example, with a single neuron n we can count symbols in s via:

$$h_{s,t}(n) = \sigma\langle h_{s,t-1}(n) + 1 \rangle \ .$$

Here the letter-dependent summand W'_a is universally 1. Similarly, for an alphabet $\Sigma = \{a_1, \ldots, a_m\}$ we can use the method of Siegelmann and Sontag (1995) to encode the complete input string s in base $m + 1$ using:

$$h_{s,t}(n) = \sigma\langle (m+1) h_{s,t-1}(n) + c(s_t) \rangle \ ,$$

where $c\colon \Sigma_{\$} \to \{0, \ldots, m\}$ is a bijection. In principle, we can thus store the entire input string (of unbounded length) in the hidden state value $h_{s,t}(n)$, but our RNN model outputs weights at each step and terminates immediately once the final delimiter \$ is read. It must assign a probability to a string *incrementally* using the chain rule decomposition $p(s_1 \cdots s_n) = p(s_1) \cdot \ldots \cdot p(s_n \mid s_1 \cdots s_{n-1})$.

Let us illustrate our notion of RNNs on some additional examples. They all use the alphabet $\Sigma = \{a\}$ and are illustrated and formally specified in Figure 1. The first column shows an RNN R_1 that assigns $R_1(a^n) = 2^{-(n+1)}$. The next-token prediction matrix ensures equal values for a and \$ at every time step. The second column shows the RNN R_2, which we already discussed. In the beginning, it heavily biases the next symbol prediction towards a, but counters it starting at $t = 1$. The third RNN R_3 uses another counting mechanism with $h_{s,t} = \sigma\langle t - 100, t - 101, t \rangle$. The first two components are ReLU-thresholded to zero until $t > 101$, at which point they overwhelm the bias towards a turning all future predictions to \$.

3 Consistency

We first investigate the consistency problem for an RNN R, which asks whether the recognized weighted language R is indeed a probability distribution. Consequently, an RNN R is *consistent*

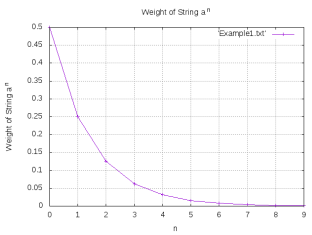
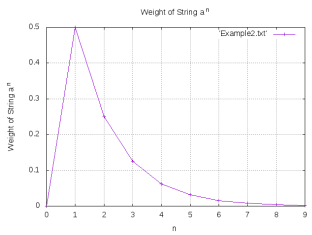
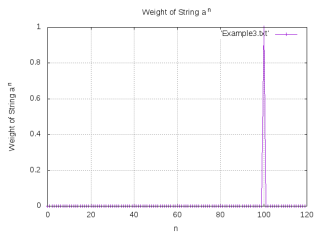
	$R_1(a^n) = 2^{-(n+1)}$	$R_2(\varepsilon) \approx 0$ $R_2(a^n) \approx 2^{-n}$ $(n \geq 1)$	$R_3(a^{100}) \approx 1$ $R_3(a^n) \approx 0$ $(n \neq 100)$
N	$\{1\}$	$\{1, 2\}$	$\{1, 2, 3\}$
h_{-1}	$\begin{pmatrix} 0 \end{pmatrix}$	$\begin{pmatrix} -1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$
W	$\begin{pmatrix} 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$
$W'_{\$}$ W'_a	$\begin{pmatrix} 0 \end{pmatrix}$ $\begin{pmatrix} 0 \end{pmatrix}$	$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} -99 \\ -100 \\ 1 \end{pmatrix}$ $\begin{pmatrix} -99 \\ -100 \\ 1 \end{pmatrix}$
$E_{\$}$ E_a	$\begin{pmatrix} 0 \end{pmatrix}$ $\begin{pmatrix} 0 \end{pmatrix}$	$\begin{pmatrix} M+1 \\ -(M+1) \end{pmatrix}$ $\begin{pmatrix} 1 \\ -1 \end{pmatrix}$	$\begin{pmatrix} M \\ -M \\ 0 \end{pmatrix}$ $\begin{pmatrix} -M \\ M \\ 0 \end{pmatrix}$
$E'_{\$}$ E'_a	0 0	$-M$ 0	$-M$ 0
			

Figure 1: Sample RNNs over single-letter alphabets, and the weighted languages they recognize. M is some positive rational number which depends on the desired error margin. If we want to express the second and the third languages with error margin δ, M is chosen so that $M > -\ln \frac{\delta}{1-\delta}$ in column 2, and chosen so that $(1 + e^{-M})^{100} < \frac{1}{1-\delta}$ in column 3.

if $\sum_{s \in \Sigma^*} R(s) = 1$. We first show that there is an inconsistent RNN, which together with our examples shows that consistency is a nontrivial property of RNNs.[2]

We immediately use a slightly more complex example, which we will later reuse.

Example 3. Let us consider an arbitrary RNN

$$R = \langle \Sigma, N, h_{-1}, W, W', E, E' \rangle$$

with the single-letter alphabet $\Sigma = \{a\}$, the neurons $\{1, 2, 3, n, n'\} \subseteq N$, initial activation $h_{-1}(i) = 0$ for all $i \in \{1, 2, 3, n, n'\}$, and the following linear combinations:

$$h_{s,t}(1) = \sigma\langle h_{s,t-1}(1) + h_{s,t-1}(n) - h_{s,t-1}(n') \rangle$$
$$h_{s,t}(2) = \sigma\langle h_{s,t-1}(2) + 1 \rangle$$
$$h_{s,t}(3) = \sigma\langle h_{s,t-1}(3) + 3h_{s,t-1}(1) \rangle$$
$$E_{s,t}(\$) = h_{s,t}(3) - h_{s,t}(2)$$
$$E_{s,t}(a) = h_{s,t}(2)$$

Now we distinguish two cases:
Case 1: If $h_{s,t}(n) - h_{s,t}(n') = 0$ for all $t \in \mathbb{N}$, then $h_{s,t}(1) = 0$ and $h_{s,t}(2) = t + 1$ and $h_{s,t}(3) = 0$. Hence we have $E_{s,t}(\$) = -(t + 1)$ and $E_{s,t}(a) = t + 1$. In this case the termination probability

$$E'_{s,t}(\$) = \frac{e^{-(t+1)}}{e^{-(t+1)} + e^{t+1}} = \frac{1}{1 + e^{2(t+1)}}$$

(i.e., the likelihood of predicting \$) shrinks rapidly towards 0, so the RNN assigns less than 15% of the probability mass to the terminating sequences (i.e., the finite strings), so the RNN is inconsistent (see Lemma 15 in the appendix).

Case 2: Suppose that there exists a time

[2] For comparison, all probabilistic finite-state automata are consistent, provided no transitions exit final states. Not all probabilistic context-free grammars are consistent; necessary and sufficient conditions for consistency are given by Booth and Thompson (1973). However, probabilistic context-free grammars obtained by training on a finite corpus using popular methods (such as expectation-maximization) are guaranteed to be consistent (Nederhof and Satta, 2006).

point $T \in \mathbb{N}$ such that for all $t \in \mathbb{N}$

$$h_{s,t}(n) - h_{s,t}(n') = \begin{cases} 1 & \text{if } t = T \\ 0 & \text{otherwise.} \end{cases}$$

Then $h_{s,t}(1) = 0$ for all $t \leq T$ and $h_{s,t}(1) = 1$ otherwise. In addition, we have $h_{s,t}(2) = t + 1$ and $h_{s,t}(3) = \sigma\langle 3(t - T - 1)\rangle$. Hence we have

$$\begin{aligned} E_{s,t}(\$) &= \sigma\langle 3(t - T - 1)\rangle - (t + 1) \\ &= \begin{cases} -(t+1) & \text{if } t \leq T \\ 2t - 3T - 4 & \text{otherwise} \end{cases} \\ E_{s,t}(a) &= t + 1 \ , \end{aligned}$$

which shows that the probability

$$E'_{s,t}(\$) = \begin{cases} \frac{1}{1+e^{2(t+1)}} & \text{if } t \leq T \\ \frac{e^{t-3T-5}}{1+e^{t-3T-5}} & \text{otherwise} \end{cases}$$

of predicting \$ increases over time and eventually (for $t \gg 3T$) far outweighs the probability of predicting a. Consequently, in this case the RNN is consistent (see Lemma 16 in the appendix).

We have seen in the previous example that consistency is not trivial for RNNs, which takes us to the consistency problem for RNNs:

Consistency: Given an RNN R, return "yes" if R is consistent and "no" otherwise.

We recall the following theorem, which, combined with our example, will prove that consistency is unfortunately undecidable for RNNs.

Theorem 4 (Theorem 2 of Siegelmann and Sontag (1995))**.** *Let M be an arbitrary deterministic Turing machine. There exists an RNN*

$$R = \langle \Sigma, N, h_{-1}, W, W', E, E' \rangle$$

with saturated linear activation, input alphabet $\Sigma = \{a\}$, and 1 designated neuron $n \in N$ such that for all $s \in \Sigma^*$ and $0 \leq t \leq |s|$

- *$h_{s,t}(n) = 0$ if M does not halt on ε, and*
- *if M does halt on empty input after T steps, then*

$$h_{s,t}(n) = \begin{cases} 1 & \text{if } t = T \\ 0 & \text{otherwise.} \end{cases}$$

In other words, such RNNs with saturated linear activation can semi-decide halting of an arbitrary Turing machine in the sense that a particular neuron achieves value 1 at some point during the evolution if and only if the Turing machine halts on empty input. An RNN with saturated linear activation is an RNN following our definition with the only difference that instead of our ReLU-activation σ the following saturated linear activation $\sigma' \colon \mathbb{Q}^N \to \mathbb{Q}^N$ is used. For every vector $v \in \mathbb{Q}^N$ and $n \in N$, let

$$\sigma'\langle v \rangle(n) = \begin{cases} 0 & \text{if } v(n) < 0 \\ v(n) & \text{if } 0 \leq v(n) \leq 1 \\ 1 & \text{if } v(n) > 1 \ . \end{cases}$$

Since $\sigma'\langle v \rangle = \sigma\langle v \rangle - \sigma\langle v - \vec{1} \rangle$ for all $v \in \mathbb{Q}^N$, and the right-hand side is a linear transformation, we can easily simulate saturated linear activation in our RNNs. To this end, each neuron $n \in N$ of the original RNN $R = \langle \Sigma, N, h_{-1}, U, U', E, E' \rangle$ is replaced by two neurons n_1 and n_2 in the new RNN $R' = \langle \Sigma, N', h'_{-1}, V, V', F, F' \rangle$ such that $h_{s,t}(n) = h'_{s,t}(n_1) - h'_{s,t}(n_2)$ for all $s \in \Sigma^*$ and $0 \leq t \leq |s|$, where the evaluation of $h'_{s,t}$ is performed in the RNN R'. More precisely, we use the transition matrix V and bias function V', which is given by

$$\begin{aligned} V(n_1, n'_1) &= V(n_2, n'_1) = U(n, n') \\ V(n_1, n'_2) &= V(n_2, n'_2) = -U(n, n') \\ V'_a(n_1) &= U'_a(n) \\ V'_a(n_2) &= U'_a(n) - 1 \\ h'_{-1}(n_1) &= h_{-1}(n) \\ h'_{-1}(n_2) &= 0 \end{aligned}$$

for all $n, n' \in N$ and $a \in \Sigma \cup \{\$\}$, where n_1 and n_2 are the two neurons corresponding to n and n'_1 and n'_2 are the two neurons corresponding to n' (see Lemma 17 in the appendix).

Corollary 5. *Let M be an arbitrary deterministic Turing machine. There exists an RNN*

$$R = \langle \Sigma, N, h_{-1}, W, W', E, E' \rangle$$

with input alphabet $\Sigma = \{a\}$ and 2 designated neurons $n_1, n_2 \in N$ such that for all $s \in \Sigma^*$ and $0 \leq t \leq |s|$

- *$h_{s,t}(n_1) - h_{s,t}(n_2) = 0$ if M does not halt on ε, and*
- *if M does halt on empty input after T steps, then*

$$h_{s,t}(n_1) - h_{s,t}(n_2) = \begin{cases} 1 & \text{if } t = T \\ 0 & \text{otherwise.} \end{cases}$$

We can now use this corollary together with the RNN R of Example 3 to show that the consistency problem is undecidable. To this end, we simulate a given Turing machine M and identify the two designated neurons of Corollary 5 as n and n' in Example 3. It follows that M halts if and only if R is consistent. Hence we reduced the undecidable halting problem to the consistency problem, which shows the undecidability of the consistency problem.

Theorem 6. *The consistency problem for RNNs is undecidable.*

As mentioned in Footnote 2, probabilistic context-free grammars obtained after training on a finite corpus using the most popular methods are guaranteed to be consistent. At least for 2-layer RNNs this does not hold.

Theorem 7. *A two-layer RNN trained to a local optimum using Back-propagation-through-time (BPTT) on a finite corpus is not necessarily consistent.*

Proof. The first layer of the RNN R with a single alphabet symbol a uses one neuron n' and has the following behavior:

$$h_{-1}(n') = 0$$
$$h_{s,t}(n') = \sigma\langle h_{s,t-1}(n') + 1\rangle$$

The second layer uses neuron n and takes $h_{s,t}(n')$ as input at time t:

$$h_{s,t}(n) = \sigma\langle h_{s,t}(n') - 2\rangle$$
$$E_{s,t}(a) = h_{s,t}(n) \qquad E_{s,t}(\$) = 0$$
$$E'_{s,t}(a) = \begin{cases} \frac{1}{2} & \text{if } t \leq 1 \\ \frac{e^{(t-1)}}{1+e^{(t-1)}} & \text{otherwise.} \end{cases}$$

Let the training data be $\{a\}$. Then the objective we wish to maximize is simply $R(a)$. The derivative of this objective with respect to each parameter is 0, so applying gradient descent updates does not change any of the parameters and we have converged to an inconsistent RNN. □

It remains an open question whether there is a single-layer RNN that also exhibits this behavior.

4 Highest-weighted string

Given a function $f\colon \Sigma^* \to \mathbb{R}$ we are often interested in the highest-weighted string. This corresponds to the most likely sentence in a language model or the most likely translation for a decoder RNN in machine translation.

	Best-path	Best-string
General RNN	**Undecidable**	
Consistent RNN	**NP-c** [3]	
Det. PFSA/PCFG	P [4]	
Nondet. PFSA/PCFG		NP-c [5]

Table 1: Comparison of the difficulty of identifying the most probable derivation (Best-path) and the highest-weighted string (Best-string) for various models.

For deterministic probabilistic finite-state automata or context-free grammars only one path or derivation exists for any given string, so the identification of the highest-weighted string is the same task as the identification of the most probable path or derivation. However, for nondeterministic devices, the highest-weighted string is often harder to identify, since the weight of a string is the sum of the probabilities of all possible paths or derivations for that string. A comparison of the difficulty of identifying the most probable derivation and the highest-weighted string for various models is summarized in Table 1, in which we marked our results in bold face.

We present various results concerning the difficulty of identifying the highest-weighted string in a weighted language computed by an RNN. We also summarize some available algorithms. We start with the formal presentation of the three studied problems.

1. **Best string:** Given an RNN R and $c \in (0, 1)$, does there exist $s \in \Sigma^*$ with $R(s) > c$?
2. **Consistent best string:** Given a consistent RNN R and $c \in (0, 1)$, does there exist $s \in \Sigma^*$ with $R(s) > c$?
3. **Consistent best string of polynomial length:** Given a consistent RNN R, polynomial $\mathcal{P}$ with $\mathcal{P}(x) \geq x$ for $x \in \mathbb{N}^+$, and $c \in (0, 1)$, does there exist $s \in \Sigma^*$ with $|s| \leq \mathcal{P}(|R|)$ and $R(s) > c$?

As usual the corresponding optimization problems are not significantly simpler than these decision problems. Unfortunately, the general problem is also undecidable, which can easily be shown using our example.

[3] Restricted to solutions of polynomial length
[4] Dijkstra shortest path / (Knuth, 1977)
[5] (Casacuberta and de la Higuera, 2000) / (Simaan, 1996)

Theorem 8. *The best string problem for RNNs is undecidable.*

Proof. Let M be an arbitrary Turing machine and again consider the RNN R of Example 3 with the neurons n and n' identified with the designated neurons of Corollary 5. We note that $R(\varepsilon) = \frac{1}{1+e^2} < 0.12$ in both cases. If M does not halt, then $R(a^n) \leq \frac{1}{1+e^{2(n+1)}} \leq \frac{1}{1+e^2} < 0.12$ for all $n \in \mathbb{N}$. On the other hand, if M halts after T steps, then

$$\begin{aligned} & R(a^{3T-5}) \\ &= \Big(\prod_{t=0}^{T} \frac{e^{2(t+1)}}{1+e^{2(t+1)}}\Big) \cdot \Big(\prod_{t=T+1}^{3T-6} \frac{1}{1+e^{t-3T-5}}\Big) \cdot \frac{1}{2} \\ &\geq \frac{2}{(-1,e^{-2})_\infty} \cdot \Big(\prod_{t=T+1}^{3T-6} \frac{e^{3T+5-t}}{e^{3T+5-t+1}}\Big) \cdot \frac{1}{2} \\ &\geq \frac{2}{(-1,e^{-2})_\infty \cdot (-1,e^{-1})_\infty} \geq 0.25 \end{aligned}$$

using Lemma 14 in the appendix. Consequently, a string with weight above 0.12 exists if and only if M halts, so the best string problem is also undecidable. □

If we restrict the RNNs to be consistent, then we can easily decide the best string problem by simple enumeration.

Theorem 9. *The consistent best string problem for RNNs is decidable.*

Proof. Let R be the RNN over alphabet Σ and $c \in (0,1)$ be the bound. Since Σ^* is countable, we can enumerate it via $f\colon \mathbb{N} \to \Sigma^*$. In the algorithm we compute $S_n = \sum_{i=0}^{n} R(f(i))$ for increasing values of n. If we encounter a weight $R(f(n)) > c$, then we stop with answer "yes." Otherwise we continue until $S_n > 1 - c$, at which point we stop with answer "no."

Since R is consistent, $\lim_{i\to\infty} S_i = 1$, so this algorithm is guaranteed to terminate and it obviously decides the problem. □

Next, we investigate the length $|w_R^{\max}|$ of the shortest string $w_R^{\max}$ of maximal weight in the weighted language R generated by a consistent RNN R in terms of its (binary storage) size $|R|$. As already mentioned by Siegelmann and Sontag (1995) and evidenced here, only small precision rational numbers are needed in our constructions, so we assume that $|R| \leq c \cdot |N|^2$ for a (reasonably small) constant c, where N is the set of neurons of R. We show that no computable bound on the length of the best string can exist, so its length can surpass all reasonable bounds.

Theorem 10. *Let $f\colon \mathbb{N}_+ \to \mathbb{N}$ be the function with*

$$f(n) = \max_{\substack{\text{consistent RNN } R \\ |R| \leq n}} |w_R^{max}|$$

for all $n \in \mathbb{N}_+$. There exists no computable function $g\colon \mathbb{N} \to \mathbb{N}$ with $g(n) \geq f(n)$ for all $n \in \mathbb{N}$.

Proof. In the previous section (before Theorem 6) we presented an RNN R_M that simulates an arbitrary (single-track) Turing machine M with n states. By Siegelmann and Sontag (1995) we have $|R_M| \leq c \cdot (4n + 16)$. Moreover, we observed that this RNN R_M is consistent if and only if the Turing machine M halts on empty input. In the proof of Theorem 8 we have additionally seen that the length $|w_R^{\max}|$ of its best string exceeds the number T_M of steps required to halt.

For every $n \in \mathbb{N}$, let $BB(n)$ be the n-th "Busy Beaver" number (Radó, 1962), which is

$$BB(n) = \max_{\substack{\text{normalized } n\text{-state Turing machine } M \text{ with} \\ \text{2 tape symbols that halts on empty input}}} T_M$$

It is well-known that $BB\colon \mathbb{N}_+ \to \mathbb{N}$ cannot be bounded by any computable function. However,

$$\begin{aligned} BB(n) &\leq \max_{\substack{\text{normalized } n\text{-state Turing machine } M \text{ with} \\ \text{and 2 tape symbols that halts on empty input}}} |w_{R_M}^{\max}| \\ &\leq \max_{\substack{\text{consistent RNN } R \\ |R| \leq c\cdot(4n+16)}} |w_R^{\max}| \\ &= f(4nc + 16c) \ , \end{aligned}$$

so f clearly cannot be computable and no computable function g can provide bounds for f. □

Finally, we investigate the difficulty of the best string problem for consistent RNN restricted to solutions of polynomial length.

Theorem 11. *Identifying the best string of polynomial length in a consistent RNN is NP-complete.*

Proof. To show NP-hardness, we reduce from the 3-SAT problem. Let $x_1, \ldots, x_m$ be m Boolean variables and

$$F = \bigwedge_{i=1}^{k} \Big(\ell_{i1} \vee \ell_{i2} \vee \ell_{i3}\Big) \ ,$$

be a formula in conjunctive normal form, where $\ell_{ij} \in \{x_1, \ldots, x_m, \neg x_1, \ldots, \neg x_m\}$. 3-SAT asks whether there is a setting of x_is that makes F true.

We initialize $h_{-1}(n) = 0, \forall\ n \in N = \{x_1, \ldots, x_m, c_1, \ldots, c_k, c'_1, \ldots, c'_k, F, n_1, n_2, n_3, \star\}$. Let $s \in \{0,1\}^*$ be the input string. Denote the value of F when $x_j = s_j$ for all $j \in [m]$ as $F(s)$. Let $t \in \mathbb{N}$ with $t \leq |s|$. Set $h_{s,t}(x_m) = \sigma\langle I(s_t)\rangle$, where $I(0) = I(\$) = 0$ and $I(1) = 1$. This stores the current input symbol in neuron x_m, so $h_{s,t}(x_m) = I(s_t)$. In addition, we let $h_{s,t}(x_j) = \sigma\langle h_{s,t-1}(x_{j+1})\rangle$ for all $j \in [m-1]$. Consequently, for all $j \in [m]$

$$h_{s,t}(x_j) = \begin{cases} I(s_{t-(m-j)}) & \text{if } m-j \leq t \\ 0 & \text{otherwise.} \end{cases}$$

Next, we evaluate the clauses. For each $i \in [k]$, we use two neurons c_i and c'_i such that

$$h_{s,t}(c_i) = \sigma\langle f_{s,t}(\ell_{i1}) + f_{s,t}(\ell_{i2}) + f_{s,t}(\ell_{i3})\rangle$$
$$h_{s,t}(c'_i) = \sigma\langle f_{s,t}(\ell_{i1}) + f_{s,t}(\ell_{i2}) + f_{s,t}(\ell_{i3}) - 1\rangle,$$

where $f_{s,t}(x_m) = I(s_t)$, $f_{s,t}(\neg x_m) = 1 - I(s_t)$, and $\forall j \in [m-1], f_{s,t}(x_j) = h_{s,t-1}(x_{j+1})$, $f_{s,t}(\neg x_j) = 1 - h_{s,t-1}(x_{j+1})$. Note that $h_{s,t}(c_i) - h_{s,t}(c'_i)$ contains the evaluation of the clause $\ell_{i1} \vee \ell_{i2} \vee \ell_{i3}$. Let

$$h_{s,t}(F) = \sigma\Big\langle \sum_{i=1}^{k} \Big(h_{s,t-1}(c_i) - h_{s,t-1}(c'_i)\Big) - k + 1 \Big\rangle,$$

so $h_{s,t}(F) = F(s)$ contains the evaluation of the formula F using the values in neurons $x_1, \ldots, x_m$.

We use three counters n_1, n_2, n_3 to ensure that the only relevant inputs are of length $m+2$:

$$h_{s,t}(n_1) = \sigma\langle h_{s,t-1}(n_3) - (m+2)\rangle$$
$$h_{s,t}(n_2) = \sigma\langle h_{s,t-1}(n_3) - (m+1)\rangle$$
$$h_{s,t}(n_3) = \sigma\langle h_{s,t-1}(n_3) + 1\rangle\ ,$$

which yields $h_{s,t}(n_3) = t + 1$, $h_{s,t}(n_2) = \sigma\langle t - (m+1)\rangle$, and $h_{s,t}(n_1) = \sigma\langle t - (m+2)\rangle$.

Our goal neuron is $\star$, which we set to

$$h_{s,t}(\star) = \sigma\langle h_{s,t-1}(F) - h_{s,t-1}(n_1) + h_{s,t-1}(n_2) - 1\rangle$$

so that

$$h_{s,t}(\star) = \begin{cases} h_{s,t-1}(F) & \text{if } t = m+2 \\ 0 & \text{otherwise,} \end{cases}$$

so $h_{s,t}(\star) = 1$ if and only if $t = m+2$ and $F(s) = 1$.

Let $m' = m + 4$. The output is set as follows:

$$E_{s,t}(0) = E_{s,t}(1) = m'\big(1 - 2h_{s,t}(\star)\big)$$
$$E_{s,t}(\$) = -m'\big(1 - 2h_{s,t}(\star)\big)\ ,$$

This yields $E_{s,t}(0) = E_{s,t}(1) = -E_{s,t}(\$) = -m'$ if $t = m+2$ and $F(s) = 1$, and m' otherwise. For $a \in \{0,1\}$,

$$E'_{s,t}(a) = \begin{cases} \frac{e^{-m'}}{2e^{-m'}+e^{m'}} & \text{if } t = m+2 \text{ and } F(s) = 1 \\ \frac{e^{m'}}{2e^{m'}+e^{-m'}} & \text{otherwise} \end{cases}$$

$$E'_{s,t}(\$) = \begin{cases} \frac{e^{m'}}{2e^{-m'}+e^{m'}} & \text{if } t = m+2 \text{ and } F(s) = 1 \\ \frac{e^{-m'}}{2e^{m'}+e^{-m'}} & \text{otherwise.} \end{cases}$$

Finally, we set the threshold $\xi = 3^{-m'}$. When $|s| \neq m+2$, $s_{m+3} \neq \$$, so the weight of s contains the factor $\frac{e^{-m'}}{2e^{-m'}+e^{m'}} = \frac{1}{2+e^{2m'}}$ and thus is upper-bounded by $\frac{1}{2+e^{2m'}} < \xi$. Hence no input of length different from $m+2$ achieves a weight that exceeds ξ. A string s of length $m+2$ achieves the weight w_s given by

$$w_s = \begin{cases} \frac{e^{m'}}{2e^{-m'}+e^{m'}} \cdot \prod_{i=1}^{m+2} \frac{e^{m'}}{2e^{m'}+e^{-m'}} & \text{if } F(s) = 1 \\ \frac{e^{-m'}}{2e^{m'}+e^{-m'}} \cdot \prod_{i=1}^{m+2} \frac{e^{m'}}{2e^{m'}+e^{-m'}} & \text{otherwise.} \end{cases}$$

When $F(s) = 0$, $w_s < \frac{e^{-m'}}{2e^{m'}+e^{-m'}} < \xi$, so if F is unsatisfiable, no input string achieves a weight above the threshold ξ. When $F(s) = 1$, $w_s = \frac{e^{m'}}{2e^{-m'}+e^{m'}} \cdot \left(\frac{e^{m'}}{2e^{m'}+e^{-m'}}\right)^{m+2} > \xi$. An input string with weight above ξ exists if and only if F is satisfiable. Obviously, the reduction can be computed in polynomial time since all constants can be computed in logarithmic space. The constructed RNN is consistent, since the output prediction is constant after $m+3$ steps. □

5 Equivalence

We prove that equivalence of two RNNs is undecidable. For comparison, equivalence of two deterministic WFSAs can be tested in time $O(|\Sigma|(|Q_A| + |Q_B|)^3)$, where $|Q_A|$, $|Q_B|$ are the number of states of the two WFSAs and $|\Sigma|$ is the size of the alphabet (Cortes et al., 2007); equivalence of nondeterministic WFSAs are undecidable (Griffiths, 1968). The decidability of language equivalence for deterministic probabilistic pushdowntown automata (PPDA) is still open (Forejt et al., 2014), although equivalence for deterministic unweighted push-downtown automata (PDA) is decidable (Sénizergues, 1997).

The equivalence problem is formulated as follows:

Equivalence: Given two RNNs R and R', return "yes" if $R(s) = R'(s)$ for all $s \in \Sigma^*$, and "no" otherwise.

Theorem 12. *The equivalence problem for RNNs is undecidable.*

Proof. We prove by contradiction. Suppose Turing machine M decides the equivalence problem. Given any deterministic Turing Machine M', construct the RNN R that simulates M' on input ϵ as described in Corollary 5. Let $E_{s,t}(a) = 0$ and $E_{s,t}(\$) = h_{s,t}(n_1) - h_{s,t}(n_2)$. If M' does not halt on ϵ, for all $t \in \mathbb{N}$, $E'_{s,t}(a) = E'_{s,t}(\$) = 1/2$; if M' halts after T steps, $E'_{s,T}(a) = 1/(e+1)$, $E_{s,T}(\$) = e/(e+1)$. Let R' be the trivial RNN that computes $\{a^n : P(a^n) = 2^{-(n+1)}, n \geq 0\}$. We run M on input $\langle R, R' \rangle$. If M returns "no", M' halts on x, else it does not halt. Therefore the Halting Problem would be decidable if equivalence is decidable. Therefore equivalence is undecidable. □

6 Minimization

We look next at minimization of RNNs. For comparison, state-minimization of a deterministic PFSA is $O(|E| \log |Q|)$ where $|E|$ is the number of transitions and $|Q|$ is the number of states (Aho et al., 1974). Minimization of a non-deterministic PFSA is PSPACE-complete (Jiang and Ravikumar, 1993).

We focus on minimizing the number of hidden neurons ($|N|$) in RNNs:

Minimization: Given RNN R and non-negative integer n, return "yes" if $\exists$ RNN R' with number of hidden units $|N'| \leq n$ such that $R(s) = R'(s)$ for all $s \in \Sigma^*$, and "no" otherwise.

Theorem 13. *RNN minimization is undecidable.*

Proof. We reduce from the Halting Problem. Suppose Turing Machine M decides the minimization problem. For any Turing Machine M', construct the same RNN R as in Theorem 12. We run M on input $\langle R, 0 \rangle$. Note that an RNN with no hidden unit can only output constant $E'_{s,t}$ for all t. Therefore the number of hidden units in R can be minimized to 0 if and only if it always outputs $E'_{s,t}(a) = E'_{s,t}(\$) = 1/2$. If M returns "yes", M' does not halt on ϵ, else it halts. □

7 Conclusion

We proved the following hardness results regarding RNN as a recognizer of weighted languages:

1. Consistency:
   (a) Inconsistent RNNs exist.
   (b) Consistency of RNNs is undecidable.
2. Highest-weighted string:
   (a) Finding the highest-weighted string for an arbitrary RNN is undecidable.
   (b) Finding the highest-weighted string for a consistent RNN is decidable, but the solution length can surpass all computable bounds.
   (c) Restricting to solutions of polynomial length, finding the highest-weighted string is NP-complete.
3. Testing equivalence of RNNs and minimizing the number of neurons in an RNN are both undecidable.

Although our undecidability results are upshots of the Turing-completeness of RNN (Siegelmann and Sontag, 1995), our NP-completeness result is original, and surprising, since the analogous hardness results in PFSA relies on the fact that there are multiple derivations for a single string (Casacuberta and de la Higuera, 2000). The fact that these results hold for the relatively simple RNNs we used in this paper suggests that the case would be the same for more complicated models used in NLP, such as long short term memory networks (LSTMs; Hochreiter and Schmidhuber 1997).

Our results show the non-existence of (efficient) algorithms for interesting problems that researchers using RNN in natural language processing tasks may have hoped to find. On the other hand, the non-existence of such efficient or exact algorithms gives evidence for the necessity of approximation, greedy or heuristic algorithms to solve those problems in practice. In particular, since finding the highest-weighted string in RNN is the same as finding the most-likely translation in a sequence-to-sequence RNN decoder, our NP-completeness result provides some justification for employing greedy and beam search algorithms in practice.

Acknowledgments

This work was supported by DARPA (W911NF-15-1-0543 and HR0011-15-C-0115). Andreas Maletti was financially supported by DFG Graduiertenkolleg 1763 (QuantLA).

References

Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. 1974. *The design and analysis of computer algorithms*. Addison-Wesley.

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. *OpenFst: A General and Efficient Weighted Finite-State Transducer Library*, Springer Berlin Heidelberg, Berlin, Heidelberg, pages 11–23.

D. Bahdanau, K. Cho, and Y. Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *Proc. ICLR*.

T. L. Booth and R. A. Thompson. 1973. Applying probability measures to abstract languages. *IEEE Transactions on Computers* C-22(5):442–450. https://doi.org/10.1109/t-c.1973.223746.

Francisco Casacuberta and Colin de la Higuera. 2000. Computational complexity of problems on probabilistic grammars and transducers. *Grammatical Inference: Algorithms and Applications Lecture Notes in Computer Science* pages 15–24. https://doi.org/10.1007/978-3-540-45257-7_2.

Corinna Cortes, Mehryar Mohri, and Ashish Rastogi. 2007. L_p distance and equivalence of probabilistic automata. *International Journal of Foundations of Computer Science* 18(04):761–779. https://doi.org/10.1142/s0129054107004966.

Manfred Droste, Werner Kuich, and Heiko Vogler. 2013. *Handbook of Weighted Automata*. Springer Berlin.

Vojtch Forejt, Petr Janar, Stefan Kiefer, and James Worrell. 2014. Language equivalence of probabilistic pushdown automata. *Information and Computation* 237:1–11. https://doi.org/10.1016/j.ic.2014.04.003.

T. V. Griffiths. 1968. The unsolvability of the equivalence problem for Λ-free nondeterministic generalized machines. *Journal of the ACM* 15(3):409–413. https://doi.org/10.1145/321466.321473.

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

Tao Jiang and B. Ravikumar. 1993. Minimal NFA problems are hard. *SIAM Journal on Computing* 22(6):1117–1141. https://doi.org/10.1137/0222067.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. https://arxiv.org/pdf/1602.02410.pdf.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1317–1327. https://aclweb.org/anthology/D16-1139.

Donald E. Knuth. 1977. A generalization of Dijkstra's algorithm. *Information Processing Letters* 6(1):1–5. https://doi.org/10.1016/0020-0190(77)90002-3.

T. Mikolov and G. Zweig. 2012. Context dependent recurrent neural network language model. In *2012 IEEE Spoken Language Technology Workshop (SLT)*. pages 234–239. https://doi.org/10.1109/SLT.2012.6424228.

Mark-Jan Nederhof and Giorgio Satta. 2006. Estimation of consistent probabilistic context-free grammars. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*. Association for Computational Linguistics, New York City, USA, pages 343–350. http://www.aclweb.org/anthology/N/N06/N06-1044.

Tibor Radó. 1962. On non-computable functions. *Bell System Technical Journal* 41:877–884.

Géraud Sénizergues. 1997. The equivalence problem for deterministic pushdown automata is decidable. In *Proc. Automata, Languages and Programming: 24th International Colloquium*, Springer Berlin Heidelberg, pages 671–681.

Hava T. Siegelmann and Eduardo D. Sontag. 1995. On the computational power of neural nets. *Journal of Computer and System Sciences* 50(1):132–150. https://doi.org/10.1006/jcss.1995.1013.

Khalil Simaan. 1996. Computational complexity of probabilistic disambiguation by means of tree-grammars. In *Proc. COLING*. pages 1175–1180. https://doi.org/10.3115/993268.993392.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., pages 3104–3112.

Appendix

Lemma 14. *For every* $k \in \mathbb{N}_+$

$$\prod_{t\in\mathbb{N}} \frac{e^{k(t+1)}}{e^{k(t+1)}+1} = \frac{2}{(-1;\, e^{-k})_\infty} \ ,$$

where $(-1; e^{-k})_\infty$ *is the infinite* e^{-k}*-Pochhammer symbol.*

Proof.

$$\prod_{t\in\mathbb{N}} \frac{e^{k(t+1)}}{e^{k(t+1)}+1} = \prod_{t\in\mathbb{N}_+} \left(\frac{e^{kt}}{e^{kt}+1}\cdot\frac{e^{-kt}}{e^{-kt}}\right)$$

$$= \prod_{t\in\mathbb{N}_+} \frac{1}{1+e^{-kt}} = \left(\left(\prod_{t\in\mathbb{N}_+} \frac{1}{1+e^{-kt}}\right)^{-1}\right)^{-1}$$

$$= \Big(\prod_{t\in\mathbb{N}_+} (1+e^{-kt}) \Big)^{-1} = \Big(\frac{1}{2} \prod_{t\in\mathbb{N}} (1+e^{-kt}) \Big)^{-1}$$
$$= \frac{2}{(-1;\, e^{-k})_\infty} \qquad \square$$

Lemma 15. *Reconsider the RNN of Example 3 and suppose that $h_{s,t}(n) - h_{s,t}(n') = 0$ for all $t \in \mathbb{N}$. Then*

$$\sum_{s\in\Sigma^*} R(s) = 1 - \frac{2}{(-1;\, e^{-2})_\infty} \approx 0.14$$

Proof.

$$\sum_{s\in\Sigma^*} R(s) = \sum_{n\in\mathbb{N}} R(a^n)$$
$$= \sum_{n\in\mathbb{N}} \Big(\frac{e^{-(n+1)}}{e^{n+1}+e^{-(n+1)}} \cdot \prod_{t=0}^{n-1} \frac{e^{t+1}}{e^{t+1}+e^{-(t+1)}} \Big)$$
$$= 1 - \prod_{t\in\mathbb{N}} \frac{e^{2(t+1)}}{e^{2(t+1)}+1} = 1 - \frac{2}{(-1;\, e^{-2})_\infty}$$
$$\approx 0.14 \ ,$$

where the final equality utilizes Lemma 14. $\square$

Lemma 16. *Reconsider the RNN of Example 3 and suppose that there exists a time point $T \in \mathbb{N}$ such that for all $t \in \mathbb{N}$*

$$h_{s,t}(n) - h_{s,t}(n') = \begin{cases} 1 & \text{if } t = T \\ 0 & \text{otherwise.} \end{cases}$$

Then

$$\sum_{s\in\Sigma^*} R(s) = 1$$

Proof.

$$\sum_{s\in\Sigma^*} R(s) = \sum_{n\in\mathbb{N}} R(a^n)$$
$$= \Big(\sum_{n=0}^{T} R(a^n) \Big) + \Big(\sum_{n=T+1}^{\infty} R(a^n) \Big)$$
$$= \sum_{n=0}^{T} \Big(\frac{e^{-(n+1)}}{e^{n+1}+e^{-(n+1)}} \cdot \prod_{t=0}^{n-1} \frac{e^{t+1}}{e^{t+1}+e^{-(t+1)}} \Big)$$
$$+ \sum_{n=T+1}^{\infty} \frac{e^{2n-3T-4}}{e^{n+1}+e^{2n-3T-4}}$$
$$\cdot \Big(\prod_{t=0}^{T} \frac{e^{t+1}}{e^{t+1}+e^{-(t+1)}} \Big) \cdot \Big(\prod_{t=T+1}^{n-1} \frac{e^{t+1}}{e^{t+1}+e^{2t-3T-4}} \Big)$$
$$= \sum_{n=0}^{T} \Big(\frac{1}{e^{2(n+1)}+1} \cdot \prod_{t=0}^{n-1} \frac{e^{2(t+1)}}{e^{2(t+1)}+1} \Big)$$
$$+ \sum_{n=T+1}^{\infty} \frac{e^{n-3T-5}}{1+e^{n-3T-5}} \cdot \Big(\prod_{t=0}^{T} \frac{e^{2(t+1)}}{e^{2(t+1)}+1} \Big)$$
$$\cdot \Big(\prod_{t=T+1}^{n-1} \frac{1}{1+e^{t-3T-5}} \Big)$$
$$= 1 - \Big(\prod_{t=0}^{T} \frac{e^{2(t+1)}}{e^{2(t+1)}+1} \Big) + \Big(\prod_{t=0}^{T} \frac{e^{2(t+1)}}{e^{2(t+1)}+1} \Big)$$
$$\cdot \sum_{n=T+1}^{\infty} \frac{e^{n-3T-5}}{1+e^{n-3T-5}} \cdot \Big(\prod_{t=T+1}^{n-1} \frac{1}{1+e^{t-3T-5}} \Big)$$
$$= 1 - \Big(\prod_{t=0}^{T} \frac{e^{2(t+1)}}{e^{2(t+1)}+1} \Big)$$
$$\cdot \left(1 - 1 + \prod_{t=T+1}^{\infty} \frac{1}{1+e^{t-3T-5}} \right)$$
$$= 1 - \Big(\prod_{t=0}^{T} \frac{e^{2(t+1)}}{e^{2(t+1)}+1} \Big) \cdot \Big(\prod_{t=T+1}^{\infty} \frac{1}{1+e^{t-3T-5}} \Big)$$
$$\geq 1 - \Big(\prod_{t=0}^{T} \frac{e^{2(t+1)}}{e^{2(t+1)}+1} \Big) \cdot \Big(\prod_{t\in\mathbb{N}} \frac{1}{1+e^{t}} \Big)$$
$$= 1 \qquad \square$$

Lemma 17.

Proof. We set $h_{s,-1}(n) = h_{-1}(n)$ for all $n \in N$ and $h'_{s,-1}(n') = h'_{-1}(n')$ for all $n' \in N'$. Then trivially $h'_{s,-1}(n_1) - h'_{s,-1}(n_2) = h_{-1}(n) - 0 = h_{s,-1}(n)$. Moreover, $h'_{s,t}(n_1) = \sigma\langle V \cdot h'_{s,t-1} + V'_{s[t]} \rangle(n_1)$

$$= \sigma\langle \sum_{n'\in N'} V(n_1,n') \cdot h'_{s,t-1}(n')$$
$$+ V'_{s[t]}(n_1) \rangle$$
$$= \sigma\langle \sum_{n'\in N} \big(V(n_1,n'_1) \cdot h'_{s,t-1}(n'_1)$$
$$+ V(n_1,n'_2) \cdot h'_{s,t-1}(n'_2) \big) + V'_{s[t]}(n_1) \rangle$$
$$= \sigma\langle \sum_{n'\in N} U(n,n') \cdot \big(h'_{s,t-1}(n'_1) - h'_{s,t-1}(n'_2) \big)$$
$$+ U'_{s[t]}(n) \rangle$$
$$= \sigma\langle \sum_{n'\in N} U(n,n') \cdot h_{s,t-1}(n') + U'_{s[t]}(n) \rangle$$

Similarly, we can show that $h'_{s,t}(n_2) =$

$$\sigma\langle \sum_{n'\in N} U(n,n') \cdot h_{s,t-1}(n') + U'_{s[t]}(n) - 1 \rangle$$

Hence $h'_{s,t}(n_1) - h'_{s,t}(n_2) = h_{s,t}(n)$ as required. $\square$

Author Index